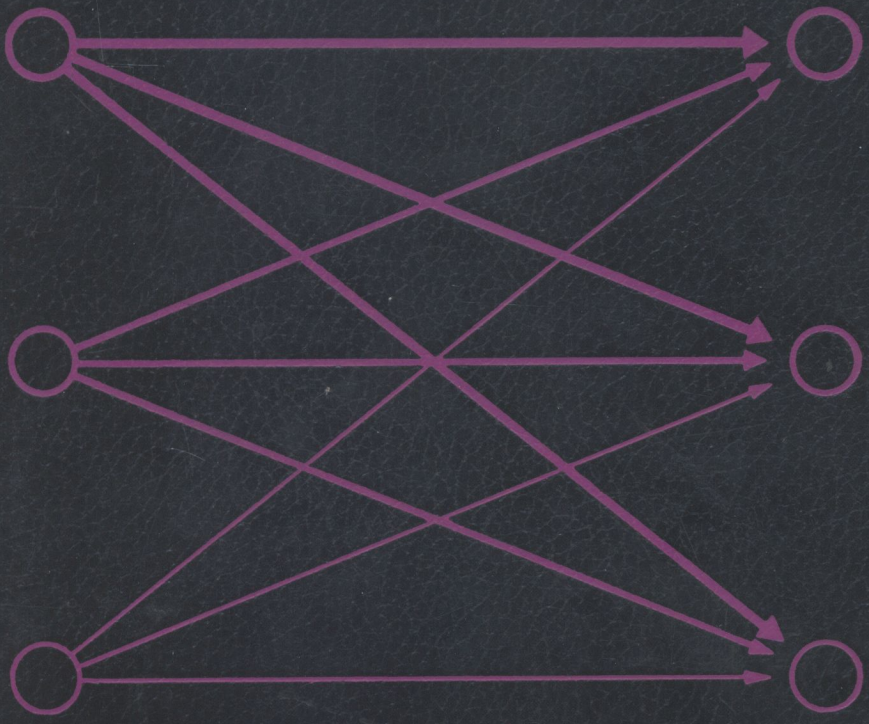
Д. ФИЛЛИПС,
А. ГАРСИА-ДИАС

МЕТОДЫ АНАЛИЗА СЕТЕЙ




ИЗДАТЕЛЬСТВО
МИР

FUNDAMENTALS OF NETWORK ANALYSIS

Don T. Phillips
Department of Industrial Engineering
Texas Transportation Institute
Texas A & M University

Alberto Garcia-Diaz
Department of Industrial Engineering
Texas A & M University
College Station, Texas 77843

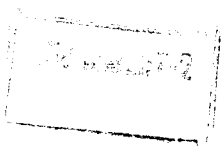
Prentice-Hall, Inc., Englewood Cliffs, N. J. 07632
1981

Д. ФИЛЛИПС,
А. ГАРСИА-ДИАС

МЕТОДЫ АНАЛИЗА СЕТЕЙ

Перевод с английского
Е. Г. Коваленко, М. Г. Фуругяна,

под редакцией
Б. Г. Сушкова



Москва «Мир» 1984

ББК 22.18
Ф52
УДК 519.95:62-50

Филлипс Д., Гарсиа-Диас А.
Ф52 Методы анализа сетей: Пер. с англ. — М.: Мир, 1984. — 496 с., ил.

В книге американских ученых излагаются методы и алгоритмы оптимизации детерминированных и стохастических сетей различного назначения с помощью теории графов. Книга иллюстрирована большим числом примеров, взятых из различных областей науки и техники.

Для специалистов, занимающихся применением вычислительной техники в экономике, планировании, биологии и медицине. Может быть использована аспирантами и студентами соответствующих специальностей.

Ф $\frac{1502000000\text{-}242}{041(01)\text{-}84}$ 7-84, ч. 1

ББК 22.18
518

Редакция литературы по новой технике

Дон Т. Филлипс, Альберто Гарсиа-Диас
МЕТОДЫ АНАЛИЗА СЕТЕЙ

Научный редактор Т. Н. Шестакова
Младший научный редактор Е. П. Орлова
Художественный редактор В. Б. Прищепа
Технический редактор Т. А. Максимова
Корректор С. А. Денисова

ИБ № 3709

Сдано в набор 31.10.83. Сдано в печать 14.03.84.
Бумага типографская № 2. Формат 60×90¹/₁₆. Гарнитура латинская.
Печать высокая. Объем 15,5 бум. л. Усл. печ. л. 31,0. Уч.-изд. л. 28,65.
Усл. кр.-отт. 31,0. Тираж 10 000 экз. Цена 2 р. 30 к. Зак. 1654.
Издательство «Мир». 129820. ГСП, Москва, И-110, 1-й Рижский пер., 2.
Московская типография № 11 Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли.
113105, Москва, Нагатинская ул., д. 1.

ПРЕДИСЛОВИЕ РЕДАКТОРА ПЕРЕВОДА

Успехи прикладной математики в количественном анализе сложных реальных систем и явлений существенно определяются двумя обстоятельствами: возможностью построения математической модели исследуемого процесса, адекватно отображающей реальность, и существованием математических средств исследования построенной модели. По мере накопления все большего числа удачных попыток применения тех или иных математических объектов для изучения реальных событий наступает качественно новый этап в развитии конкретной области науки: появляется формальный математический язык для описания исследуемых явлений. Формальные объекты этого языка становятся предметом исследования математиков, зачастую полностью абстрагирующихся от предметной области, к которой эти объекты поначалу относились, появляются обладающие большой общностью методы расчета этих объектов. Негативной стороной этого процесса является усложнение вводимых понятий и построений, затрудняющее использование полученных результатов специалистами из конкретных областей, не обладающими профессиональной математической подготовкой.

Подобную эволюцию испытали и методы анализа сетей, берущие свое начало из сформулированных Г. Р. Кирхгофом законов протекания электрического тока в системах разветвляющихся проводников. В настоящее время область применения этих методов чрезвычайно широка. К примеру, она включает в себя проектирование сложных систем связи, исследование транспортных потоков, передачу информации в вычислительных системах, анализ путей материального снабжения современных заводов, финансовые операции. Соответственно, развились и усложнились методы численного анализа сетевых моделей.

Предлагаемая вниманию читателей книга является удачным сочетанием полноты, строгости, ясности и доступности в изложении. Авторы добились этого двумя путями. С одной стороны, книга содержит описание практически всех современных методов исследования сетей. Причем большинство фундаментальных свойств этих методов строго доказано и обосновано. С другой стороны, в книге приведено большое количество конкретных практических задач из различных областей человеческой деятельности. Чрезвычайно важно, что в этих примерах представлены все этапы решения задачи, включая ее постановку, построение соответствующей сетевой модели, выбор алгоритма решения, получение численных результатов и их анализ с целью выработки рекомендаций, улучшающих исследуемую систему. Многочисленные упражнения должны стимулировать читателя к самостоятельному проведению подобной работы для анализа самых различных объектов и явлений, начиная от выбора рационального маршрута развлекательной поездки вплоть до решения таких серьезных практических задач, как проектирование разветвленных систем транспортировки нефти от приисков к нефтеперерабатывающим заводам и потребителям. Весьма ценным приложением является библиотека стандартных программ на языке ФОРТРАН, реализующая наиболее употребительные методы анализа сетей.

Каждая глава книги содержит обширную библиографию, включающую основополагающие работы по проблематике данной главы. К сожалению, в этой библиографии практически не отражены достижения советской школы

дискретной оптимизации. Для примера можно привести результаты В. С. Танаева, В. В. Шкурбы, Е. Г. Гольштейна, Д. Б. Юдина, В. С. Михалевича, А. И. Куксы, А. А. Корбута, Э. Г. Давыдова, Г. М. Адельсона-Вельского и многих других.

Можно с уверенностью утверждать, что книга послужит хорошим подспорьем для математиков, специализирующихся в области разработки методов дискретного анализа, руководством к практическому применению сетевых методов для специалистов в прикладных областях, учебным пособием для студентов и аспирантов, изучающих численные методы анализа сетей.

Б. Г. Сушков

Перевод книги выполнили М. Г. Фуругян (гл. 1—3, 5) и Е. Г. Коваленко (гл. 4).

ПРЕДИСЛОВИЕ

Эта книга является работой, в которой всесторонне рассмотрены вопросы теории и методов вычислений детерминированных потоков в сетях. Книга написана на основе лекций, которые читались в течение 10 лет и были составлены самими авторами, а также с использованием многочисленных научных статей и трудов ведущих специалистов в этой области. Основная направленность книги — практический подход к разработке и реализации потоковых алгоритмов. Рискуя подвергнуться критике со стороны своих коллег, мы иногда не останавливались на изложении теоретических результатов и проведении математических доказательств, а рассматривали лишь вычислительную сторону изучаемого вопроса. При рассмотрении каждой задачи вначале, если это уместно, дается ее формулировка в виде задачи линейного программирования, а затем описывается алгоритм ее решения, являющийся более быстрым и более эффективным по сравнению с процедурой, основанной на непосредственном использовании математической модели. Каждый алгоритм используется для решения одной или нескольких практических задач. Особенностью книги является наличие в ней описаний и листингов программ сетевой оптимизации, написанных на языке ФОРТРАН IV, которые применимы к решению задач малой и средней размерности. Программы могут быть использованы как в целях обучения, так и при проведении исследовательской работы.

Эта книга представляет собой введение в теорию сетевых потоков и окажется полезной студентам и аспирантам в качестве учебника по вводному курсу сетевого анализа. Для понимания материала, содержащегося в книге, не требуется специальной математической подготовки. Однако полезным было бы некоторое знакомство с обозначениями, используемыми в линейном программировании, и языком ФОРТРАН, но и оно не является необходимым для понимания описанных алгоритмов.

Нам было приятно отметить, что за последние 10 лет интерес к теории и приложениям потоковых алгоритмов значительно возрос. В области образования, по-видимому, не существует такого предмета, который не включал бы в себя сетевой анализ; он постоянно находит применение в управлении частными предприятиями и компаниями, во всех технических науках, при проектировании транспортных систем, в области планирования и управления работами над проектом, при составлении расписаний и во многих других областях. Основное достоинство сетевых моделей заключается в их гибкости и возможности графического их описания. Кроме того, сетевые процедуры поиска решений, которые были разработаны сравнительно недавно, являются значительно более эффективными, чем обычные методы линейного программирования.

Основным препятствием на пути широкого распространения и применения сетевого анализа, безусловно, являются трудности, возникающие при формализации языка. За небольшим исключением, все предыдущие научные и технические разработки в данной области велись на основе теории графов и математического программирования. Полученные результаты обычно излагались в научных статьях и докладах. В этой книге мы стремились избежать излишней математической строгости и рассмотреть основные вопросы сетевого анализа неформальным образом.

Книга состоит из пяти глав и приложения. В гл. 1 вводятся используемые в дальнейшем обозначения и даются определения, связанные с со-

держанием последующих глав. Гл. 2 посвящена всестороннему изучению детерминированных потоков в сетях. Она начинается с рассмотрения ряда примеров, иллюстрирующих разнообразие сетевых постановок практических задач. В ней решается большое число различных примеров, а для решения практических задач большей размерности дается описание программ, написанных на языке ФОРТРАН IV. Гл. 3 содержит унифицированное и исчерпывающее описание изящного алгоритма дефекта, а также подробное рассмотрение вопросов теоретического и вычислительного характера, связанных с этим мощным методом. В этой главе читателю предлагаются многочисленные приложения метода, а для иллюстрации процедур построения модели рассматривается несколько задач. В гл. 4 дается полное описание процедур планирования и управления проектом, основанных на системах ПЕРТ и МКП. В этой главе подробно рассматриваются вопросы распределения ресурсов и регулирования потребления, а также описываются методы вычислений и машинные процедуры. В гл. 5 изучаются общие постановки задач: потоки в сетях с выигрышами и проигрышами, ГЕРТ-процедуры для стохастических сетей, имеющих специальную структуру, и многопродуктовые потоки. В приложении дается полный текст программ сетевой оптимизации и инструкции по их использованию.

Мы стремились использовать знания и опыт многих специалистов в этой области. Некоторые из них непосредственно участвовали в написании книги. Часть материала гл. 2 и, в частности, многие практические примеры были предоставлены нам доктором Г. Е. Беннингтоном. Часть материала гл. 3, касающегося алгоритма дефекта, была взята из лекций доктора Поля А. Йенсена, а некоторые примеры были предоставлены нам доктором Вулсеем и доктором Хантером Суонсоном. Бо́льшую часть материала, касающегося вычислительных аспектов систем ПЕРТ и МКП (гл. 4, ч. I), предоставил доктор Уорэн Томас. Весь материал раздела гл. 4, связанный с управлением ресурсами, был предоставлен доктором Эдвардом Дэвисом, а материал гл. 4, касающийся программного обеспечения, был взят из статьи доктора Лэри А. Смита и Питера Малера. Материал гл. 5, связанный с теорией обобщенных сетей, был взят из работ доктора Г. Бомика и доктора П. Йенсена, а некоторые примеры — из работ доктора Дарвина Клингмана и доктора Фреда Глоувера. Теоретические результаты и методы вычислений, касающиеся системы ГЕРТ, принадлежат доктору А. Алану Б. Притскеру. И наконец, весь материал, связанный с многопродуктовыми потоками в сетях, был предоставлен доктором Джеймсом Эвансом.

Помимо того что ряд специалистов оказал нам непосредственную помощь при написании гл. 3 и 4, в этой книге использованы материалы научных статей, написанных многими нашими коллегами, не всех из которых мы смогли здесь упомянуть. Они сами увидят результат своего личного вклада. Мы благодарны им за ту помощь, которую оказала нам их работа. В заключение нам хотелось бы особо отметить большой личный вклад доктора Поля А. Йенсена и доктора Вулсея. Они поймут и примут нашу особую благодарность за их неоценимую поддержку при написании книги. Особую благодарность за всестороннюю критику и участие при написании книги заслуживает доктор Джеймс Эванс. Доктор Эванс прочитал первый вариант рукописи и сделал много ценных замечаний, за которые мы ему очень признательны. Мы были бы несправедливы, если бы не поблагодарили Джен Бертч и Кэнди Филлипс, перепечатавших рукопись и испытавших немало волнений, когда материал был собран. Мы также в большом долгу перед Американским институтом инженеров-технологов, давшим разрешение на перепечатку опубликованного материала.

Дон Т. Филлипс, доктор философии, инженер-нефтяник
Альберто Гарсиа-Диас, доктор философии

Колледж-Стейшен, Техас

Глава 1

ВВЕДЕНИЕ

— Чеширский Мурлыка..., — заговорила Алиса несмело.
— Скажите, пожалуйста, куда мне отсюда идти?
— Это во многом зависит от того, куда ты хочешь прийти, — ответил Кот.
— Да мне почти все равно, — начала Алиса.
— Тогда все равно, куда идти, — сказал Кот.
— Лишь бы попасть куда-нибудь, — пояснила Алиса.

Л. Кэрролл. Приключения Алисы в стране чудес[1]

По остроумному замечанию Чеширского Кота вперед можно продвинуться, даже если просто скитаться без определенной цели. Однако для поиска оптимального решения часто существуют более разумные методы, чем тот, который избрала Алиса, блуждая по стране чудес. В частности, сетевое моделирование предоставляет такие теоретические результаты и вычислительные алгоритмы, которые позволяют в значительной степени усовершенствовать ряд традиционных подходов системного анализа. В настоящей главе вводятся понятия, необходимые для понимания и практического применения основных алгоритмов, содержащихся в данной книге. Конечно, при практическом использовании методов сетевого анализа часто встает не только вопрос о том, «до какого места Вы хотите дойти», но и вопрос о том, «какой путь следует избрать». Надеемся, что настоящая глава поможет при ответе на оба этих вопроса.

Современное общество отчасти можно рассматривать как систему сетей, предназначенных для транспортирования, передачи и распределения электроэнергии, товаров и информации. Поскольку каждая из подсистем имеет весьма сложную структуру и является дорогостоящей, возникает необходимость в эффективном использовании уже существующих технических средств и в рациональном проектировании новых средств. При проектировании и усовершенствовании больших и сложных систем, а также при поиске путей их наиболее рационального использования существенное значение могут иметь методы сетевого анализа.

---

[1] Перевод Бориса Заходера.

Методы решения потоковых задач являются мощным математическим средством, с помощью которого могут быть сформулированы и решены многие технические задачи. Наглядность и логическая обоснованность этих методов нередко позволяет выработать новый и довольно естественный подход к решению поставленной задачи, позволяющий определить пути последующего прикладного анализа. Методы сетевого анализа, занимавшие некогда лишь незначительное место в исследовании операций, в настоящее время стали легко реализуемыми на ЭВМ и широко используются на практике при решении важных задач, стоящих перед современными техническими дисциплинами.

Сетевой анализ возник в результате многочисленных исследований. В значительной степени он основан на теории графов — области математики, источником развития которой явилась известная задача о Кенигсбергских мостах, сформулированная и решенная Леонардом Эйлером в 1736 г. [7]. Спустя более века Джеймс Клерк Максвелл и Густав Роберт Кирхгоф, исследуя электрические цепи, сформулировали некоторые основные принципы сетевого анализа. С тех пор сетевой анализ стал широко использоваться при изучении электрических систем. В начале XX в. европейскими и американскими инженерами были разработаны методы расчета наибольшей пропускной способности телефонных линий и коммутаторов, позволяющие обеспечить гарантированное обслуживание определенного числа абонентов. В сороковых годах, во время второй мировой войны, в результате развития теории исследования операций был получен ряд математических методов, необходимых для анализа больших систем. Первые работы по современному сетевому анализу были написаны Хитчкоком [11] в 1941 г. и Купмансом [15] в 1947 г. С тех пор в области сетевого анализа было проведено большое количество плодотворных исследований и опубликовано свыше тысячи печатных работ. В пятидесятых и начале шестидесятых годов основное внимание уделялось построению новых моделей и разработке новых алгоритмов. Позднее стали проводить более глубокие исследования построенных ранее моделей и искать пути реализации алгоритмов на вычислительной машине. Был издан ряд обзорных статей, написанных такими авторами, как Фалкерсон [10], Элмаграби [6], Брэдли [2], Магнанти и Голден [16] и др. С развитием сетевого анализа возникла необходимость в книгах, которые бы отражали различные направления исследований, проводимых в данной области. Ряд книг, содержащих наиболее полное изложение вопросов, связанных с сетями, написали Форд и Фалкерсон [8], Чарнес и Купер [4], Данциг [5], Басакер и Саати [3], Ху [12], Фрэнк и Фриш [9],

Уайтхаус [21] и позднее Йенсен и Барнес [14], Базарра и Джэрвис [1], Майника [17].

Сетевые модели широко применяются в иследовании операций и могут быть использованы на практике, например, при проектировании больших оросительных систем, вычислительных комплексов, транспортных сетей, телетрансляционных сетей, систем космической связи. Для решения практических задач, связанных со складированием и распределением товаров, календарным планированием, заменой оборудования, контролем за уровнем издержек, перевозками, работой систем массового обслуживания, обеспечением ритмичности производственного процесса, управлением запасами, распределением рабочей силы, а также многих других задач были разработаны эффективные алгоритмы, основанные на методах сетевого анализа.

Притскер [20] дает следующее объяснение широкому применению на практике методов сетевого анализа:

«Сетевой анализ играет всевозрастающую роль как при описании управляющих систем, так и при их усовершенствовании. В первую очередь это связано с тем, что с помощью сетей можно довольно просто построить модель системы. Кроме того, расширение области использования сетей связано с тем, что методы сетевого анализа позволяют

1. Построить модель сложной системы как совокупности простых систем.

2. Составить формальные процедуры для определения качественных характеристик системы.

3. Указать механизм взаимодействия компонентов управляющей системы с целью описания последней в терминах ее основных характеристик.

4. Определить, какие данные необходимы для исследования системы.

5. Провести начальные исследования управляющей системы и составить предварительное расписание работы ее компонентов.

Впервые сети стали использовать благодаря их возможностям, указанным в п. 5. Благодаря преимуществам, которые дают аналитические процедуры, сетевой подход утвердился как метод исследования. В настоящее время большое число работ имеет целью углубить существующие знания в области аналитических процедур, что необходимо для решения новых сетевых задач, постоянно возникающих на практике».

Сетевой анализ не является дисциплиной, принадлежащей какой-то одной области науки или отрасли производства. Основное достоинство сетевого подхода заключается в том, что он может быть успешно применен к решению практически лю-

бой задачи, когда исследователь обладает необходимыми знаниями и способностью точно построить сетевую модель. Преимущества использования сетевых моделей можно сформулировать следующим образом:

1. Сетевые модели могут точно описать многие реально существующие системы.

2. Для людей, не занимающихся научной работой, сетевые модели являются, вероятно, более понятными, чем любые другие модели, используемые в исследовании операций. По-видимому, это следует из того, что «лучше один раз увидеть, чем сто раз услышать». Руководителю предприятия, наверное, легче понять сетевую диаграмму, чем абстрактные формулы. Кроме того, связанные, как правило, с конкретными прикладными задачами сетевые модели являются несложными для работников, не имеющих специального математического образования.

3. Сетевые алгоритмы позволяют находить наиболее эффективные решения при изучении некоторых больших систем.

4. По сравнению с другими методами оптимизации сетевые алгоритмы нередко позволяют решать задачи со значительно бо́льшим числом переменных и ограничений. Это становится возможным благодаря тому, что при использовании методов сетевого анализа часто удается ограничиться изучением лишь части рассматриваемой системы.

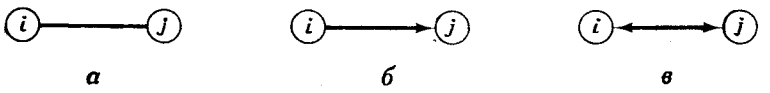
1.1. ОПРЕДЕЛЕНИЯ И ОБОЗНАЧЕНИЯ

Сеть состоит из множества *узлов* и множества *дуг*, соединяющих их. Узлы иногда называют вершинами или точками, а дуги — ребрами, звеньями, линиями или ветвями. Сеть может быть представлена в виде $G=(\mathbf{N}, \mathbf{A})$, где $\mathbf{N}$ — множество узлов сети G, $\mathbf{A}$ — множество дуг сети G.

Примерами узлов сети могут быть пересечения автострад, электростанции, телефонные узлы, железнодорожные парки, аэропорты, водохранилища, вычислительные машины, точно определенные во времени события, или попросту «километровые столбы» проекта. Вообще узлом может быть точка, являющаяся источником или стоком некоторого потока, а также точка, в которой величина этого потока изменяется. Поэтому узел можно рассматривать как точку ветвления в сети.

Примерами дуг могут быть дороги, линии электропередачи, телефонные линии, авиалинии, водные магистрали или обобщенные каналы, через которые протекает некоторый поток. Иногда дуги не имеют физического смысла, а используются для того, чтобы направить поток через узлы в нужной логической последовательности или чтобы выполнялись определенные отношения предшествования.

Обычно узлам приписываются номера, чтобы различать их или иметь возможность ими оперировать в процессе анализа. Очевидно, что нумерацию узлов можно осуществлять произвольным образом[1]. Однако исследования могут упроститься, если их занумеровать в определенном порядке. Говорят, что дуга, соединяющая узлы i и j, направлена от i к j, если единица потока, посылаемого от узла i к узлу j по этой дуге, уве-



а *б* *в*

Рис. 1.1. Графическое изображение дуг и узлов.
а — неориентированная дуга; *б* — ориентированная дуга; *в* — биориентированная дуга.

личивает текущую величину потока в данной дуге, а единица потока противоположного направления уменьшает эту величину. Дуги сети могут быть неориентированными, ориентированными и биориентированными. *Неориентированной* называется дуга, не имеющая определенного направления, *ориентированной* — дуга, имеющая одно направление, и *биориентированной* — дуга, имеющая два направления.

а *б*

Рис. 1.2. Замена биориентированной дуги двумя противоположно направленными дугами.
а — биориентированная дуга; *б* — эквивалентное представление.

При графическом представлении сети узлы мы обозначаем кружками, дуги — отрезками прямой, а направления дуг указываем стрелками. Парой (i, j) мы обозначаем ориентированную дугу, направленную от узла i к узлу j. На рис. 1.1 изображены неориентированная, ориентированная и биориентированная дуги соответственно.

При решении большинства практических задач нет необходимости делать различие между неориентированными и биориентированными дугами. Кроме того, очевидно, что биориентированную дугу, соединяющую два произвольных узла i и j, можно заменить двумя ориентированными дугами (i, j) и (j, i) (рис. 1.2).

Как отмечалось выше, дуги сети можно рассматривать как каналы, по которым протекает поток некоторого обобщенного

[1] Интересное исключение из этого правила можно получить при анализе систем МКП и ПЕРТ. Как показано в гл. 4, нумерацию узлов лучше производить последовательно в возрастающем порядке.

продукта. Чистая величина этого продукта в заданном узле равна разности величин потока, втекающего в узел, и потока, вытекающего из него. Если эта разность положительна, то узел называется источником, а если отрицательна — стоком или конечным узлом сети. Сеть может содержать более одного стока или источника. Для удобства мы часто будем называть *источником* каждый узел, из которого поток только вытекает, а *стоком* — каждый узел, в который поток только втекает. Например, в сети, изображенной на рис. 1.3, узлы 1 и 2 являются источниками, а узлы 6 и 7 — стоками. При описании некоторых алгоритмов предполагается, что сеть содержит только один источник и один сток. Такую сеть можно построить, вводя *главный источник* и *главный сток* и затем соединяя фиктивными дугами главный источник со всеми источниками, а главный сток — со всеми стоками.

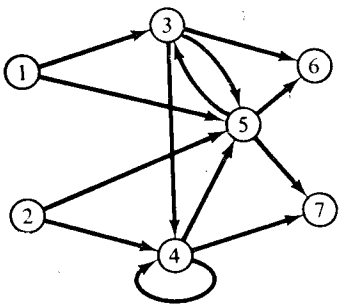


Рис. 1.3. Пример сети.

При сведении практических задач к потоковым задачам нередко бывает полезно разбить сеть на составные части исходя из направления потока. В связи с этим следует ввести часто используемое в теории сетей понятие, связанное с соединением двух узлов i и j последовательностью дуг $\mathbf{E}=\{a_1, a_2, ..., a_k\}$. *Цепью* из угла i в узел j называется последовательность дуг и узлов, в которой конечный узел каждой дуги является начальным узлом следующей (исключая первый и последний узлы последовательности). Следовательно, каждая дуга в такой последовательности ориентирована по направлению от узла i к узлу j. Путь отличается от цепи лишь тем, что в нем одна или несколько дуг могут быть направлены не к узлу j, а к узлу i. Это означает, что в отличие от цепи в пути не все дуги имеют одинаковое направление. Согласно этим определениям, цепь не может содержать биориентированную дугу, но может содержать неориентированную дугу. Обычно цепи и пути различают лишь при изучении потоков в ориентированных дугах; в данной книге мы также будем придерживаться этого соглашения. В сети, изображенной на рис. 1.3, последовательность дуг (1, 3), (3, 5), (5, 3), (3, 6) образует цепь, а последовательность дуг (2, 4), (3, 4), (3, 6) — путь.

Сеть называется *неориентированной,* если ни одна из ее дуг не имеет ориентации. В неориентированной сети понятия цепи и пути являются взаимозаменяемыми и соответствуют последовательности дуг, соединяющих два произвольных узла.

Циклом называется конечный путь, начальный и конечный узлы которого совпадают. На рис. 1.3 последовательность дуг (4, 7), (4, 5), (5, 7) образует цикл. В частности, циклами являются контуры. *Контуром* называется конечная цепь, начальный и конечный узлы которой совпадают. Очевидно, что циклы

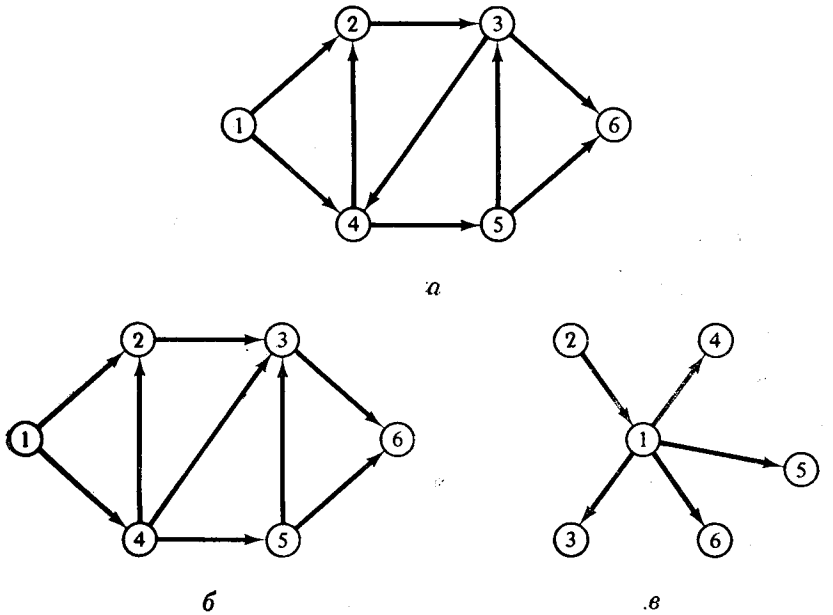


Рис. 1.4. Циклические и ациклические сети.
а — сеть, содержащая контуры; *б* — сеть, не содержащая контуров; *в* — ациклическая сеть.

являются замкнутыми путями, а контуры — замкнутыми цепями. *Петля* образуется одним узлом и одной дугой и поэтому является как контуром, так и циклом. В сети, изображенной на рис. 1.3, последовательность дуг (3, 4), (4, 5), (5, 3) образует контур. Вырожденный цикл, или петлю, иногда называют одноступенчатым циклом, поскольку он содержит только одну дугу. Дуга (4, 4) сети, изображенной на рис. 1.3, образует петлю. Говорят, что сеть является *бесконтурной,* если она не содержит контуров. Иными словами, сеть, содержащая только ориентированные дуги, называется *бесконтурной,* если ее дуги не образуют контуров. Такие сети обладают рядом специальных свойств, которые используются при решении некоторых практических задач, связанных с поиском кратчайших и максимальных путей. На этих свойствах основаны алгоритмы

в системах МКП и ПЕРТ. На рис. 1.4, *а, б* изображены сеть, имеющая контуры, и бесконтурная сеть соответственно. Сеть, изображенная на рис. 1.4, *в*, является *ациклической*, т. е. не содержит циклов.

Сеть называется *связной*, если для любых двух различных узлов существует по крайней мере один соединяющий их путь или цепь.

Частным случаем связных ориентированных и неориентированных сетей являются *деревья*. Напомним, что множество всех узлов и дуг задается в виде $G=(\mathbf{N}, \mathbf{A})$. Подмножество множества G определим как $\bar{G}=(\bar{\mathbf{N}}, \bar{\mathbf{A}})$. *Дерево* определяется как связное подмножество $\bar{G}$, не содержащее циклов. Следовательно, для любых двух узлов дерева существует единственный путь, соединяющий их. В сети, содержащей n узлов, подграф из k узлов ($k \leqslant n$) является деревом, если выполнены любые два из следующих условий:

1. Подграф является связным.
2. Подграф не имеет циклов.
3. Число дуг в подграфе равно $k-1$.

Остовным деревом[1] называется дерево, содержащее все узлы сети. Следовательно, если сеть содержит n узлов, то дерево с n узлами и $n-1$ дугой является остовным.

Если для указания некоторых естественных ограничений, накладываемых на дуги сети, сопоставить последним числа[2], соответствующие, например, расстоянию, стоимости или другим аддитивным величинам, то мы сможет ввести понятие *кратчайшего* (или максимального) *остова*. Вес дерева определяется как сумма весов его дуг. Кратчайшим (максимальным) остовом называется дерево с минимальным (максимальным) весом.

Деревья и остовы определяются как для ориентированных, так и для неориентированных сетей. Однако для сетей, содержащих только ориентированные дуги, следует ввести понятие древовидности. *Древовидность* можно представить себе как дерево, все дуги которого ориентированные. Кроме того, древовидность всегда имеет корень. *Корень* древовидности — это единственный узел сети, из которого ориентированные дуги могут только исходить. Древовидность не может иметь более одного корня. Следовательно, древовидность из k узлов содержит $k-1$ узел, в каждый из которых заходит только одна дуга, и один узел, из которого дуги могут только исходить.

Дерево, являющееся древовидностью, будем называть *древовидным деревом*, а остов, являющийся древовидностью — *древовидным остовом*.

---

[1] Или, короче, *остовом*. — *Прим. перев.*

[2] Называемые *весами* дуг. — *Прим. перев.*

Формально древовидность с корнем $x_1 \in \mathbf{N}$ определяется следующим образом:

1. Каждый узел, отличный от x_1, является конечным для одной единственной дуги.

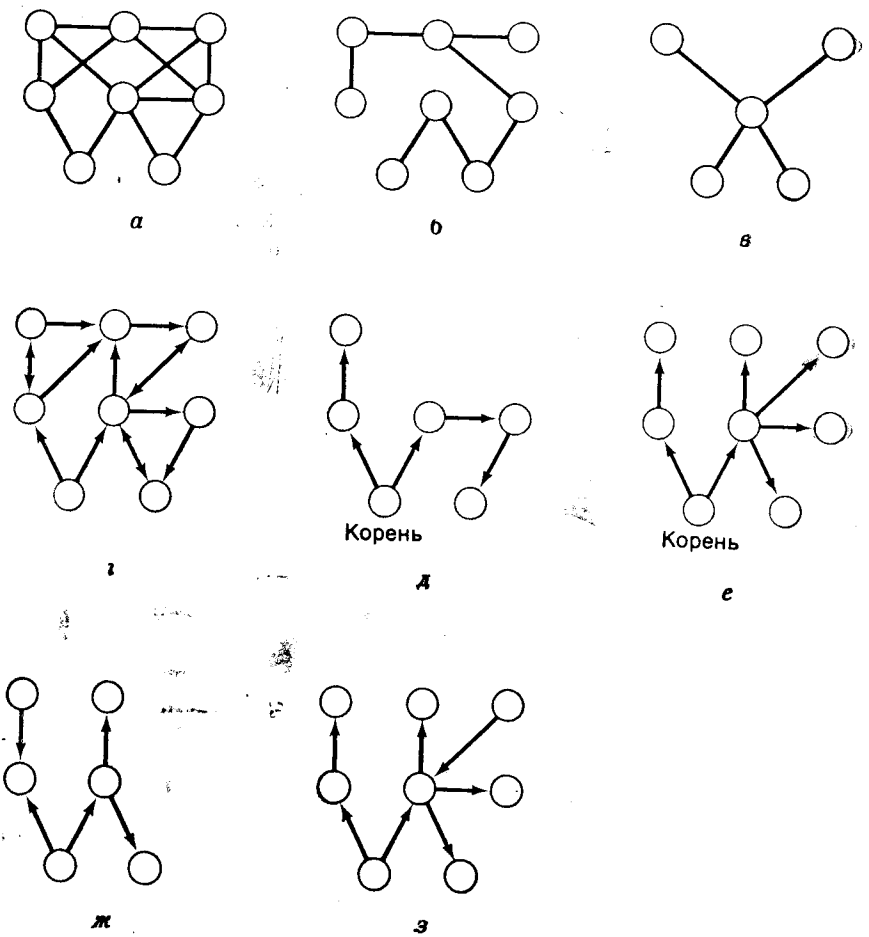


Рис. 1.5. Деревья и древовидности.

а — неориентированная сеть; *б* — остов сети «а»; *в* — дерево сети «а»; *г* — ориентированная сеть; *д* — древовидность сети «г»; *е* — древовидный остов сети «г»; *ж* — дерево сети «г»; *з* — остов сети «г».

2. Узел x_1 не является конечным ни для одной дуги.

3. Сеть не содержит контуров.

Отметим, что для каждого узла, отличного от x_1, существует единственная цепь, соединяющая его с x_1.

Рис. 1.5 иллюстрирует основные определения, приведенные для неориентированных и ориентированных сетей, деревьев и древовидностей. На рис. 1.5, *а—в* изображены неориентированная сеть, остов этой сети и дерево, не являющееся остовным, соответственно. На рис. 1.5, *г—е* изображены ориентированная сеть, древовидность этой сети и ее древовидный остов соответственно. На рис. 1.5, *ж, з* изображены дерево и остов ориентированной сети соответственно.

1.2. МАТРИЧНЫЕ ПРЕДСТАВЛЕНИЯ СЕТЕЙ

Очевидно, что сети, которые графически представлены различным образом, могут соответствовать одним и тем же элементам некоторой системы и выражать одни и те же отношения. Однако установить эквивалентность двух сетей большой размерности иногда бывает чрезвычайно сложно. В теории графов две сети, между дугами и узлами которых может быть установлено взаимно однозначное соответствие, называют *изоморфными.*

Если два графа изоморфны, то любое уравнение, справедливое для одного из них, справедливо и для другого графа. Нередко некоторые общие формы представления сетей позволяют упростить расчет или установить такие свойства исследуемого объекта, благодаря которым отпадает необходимость в выполнении части вычислений или проведении дальнейшего анализа. Помимо того что исследование графической и матричной форм представления сетей помогает при выполнении вычислений, их изучение полезно и с педагогической точки зрения. Количественные характеристики дуг сети, а также взаимосвязь между ее узлами могут быть представлены с помощью матрицы расстояний, или матрицы стоимостей. Кроме того, взаимосвязь между узлами задается с помощью матрицы смежности или матрицы инциденций узлы-дуги.

Для простоты изложения вопросов, связанных с матричным представлением сетей, будем рассматривать сеть $G=(\mathbf{N}, \mathbf{A})$, в которой узлы из множества **N** занумерованы числами 1, 2,...,n, а каждой дуге (i, j) из множества **A** поставлен в соответствие количественный параметр c_{ij}, который обычно называют *обобщенной стоимостью,* или *длиной дуги. Матрица стоимостей* задается массивом $\mathbf{C}=[c_{ij}]$ (для всех $(i, j)\in\mathbf{A}$).

Элементы матрицы **C** определяют возможности рассматриваемой системы или естественные ограничения, накладываемые на нее. Они могут соответствовать расстоянию, транспортным затратам, пропускным способностям водных магистралей, максимально допустимым нагрузкам на линии электропередачи, надежности компонент системы и т. п. Если G — неориентиро-

ванная сеть, то $c_{ij}=c_{ji}$ для всех $(i, j)\in\mathbf{A}$. В этом случае матрица $\mathbf{C}$ является симметричной.

Матрица смежности ориентированной сети задается массивом $\mathbf{X}=[x_{ij}]$, где

$$x_{ij}=\begin{cases}1, & \text{если дуга } (i, j)\in\mathbf{A} \text{ направлена от узла } i\in\mathbf{N} \text{ к узлу } j\in\mathbf{N},\\ 0 & \text{в противном случае.}\end{cases}$$

Если сеть G является неориентированной, то каждую дугу $(i, j)\in\mathbf{A}$ можно заменить двумя противоположно направленными дугами. При этом матрица $\mathbf{X}$ будет симметричной.

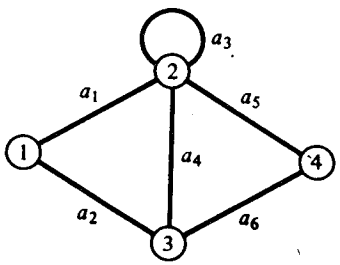


Рис. 1.6. Неориентированная сеть.

Матрица инциденций узлы-дуги ориентированной сети задается массивом $\mathbf{Z}=[z_{ik}]$, где

$$\begin{cases}1, & \text{если узел } i\in\mathbf{N} \text{ является начальным узлом дуги } a_k\in\mathbf{A},\\ -1, & \text{если узел } i\in\mathbf{N} \text{ является конечным узлом дуги } a_k\in\mathbf{A},\\ 0 & \text{в противном случае.}\end{cases}$$

Если сеть G является неориентированной, то величины z_{ik} определяются следующим образом:

$$z_{ik}=\begin{cases}1, & \text{если узел } i\in\mathbf{N} \text{ принадлежит дуге } a_k\in\mathbf{A},\\ 0 & \text{в противном случае.}\end{cases}$$

С помощью матричного представления сетей могут быть получены важные свойства как неориентированных, так и ориентированных сетей. В качестве примера рассмотрим вначале неориентированную сеть $G=(\mathbf{N}, \mathbf{A})$, изображенную на рис. 1.6. Множества $\mathbf{N}$ и $\mathbf{A}$ определяются следующим образом:

$$\mathbf{N}=\{i\}=\{1, 2, 3, 4\},$$
$$\mathbf{A}=\{a_k\}=\{a_1, a_2, a_3, a_4, a_5, a_6\}.$$

Матрица смежности **X** и матрица инциденций **Z** в данном примере имеют вид

$$\mathbf{X} = [x_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array},$$

$$\mathbf{Z} = [z_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccc} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}.$$

Исходя из структуры матрицы смежности **X**, можно утверждать следующее:

1. Если к сети добавляется узел, не связанный с ней дугами, то к матрице следует приписать нулевую строку и нулевой столбец.
2. Матрица является симметричной.
3. Каждый ненулевой элемент главной диагонали соответствует петле.

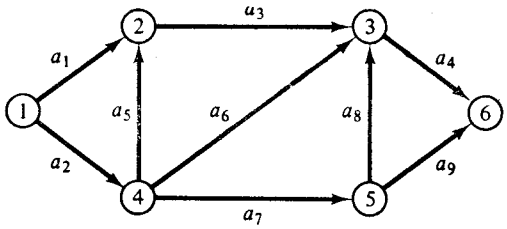


Рис. 1.7. Ориентированная сеть.

В качестве упражнения читателю предлагается вывести аналогичные свойства матрицы инциденций **Z**.

Перейдем к рассмотрению ориентированной сети (рис. 1.7). Множество узлов **N** и множество дуг **A** данной сети определяются следующим образом:

$$\mathbf{N} = \{i\} = \{1, 2, 3, 4, 5, 6\},$$

$$\mathbf{A} = \{a_k\} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}.$$

Матрица смежности **X** и матрица инциденций **Z** в данном примере имеют вид

$$\mathbf{X} = [x_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array},$$

$$\mathbf{Z} = [z_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccccc} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 \end{array} \\ \begin{bmatrix} +1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & +1 & 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & +1 & +1 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & +1 & +1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \end{array}.$$

Исходя из структуры матрицы смежности **X** рассматриваемой ориентированной сети, можно утверждать следующее:

1. Каждый нулевой столбец матрицы соответствует источнику.
2. Каждая нулевая строка матрицы соответствует стоку.
3. Если все элементы главной диагонали нулевые, то сеть не содержит петель, и наоборот, ненулевой элемент главной диагонали соответствует петле.
4. Матрица не является симметричной.

В качестве упражнения читателю предлагается вывести аналогичные свойства матрицы инциденций **Z** ориентированной сети.

Следует отметить, что все сформулированные выше утверждения справедливы и тогда, когда вместо величин x_{ij}, принимающих значения 0 или 1, берутся произвольные величины c_{ij}. Матричные представления сетей нередко оказываются чрезвычайно полезными при определении отношений предпочтения или установления эквивалентности сетевых моделей.

1.3. СОХРАНЕНИЕ ПОТОКА

При изучении характеристик сети часто возникает необходимость в вычислении оптимального значения функции потока, протекающего от источника s к стоку t. Обычно такие вычисления проводятся в задачах, связанных с *однопродуктовым* потоком, поскольку потоки в дугах сети соответствуют потокам некоторого однородного продукта, такого, как электроэнергия, вода, информация, самолеты на воздушных трассах и т. п.

Пусть β_i — множество всех узлов, связанных с узлом i дугами, направленными к i, а α_i — множество всех узлов, связанных с i дугами, направленными в противоположную сторону. Целочисленная функция f_{ij}, определенная на множестве $\mathbf{A}$, называется *потоком* в ориентированной сети $G=(\mathbf{N}, \mathbf{A})$, если

$$f_{ij} \geqslant 0 \quad \text{для всех} \quad (i, j) \in \mathbf{A}, \tag{1.1}$$

$$\sum_{j \in \alpha_i} f_{ij} - \sum_{j \in \beta_i} f_{ji} = 0 \quad \text{для всех} \quad i \in \mathbf{N}, \quad i \neq s, \quad i \neq t, \tag{1.2}$$

$$f_{ij} \leqslant c_{ij} \quad \text{для всех} \quad (i, j) \in \mathbf{A}. \tag{1.3}$$

Согласно приведенному определению, значение f_{ij} можно рассматривать как объем продукта, протекающего по дуге (i, j) от узла i к узлу j, причем данный объем не превосходит пропускной способности c_{ij} дуги (i, j). Если узел j не является ни источником, ни стоком, то величина потока, втекающего в j, должна равняться величине потока, вытекающего из этого узла. Данное положение известно под названием принципа *сохранения потока*. Условие сохранения потока описывается с помощью выражения (1.2). В рассматриваемых в данной книге задачах, связанных с однопродуктовым потоком в сетях, принцип сохранения потока выполняется всегда. Поток в дуге может изменяться, что имеет место, например, для сетей с выигрышами или проигрышами (см. разд. 5.2).

1.4. ТЕОРЕМА О МАКСИМАЛЬНОМ ПОТОКЕ И МИНИМАЛЬНОМ РАЗРЕЗЕ

Пусть V — величина потока, протекающего из источника s в сток t по дугам сети $G=(\mathbf{N}, \mathbf{A})$. Если предположить, что пропускная способность дуг из множества $\mathbf{A}$ конечна, то максимальная величина потока будет ограничена величинами пропускной способности дуг сети.

Максимальный поток определяется с помощью одного из основных понятий теории сетей — *разреза*. Разрез может быть определен как множество дуг, исключение которых из сети отделило бы некоторое множество узлов от остальной части

узлов. На рис. 1.8 изображена сеть, в которой разрез, состоящий из дуг (2, 4) и (3, 4), отделяет узел 4 от группы узлов 1, 2, 3. При решении задачи о максимальном потоке мы будем рассматривать только разрезы, отделяющие источник от стока.

Пропускной способностью, или *величиной*, разреза называется сумма пропускных способностей всех дуг (ориентированных определенным образом[1]) разреза. Пропускная способность разреза, указанного на рис. 1.8, равна $c_{24}+c_{34}$.

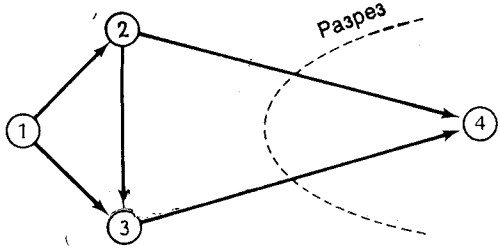


Рис. 1.8. Разрез.

Принцип работы алгоритмов поиска максимального потока в сети достаточно прост. Пусть $\mathbf{X}$ — такое подмножество множества $\mathbf{N}$, что $s \notin \mathbf{X}$, а $t \in \mathbf{X}$. По определению множество всех дуг, инцидентных $\mathbf{X}$ и заходящих в $\mathbf{X}$[2]), образует разрез сети G[3]). Обозначим через $c(\mathbf{A}_X)$ пропускную способность этого разреза. Тогда величина максимального потока из узла s в узел t должна удовлетворять условию

$$V \leqslant c(\mathbf{A}_X). \tag{1.4}$$

Таким образом, если для некоторого потока величиной V и некоторого разреза $\mathbf{A}_V$ выполнено равенство $V=c(\mathbf{A}_V)$, то данный поток является максимальным.

Сформулируем теперь один из наиболее важных результатов теории потоков в сетях — *теорему о максимальном потоке и минимальном разрезе*, доказанную Фордом и Филкерсоном [8]: для любой сети с одним источником и одним стоком величина максимального потока от источника к стоку равна ве-

[1]) Пусть разрез отделяет множество узлов $\mathbf{X}$ от множества узлов $\overline{\mathbf{X}}$, и предположим, что $t \in \mathbf{X}$, $s \in \overline{\mathbf{X}}$. Тогда для ориентированной сети пропускной способностью разреза в направлении от s к t называется сумма пропускных способностей всех его ориентированных дуг, начальные узлы которых принадлежат множеству $\overline{\mathbf{X}}$, а конечные — множеству $\mathbf{X}$. — *Прим. перев.*

[2]) То есть начальные узлы этих дуг принадлежат множеству $\overline{\mathbf{X}}=\mathbf{N}/\mathbf{X}$, а конечные — множеству $\mathbf{X}$. — *Прим. перев.*

[3]) Данное определение разреза отличается от приведенного выше определения. — *Прим. перев.*

личине минимального разреза. Доказательство этой теоремы приводится в гл. 2. Рассмотрим сеть, изображенную на рис. 1.9. Числа, приписанные дугам, соответствуют их пропускным способностям в направлениях, указанных стрелками. Можно по-

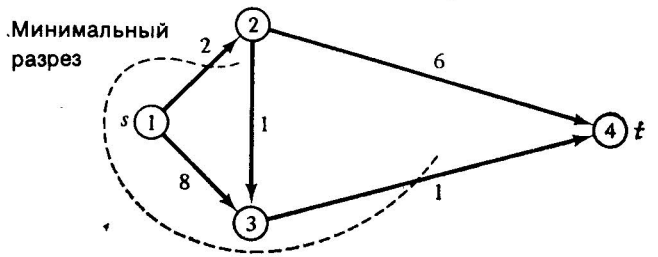


Рис. 1.9. Минимальный разрез.

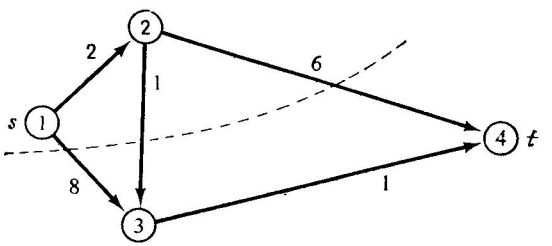


Рис. 1.10. Разрез, не являющийся минимальным.

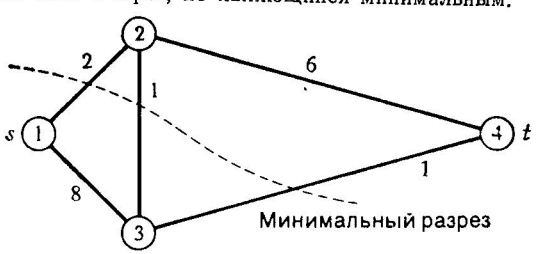


Рис. 1.11. Минимальный разрез неориентированной сети.

казать, что величина максимального потока из s в t равна 3. Минимальный разрез[1]) состоит из дуг (1, 2) и (3, 4) и имеет пропускную способность, равную 3.

Рассмотрим другой разрез, отделяющий узел 1 от узла 4 и состоящий из дуг (1, 3), (2, 3) и (2, 4) (рис. 1.10). Этот разрез имеет пропускную способность, равную 15, и не может быть минимальным.

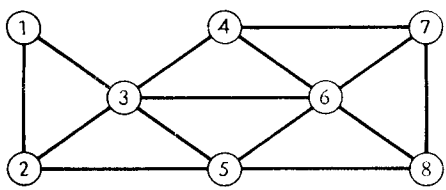
---

[1]) Согласно второму определению разреза. — *Прим. перев.*

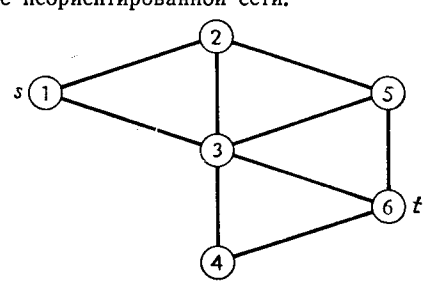
В заключение отметим, что если рассматриваемая сеть была бы неориентированной, то максимальный поток из s в t равнялся бы 4, поскольку минимальный разрез в данном случае состоит из дуг (1, 2), (2, 3) и (3, 4) (рис. 1.11).

УПРАЖНЕНИЯ

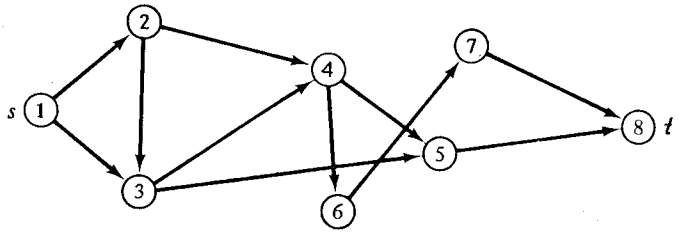
1. Почему методы сетевого анализа являются более предпочтительными по сравнению с другими методами исследования операций? Указать четыре причины.
2. Дать определения следующих понятий: (а) дуга; (б) узел; (в) ориентированная дуга; (г) биориентированная дуга; (д) неориентированная дуга; (е) источник; (ж) сток; (з) главный источник; (и) главный сток.
3. Дать определение цепи и пути. В чем различие между ними?
4. Дать определение цикла и контура. В чем различие между ними?
5. Дать определение циклической и ациклической сетей.
6. Почему для циклических сетей в алгоритме нахождения потока минимальной стоимости могут возникнуть трудности?
7. Дать определение дерева и остовного дерева: (а) словесно; (б) математически; (в) с использованием понятий теории графов.
8. Дать определение древовидности. В чем различие между древовидностью и остовным деревом?
9. Используя изображенную ниже сеть, построить: (а) дерево; (б) остовное дерево.



10. Сформулировать все положения, вытекающие из структуры матрицы инциденций сети, изображенной на рис. 1.7.
11. Сформулировать принцип сохранения потока: (а) словесно; (б) математически.
12. Что такое (а) разрез; (б) величина разреза; (в) максимальный разрез; (г) минимальный разрез?
13. Выписать матрицу инциденций узлы-дуги и матрицу смежности для изображенной ниже неориентированной сети.



14. Выписать матрицу инциденций узлы-дуги и матрицу смежности для изображенной ниже ориентированной сети.



15. В сети из упражнения 13 найти цикл, контур, путь из s в t и цепь из s в t. Аналогичную задачу решить для сети из упражнения 14.

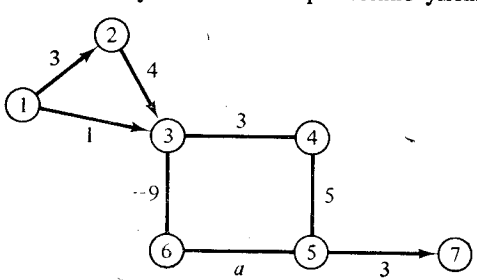
16. Рассмотреть сеть $G=(\mathbf{N}, \mathbf{A})$, где $\mathbf{N}=\{1, 2, ..., 9\}$, $\mathbf{A}=\{a_1, a_2, ..., a_{12}\}$, а матрица инциденций узлы-дуги имеет вид

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
1	1	1	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	1	0	0	0	0	0	0	0
3	1	0	1	1	0	0	0	0	0	0	0	0
4	0	0	0	1	0	1	1	0	0	0	1	0
5	0	0	0	0	1	0	1	0	0	0	1	0
6	0	0	0	0	0	0	0	0	0	1	0	1
7	0	0	0	0	0	1	0	0	1	1	0	0
8	0	0	1	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	1	1	0	0	1

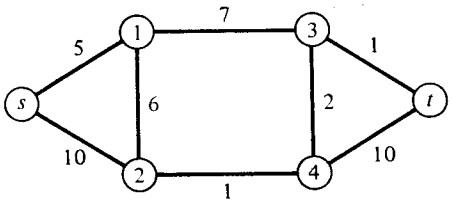
(а) начертить сеть; (б) выписать матрицу смежности.

17. Рассмотреть ориентированную сеть $G=(\mathbf{N}, \mathbf{A})$, где $\mathbf{N}=\{1, 2, 3, 4, 5\}$, $\mathbf{A}=\{(1, 2), (1, 3), (3, 1), (2, 3), (2, 4), (2, 5), (3, 5), (4, 5)\}$: (а) начертить сеть; (б) выписать матрицу смежности; (в) выписать матрицу инциденций узлы-дуги.

18. Для изображенной ниже сети найти минимальное значение параметра a, при котором узел 4 принадлежит кратчайшему пути из узла 1 в узел 7, не содержащему циклов. При каком условии, наложенном на параметр a, длина кратчайшего пути может неограниченно уменьшаться?



19. Чему равна величина максимального потока из узла s в узел t для изображенной ниже сети? Найти минимальный разрез. Указанные числа соответствуют пропускным способностям дуг.



20. Рассмотреть сеть из упражнения 19. Написать уравнение сохранения потока для узлов s, 3 и t.

ЛИТЕРАТУРА

1. Bazarra M., Jarvis J. J., Linear Programming and Network Flows, Wiley, Inc., New York, 1978.
2. Bradley G. H., A Survey of Deterministic Networks, *AIIE Transactions,* **7**, 222—234 (1975).
3. Busacker R. G., Saaty T., Finite Graphs and Networks, McGraw-Hill Co., New York, 1965. [Имеется перевод: Басакер Р., Саати Т. Л. Конечные графы и сети. — М.: Наука, 1974.]
4. Charnes A., Cooper W. W., Management Models and Industrial Applications of Linear Programming, Vols. 1—2, Wiley, Inc., New York, 1961.
5. Dantzig G. L., Linear Programming and Extensions, Princeton University Press, Princeton, N. J., 1963. [Имеется перевод: Данциг Д. Л. Линейное программирование и его применения и обобщения. — М.: Прогресс, 1966.]
6. Elmaghraby S., Some Network Models in Management Science, Springer-Verlag, Inc., New York, 1970.
7. Euler L., The Konigsberg Bridges, *Scientific American,* **189**, 66—70 (1953).
8. Ford L. R., Fulkerson D. R., Flows in Networks, Princeton University Press, Princeton, N. J., 1962. [Имеется перевод: Форд Л. Р., Фалкерсон Д. Потоки в сетях. — М.: Мир, 1966.]
9. Frank H., Frisch I. T., Communication, Transmission, and Transportation Networks, Addison-Wesley Publ. Co., Inc., Reading Mass., 1971. [Имеется перевод: Фрэнк Г., Фриш И. Сети, связь и потоки. — М.: Связь, 1978.]
10. Fulkerson D. R., Flow Networks and Combinatorial Operations Research, *American Mathematical Monthly,* **73**, 115—138 (1966).
11. Hitchcock F. L., The Distribution of a Product from Several Sources to Numerous Localities, *Journal of Mathematics and Physics,* **20**, 224—230 (1941).
12. Hu T. C., Integer Programming and Network Flows, Addison-Wesley Publ. Co., Inc., Reading Mass., 1969. [Имеется перевод: Ху Т. Целочисленное программирование и потоки в сетях. — М.: Мир, 1974.]
13. Iri M., Network Flow, Transportation and Scheduling, Academic Press, Inc., New York, 1969.
14. Jensen P. A., Barnes W., Network Flow Programming, Wiley, Inc., New York, 1980.
15. Koopmans T. C., Optimum Utilization of the Transportation System, Proceedings of the International Statistical Conference, Washington, D. C., 1947.

16. Magnanti T. L., Golden B. L., Transportation Planning: Network Models and Their Implementation, Working Paper 77-008, University of Maryland, General Research Board, Faculty Research Award, 1977.
17. Minieka E., Optimization Algorithms for Networks and Graphs, Marcel Dekker, Inc., New York, 1978. [Имеется перевод: Майника Э. Алгоритмы оптимизации на сетях и графах. — М.: Мир, 1981.]
18. Phillips D. T., Ravindran A., Solberg J. J., Operations Research: Principles and Practice, Wiley, Inc., New York, 1977.
19. Potts R. B., Oliver R. M., Flows in Transportation Networks, Academic Press, Inc., New York, 1972.
20. Pritsker A. A. B., Happ W. W., GERT: Part I — Fundamentals, *Journal of Industrial Engineering,* 17 (5), 267 (1966).
21. Whitehouse G. W., Systems Analysis and Design Using Network Techniques, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1973.

Глава 2

ДЕТЕРМИНИРОВАННЫЕ ПОТОКИ В СЕТЯХ

Кролик Питер скорчил рожу Скунсу Джимми:
— Я не люблю, когда меня поучают.
— Я не поучаю; я просто говорю,
что Вам следует знать без всяких объяснений, —
ответил Скунс Джимми.

Т. Баргес. Приключения кролика Питера

В настоящей главе рассматриваются некоторые детерминированные потоковые модели, часто используемые при постановке и решении важных экономических и инженерных задач. Для каждой рассматриваемой задачи описан эффективный алгоритм ее решения. В числе рассмотренных будут следующие:

1. Алгоритм Дейкстры решения задачи о кратчайшей цепи [11].
2. Алгоритм поиска многополюсной кратчайшей цепи [31].
3. Алгоритм поиска кратчайшего пути для решения класса задач с фиксированными платежами [41].
4. Алгоритм двойного поиска для решения задачи о K кратчайших путях [47].
5. Алгоритм построения кратчайшего остова [31].
6. Метод ветвей и границ для решения задачи коммивояжера [40].
7. Алгоритм нахождения максимального потока [15].
8. Алгоритм нахождения многополюсного максимального потока [31].
9. Алгоритм нахождения многополюсной цепи с максимальной пропускной способностью [32].
10. Алгоритм нахождения числа узлов и дуг, исключение которых из сети приводит к ее разрыву [18].

Кроме того, для более глубокого понимания принципов работы указанных алгоритмов мы остановимся на нескольких специальных.

В главе большое внимание уделяется рассмотрению конкретных примеров. Для каждой изучаемой проблемы дается общее описание, математическая постановка и затем — описание алгоритма решения. Для иллюстрации основных понятий и методов вычислений, используемых в каждом из описанных алгоритмов, рассматриваются и решаются вручную один или несколько примеров. Затем формулируется некоторая задача, решение которой может быть получено с помощью специальной программы, написанной на языке ФОРТРАН АНСИ. Каждая программа содержит раздел с документацией и инструкциями для пользователя. В ряде случаев на модельных примерах удается получить дополнительные результаты, касающиеся изучаемой проблемы. Тексты ФОРТРАН-программ приводятся в приложении.

2.1. ПРИЛОЖЕНИЯ ПОТОКОВЫХ МОДЕЛЕЙ[1]

В данном разделе будут рассмотрены задачи, при решении которых используются потоковые алгоритмы. Это познакомит читателя с сетевым моделированием и сообщит ему основные сведения, необходимые для эффективного использования сетевых алгоритмов.

2.1.1. ЗАМЕНА ОБОРУДОВАНИЯ

В инженерном проектировании и прикладном анализе часто встречаются задачи, которые могут быть сформулированы в виде *задач о кратчайшей цепи.* Задача о кратчайшей цепи заключается в следующем. Заданы множества узлов и дуг. Каждой дуге (i, j) поставлена в соответствие величина c_{ij}, равная стоимости единицы потока по этой дуге. Требуется найти цепь из источника s в сток t, минимизирующую стоимость единицы потока, протекающего из s в t. В качестве примера рассмотрим задачу периодической замены оборудования. Для удобства будем предполагать, что вышедшее из строя оборудование предпочтительнее заменить на новое, чем отремонтировать его. Предполагается также, что после трех лет работы в поточной линии оборудование устаревает и в дальнейшем не может быть использовано. Ликвидационная стоимость при различных сроках службы оборудования, а также эксплуатационные расходы и расходы на техническое обслуживание и текущий ремонт в каждый промежуток времени считаются известными. Обозначим узлом начало каждого планируемого периода. Тогда заданный период полезного срока службы оборудования может быть обозначен дугой. Данная информация позволяет вычислить величину c_{ij} в виде суммы затрат на приобретение нового оборудования в начале периода i и торговых издержек в конце периода j—1 (или в начале периода j). Эта величина рассматривается как «длина» дуги (i, j).

Множество всех планов замены оборудования за n—1 год может быть представлено совокупностью цепей сети, изображенной на рис. 2.1 (n=4). Каждый узел сети соответствует началу планируемого периода; узел 4 соответствует концу планируемого периода. Каждая ориентированная цепь из узла 1 в узел 4 представляет собой план замены оборудования на трехлетний период. Цепь (1, 3), (3, 4) соответствует закупке оборудования в начале первого периода, его содержанию в течение двух лет и его продаже в конце второго периода. Затем в

[1] Часть материала данного раздела была заимствована у Беннингтона Г. Е. и перепечатана с разрешения American Institute of Industrial Engineers [4].

начале третьего периода закупается новое оборудование, которое продается в конце этого же периода. Для данного плана замены оборудования общие затраты составляют $c_{13}+c_{34}$. Кратчайшая цепь из узла 1 в узел 4 в данной сети будет соответствовать плану с минимальными затратами на рассмотренные три периода.

Сделаем несколько замечаний. Нами была рассмотрена временна́я, или динамическая, модель, в которой временны́е

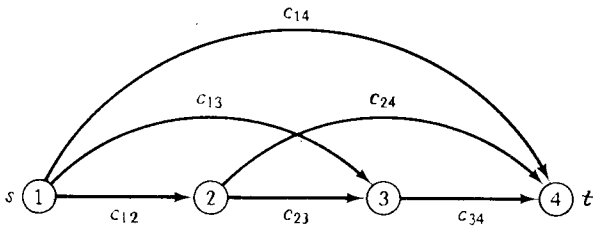


Рис. 2.1. Сеть планов замены оборудования.

периоды представлены узлами сети. Поскольку в начале первого периода оборудование следует закупить, а в конце периода планирования его следует продать, то все цепи в сети должны вести из узла 1 в узел 4. Модели подобного типа называются *моделями с конечным планируемым периодом*. Задачи о кратчайшей цепи рассматриваются в разд. 2.3 и 2.4.

2.1.2. ПЛАНИРОВАНИЕ РАБОТ ПО ОСУЩЕСТВЛЕНИЮ ПРОЕКТА

Значительное место в теории сетей занимают задачи, связанные с планированием и составлением расписания выполнения работ по осуществлению больших проектов, таких, как запуск космических кораблей, проведение научных исследований и опытных разработок, организация сложных конструкторских разработок. В рассматриваемой модели проект представляется множеством работ и заданными между ними отношениями предшествования. Время d_i выполнения каждой работы известно. Кроме того, прежде чем работа может начаться, все предшествующие ей работы должны быть завершены. Предположим, например, что заданы пять работ и что работа 1 предшествует работе 3, работы 1, 2 предшествуют работе 4, а работы 1, 2, 3 и 4 предшествуют работе 5. Данная совокупность работ может быть представлена сетью, изображенной на рис. 2.2. Для обозначения начала и завершения проекта вводятся два фиктивных узла. Остальные узлы соответствуют началу выполнения рассматриваемых работ. Если работа j мо-

жет быть начата непосредственно после окончания работы i, то в сеть включается дуга (i, j) длиной d_i. Поскольку работа 1 предшествует работе 3, которая в свою очередь предшествует работе 5, то дуга (1, 5) в сеть не включается. Длина максимальной цепи из начального узла сети в некоторый задан-

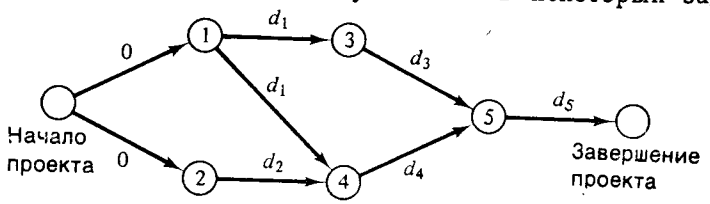


Рис. 2.2. Сетевая модель узел — работа.

ный узел i соответствует самому раннему сроку начала выполнения работы i. Поэтому длина максимальной цепи из начального узла сети в конечный узел соответствует минимальному времени осуществления проекта.

Если сеть содержит контуры положительной длины, то задача нахождения максимальной цепи, не содержащей конту-

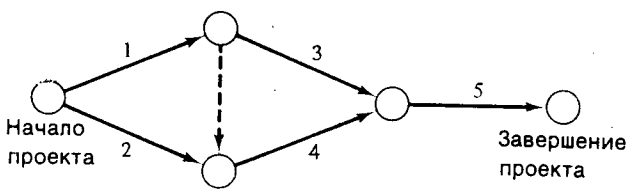


Рис. 2.3. Сетевая модель дуга — работа.

ров, является весьма сложной. Рассматриваемая нами сеть проекта не содержит контуров, и цепь максимальной длины определяется несложно.

Рассмотренная модель называется *моделью узел — работа*, поскольку началу выполнения каждой работы соответствует узел. В другой сетевой модели, называемой *моделью дуга — работа*, каждая работа представляется дугой. Соответствующая сеть рассматриваемого нами проекта изображена на рис. 2.3. Числа, приписанные дугам, обозначают номера соответствующих работ.

Пунктирной стрелкой обозначена фиктивная работа нулевой продолжительности, введенная для того, чтобы работа 1 предшествовала работе 4. Каждый узел в данной сети соответствует некоторому событию, например завершению закладки фундамента при строительстве здания. В общем случае пред-

полагается, что событие является точно определенным во времени. Как и раньше, длина максимальной цепи из начального узла сети проекта в конечный узел равна минимальному времени осуществления проекта.

Метод планирования работ по осуществлению проекта, использующий сетевое представление совокупности работ, называется *методом критического пути* (МКП). Впервые он был разработан Дюпоном и Ремингтоном Рандом для управления работой химических заводов. Цепь максимальной длины называется критическим путем потому, что задержка в выполнении любой работы, принадлежащей этому пути, приведет к увеличению времени осуществления проекта. Аналогичный метод был независимо разработан для построения стохастической модели управления научными исследованиями и опытными разработками, которая использовалась при создании ракетной системы «Поларис». Этот метод называется *методом оценки и пересмотра планов* (ПЕРТ). Он позволяет получить оценку среднего времени выполнения проекта и дисперсию этой оценки. Процедуры поиска решения в системах МКП и ПЕРТ описаны в гл. 4.

2.1.3. СОСТАВЛЕНИЕ РАСПИСАНИЯ ДВИЖЕНИЯ ГРУЗОВОГО СУДНА

Предположим, что имеется грузовое судно и требуется составить оптимальный график прибытия в порты. Рассмотрим сеть, в которой узлы соответствуют портам, а стоимость c_{ij} дуги (i, j) равна затратам на транспортировку груза из порта i в порт j за время t_{ij}, которое включает в себя время, необходимое для погрузки судна в порту i, перевозки груза из порта i в порт j и разгрузки судна в порту j. Если прибыль рассматривать как отрицательные затраты, то для ориентированного цикла с минимальным значением $(\Sigma c_{ij})/(\Sigma t_{ij})$ величина средней прибыли в единицу времени будет максимальной.

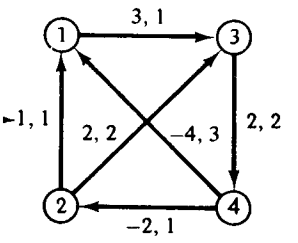


Рис. 2.4. Сеть, описывающая графики прибытия судна в порты.

Оптимальное расписание движения судна задается контуром (1, 3), (3, 4), (4, 1) (рис. 2.4). При этом величина $(\Sigma c_{ij})/(\Sigma t_{ij})$ равна $(3+2-4)/(1+2+3) = {}^1/_6$. Числа, приписанные дугам, указывают затраты c_{ij} и времена t_{ij}. Если все t_{ij} неотрицательные, то поставленная задача решается с помощью методов, аналогичных тем, которые используются при поиске кратчайшей цепи.

2.1.4. ЗАДАЧИ О МАКСИМАЛЬНОМ ПОТОКЕ И ПОТОКЕ МИНИМАЛЬНОЙ СТОИМОСТИ

Предыдущие примеры были сведены к задаче поиска цепи или контура, на которых достигается оптимальное значение специальным образом подобранной целевой функции. В этих задачах дуги можно рассматривать как устройства, в результате работы которых от источника к стоку через узлы сети протекает 1 единица потока. Параметром дуги, или расстоянием, является обобщенная стоимость использования дуги. Обычно такие модели называют *сетями с расстояниями.*

В более общей интерпретации дуга представляется как звено в механизме, предназначенном для транспортирования многих единиц потока. В этом случае параметр дуги соответствует максимальной величине потока, который может протекать по дуге в единицу времени. Этот параметр называется *пропускной способностью дуги,* а сеть — сетью *с ограниченной пропускной способностью.*

Пусть $G=(\mathbf{N}, \mathbf{A})$ — ориентированная сеть, где $\mathbf{N}=\{1, 2, \ldots, n\}$ — множество узлов, $\mathbf{A}$ — множество дуг, и пусть U_{ij} — пропускная способность дуги (i, j). Для удобства будем предполагать, что узел 1 является источником, а узел n — стоком.

Согласно определению, данному в разд. 1.3, целочисленная функция f_{ij}, определенная на множестве $\mathbf{A}$, называется потоком в сети G, если она удовлетворяет ограничениям (1.1)—(1.3). Значение $\sum_j f_{sj}=\sum_j f_{jt}=V$ называется величиной потока. Задача о максимальном потоке заключается в определении максимально допустимой величины V и может быть сформулирована следующим образом:

$$\text{максимизировать } V \tag{2.1}$$

при условии, что

$$\sum_j f_{ij}-\sum_j f_{ji}=\begin{cases} V, & i=1, \\ 0, & i\neq 1, \quad i\neq n, \\ -V, & i=n, \end{cases} \tag{2.2}$$

$$0\leqslant f_{ij}\leqslant U_{ij}, \quad (i, j)\in\mathbf{A}. \tag{2.3}$$

В уравнениях (2.2) суммирование производится по всем узлам, для которых функция f_{ij} определена. При целочисленных дуговых параметрах U_{ij} рассматриваемая модель является абсолютно унимодулярной и поэтому оптимальное решение f_{ij} также является целочисленным. Хотя данная задача может быть решена с помощью методов линейного программирования, более эффективным решением оказывается метод расстановки пометок, состоящий в построении возрастающей последова-

тельности потоков, последним членом которой является максимальный поток. Метод расстановки пометок описан в разд. 2.14. Определение понятия унимодулярности дается в разд. 2.2.

Для того чтобы создать более общую теоретическую основу, необходимую для понимания рассматриваемых ниже прикладных задач, введем несколько дополнительных понятий. Для каждой дуги (i, j) определим линейный коэффициент стоимости c_{ij}, а помимо верхней границы U_{ij} введем нижнюю границу L_{ij}. Для каждого узла i будет определено предложение $b_i \geqslant 0$. При этом будем предполагать, что отрицательное значение b_i соответствует спросу в узле i. В описанной модели при $b_i > 0$ узел i можно рассматривать как источник, при $b_i = 0$ — как промежуточный узел, а при $b_i < 0$ — как сток. После проведенного обобщения задача поиска потока минимальной стоимости может быть сведена к поиску таких целочисленных дуговых потоков f_{ij}, которые являются решением следующей задачи:

$$\text{минимизировать} \sum_i \sum_j c_{ij} f_{ij} \tag{2.4}$$

при условии, что

$$\sum_j f_{ij} - \sum_j f_{ji} = b_i, \quad i \in \mathbf{N}, \tag{2.5}$$

$$L_{ij} \leqslant f_{ij} \leqslant U_{ij}, \quad (i, j) \in \mathbf{A}. \tag{2.6}$$

Хотя данная задача может быть решена с помощью симплексного метода, более эффективным решением является процедура, основанная на использовании сетевой структуры рассматриваемой модели и условии дополняющей нежесткости, формулируемом в линейном программировании. Данная процедура известна под названием метода дефекта [15] (см. гл. 3).

2.1.5. ТРАНСПОРТНАЯ ЗАДАЧА

Транспортная задача является одной из первых потоковых задач. Впервые она была решена в 1941 г. Хитчкоком [29] и с тех пор применяется при решении многих задач перевозки и распределения. Несомненно, данная модель является наиболее широко используемой моделью. Постановка транспортной задачи, а также методы ее решения содержатся в разд. 2.10.

Наиболее наглядной данная модель является при описании способов перевозки груза с заводов на склады. Предположим, что имеются m заводов и n складов. Предложение i-го завода равно s_i, $i = 1, 2, \ldots, m$, а спрос на j-м складе равен d_j, $j = 1, 2, \ldots, n$. Задача минимизации общих затрат на перевозку

груза эквивалентна определению величин x_{ij}, являющихся решением следующей задачи:

$$\text{минимизировать} \sum_i \sum_j c_{ij}x_{ij} \tag{2.7}$$

при условии, что

$$\sum_i x_{ij} = s_i, \quad i = 1,2,..., m, \tag{2.8}$$

$$\sum_i x_{ij} = d_j, \quad j = 1,2,..., n, \tag{2.9}$$

$$x_{ij} \geqslant 0, \quad i = 1,2 ..., m, \quad j = 1,2,..., n. \tag{2.10}$$

Для того чтобы система уравнений (2.8), (2.9) была совместной, предполагается, что $\sum_i s_i = \sum_j d_j$. Данная модель может быть описана с помощью сети, если предположить, что узлами являются заводы и склады, а дугами — имеющиеся дороги для перевозки груза. Тогда дорога, соединяющая i-й завод с j-м складом, представляется ориентированной дугой (i, j). На рис. 2.5 изображена сеть, построенная для $m=2$, $n=3$. Следует отметить, что транспортная задача является частным случаем задачи поиска потока минимальной стоимости, сформулированной в разд. 2.1.4, и сводится к последней при $b_j = -d_j$, $b_i = s_i$, $L_{ij}=0$ и $U_{ij}=\infty$.

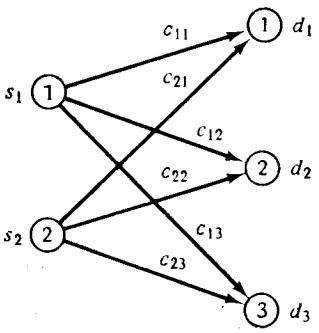


Рис. 2.5. Сеть в транспортной задаче.

Частным случаем транспортной задачи (при $m=n$, $s_i=1$, $d_j=1$) является задача о *назначениях*, которая называется так потому, что типичная ее постановка связана с оптимальным назначением рабочих на рабочие места. Задача о назначениях и методы ее решения рассматриваются в разд. 2.12.

2.1.6. ЗАДАЧА О ПОСТАВЩИКЕ

Владелец кафе должен иметь d_j салфеток на каждый из n последовательных дней. Он может покупать новые салфетки стоимостью α центов каждая или отдавать грязные салфетки в стирку, причем в прачечной имеются два вида обслуживания — обычное и срочное. Для простоты будем предполагать, что салфетка, отданная в обычную стирку, бывает готова через

2 дня, а плата при этом равна b центам; для салфетки, отданной в срочную стирку, соответствующие величины равны 1 дню и c центам, причем $c>b$. Задача заключается в выборе такого плана покупки и стирки салфеток, при котором спрос удовлетворяется при минимальных затратах. Данная задача может быть сформулирована в виде сетевой задачи. Для этого введем следующие обозначения: p_j — число салфеток, купленных для пользования в j-й день; s_j — число салфеток, отдан-

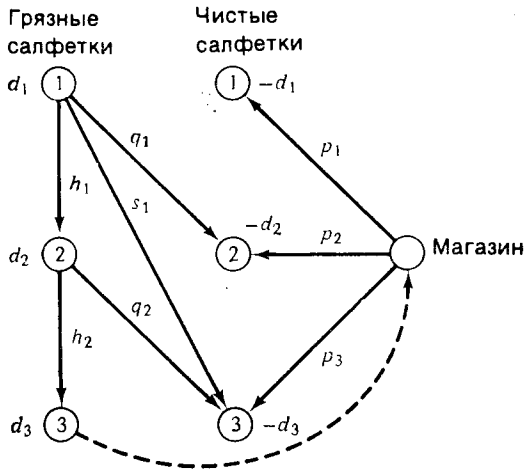


Рис. 2.6. Сеть в задаче о поставщике.

ных в обычную стирку в j-й день; q_j — число салфеток, отданных в срочную стирку в j-й день; h_j — число грязных салфеток, отложенных до следующего дня.

На рис. 2.6 изображена соответствующая сеть при $n=3$. Каждой дуге приписана одна из введенных выше переменных, а каждому узлу — соответствующее предложение. Пунктирная дуга используется для того, чтобы уравновесить спрос и предложение. Единственной особенностью данной задачи является то, что требуемое число чистых салфеток d_j равно числу салфеток, использованных к концу дня. После некоторых преобразований решение данной задачи может быть сведено к решению транспортной задачи.

Хотя эта формулировка задачи выглядит несколько фривольной, впервые ее решение было использовано при планировании работы по ремонту оборудования, производимому на военных заводах. В данном случае каждая величина спроса d_j известна либо соответствует предполагаемому количеству единиц оборудования специального типа, которое должно быть от-

ремонтировано к заданному сроку. Обычному и срочному обслуживанию соответствует выполнение ремонта в местной мастерской и отправка оборудования в центральную мастерскую.

2.1.7. КАЛЕНДАРНОЕ ПЛАНИРОВАНИЕ ТРУДОВЫХ РЕСУРСОВ

Рассматриваемая ниже модель может быть использована для выработки политики в области занятости, позволяющей надлежащим образом сбалансировать затраты, связанные с наймом и увольнением рабочих, и затраты на содержание людей, не имеющих работу в течение непродолжительного периода времени. Предполагается, что спрос на рабочую силу является детерминированным, но не постоянным на протяжении заданного периода планирования.

Предположим, что для каждого периода j известен минимальный спрос D_j на рабочую силу. Пусть x_{ij} — число людей, нанятых на работу в начале i-го периода и уволенных в конце j—1-го периода, а c_{ij} — затраты на одного рабочего из числа x_{ij}. Тогда число людей, работающих в j-й период, равно $\sum_{r\leqslant j}\sum_{t>j} x_{rt}$, или $\sum_{r=1}^{j}\sum_{t=j+1}^{n} x_{rt}$. Если избыток рабочей силы на период j обозначить через s_j, то задача календарного планирования трудовых ресурсов на n—1 периодов заключается в нахождении неотрицательных целых чисел x_{ij} и s_j, минимизирующих функционал

$$\sum_{r=1}^{n-1}\sum_{t=r+1}^{n} c_{rt}x_{rt} \tag{2.11}$$

при условии, что

$$\sum_{r=1}^{j}\sum_{t=j+1}^{n} x_{rt} - s_j = D_j, \quad j=1,\ldots,n-1, \tag{2.12}$$

$$s_j \geqslant 0, \quad x_{ij} \geqslant 0. \tag{2.13), (2.14}$$

Данная постановка не допускает достаточно простую сетевую интерпретацию; это становится возможным лишь после некоторых преобразований. Предлагаемая ниже процедура описана для случая $n=4$. Ограничения (2.12) задаются уравнениями

$$x_{12}+x_{13}+x_{14} \qquad -s_1 \qquad = D_1, \tag{2.15}$$

$$x_{13}+x_{14}+x_{23}+x_{24} \qquad -s_2 \qquad = D_2, \tag{2.16}$$

$$x_{14} \qquad x_{24}+x_{34} \qquad -s_3 = D_3. \tag{2.17}$$

Вычитая (2.15) из (2.16), (2.16) из (2.17) и добавляя избыточное условие, полученное умножением обеих частей уравнения (2.17) на —1, мы получаем следующую систему уравнений:

$$\begin{aligned}
x_{12}+x_{13}+x_{14} \qquad\qquad -s_1 \qquad\quad &= D_1,\\
-x_{12} \qquad +x_{23}+x_{24} \qquad +s_1-s_2 \qquad &= D_2-D_1,\\
-x_{13} \qquad -x_{23} \qquad +x_{34} \qquad +s_2-s_3 &= D_3-D_2,\\
-x_{14} \qquad -x_{24}-x_{34} \qquad\quad +s_3 &= -D_3.
\end{aligned}$$

Нетрудно видеть, что каждая переменная входит в полученные уравнения дважды — один раз с коэффициентом +1 и один раз с коэффициентом —1. Такая структура коэффициентов имеет

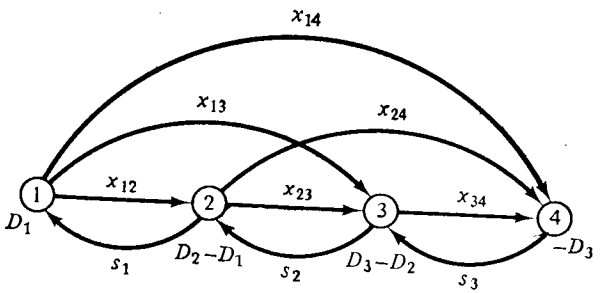


Рис. 2.7. Сеть в задаче календарного планирования трудовых ресурсов.

особый смысл, и в разд. 2.2 мы подробно остановимся на вопросах, связанных с ее использованием. На рис. 2.7 изображена сеть для рассматриваемого примера. Переменные здесь приписаны дугам, а величины предложения — узлам. Данная сеть во многом схожа с сетью, построенной нами при рассмотрении задачи о замене оборудования, за исключением дуг s_j.

2.1.8. ЗАДАЧА СОСТАВЛЕНИЯ РАСПИСАНИЯ ДВИЖЕНИЯ ТРАНСПОРТНЫХ СУДОВ

На практике часто возникают задачи планирования работы транспортных средств, решение которых необходимо для усовершенствования процесса перевозки некоторого продукта из пунктов снабжения к пунктам потребления. Частный случай такой задачи — составление расписания движения грузовых судов — связан с определением минимального числа судов, необходимого для выполнения плана перевозок к заданному сроку, и составлением оптимального маршрута их движения.

Предположим, что вы являетесь владельцем судоходной компании, специализирующейся на перевозке скоропортящихся

Таблица 2.1. Исходные данные в задаче составления расписания движения грузовых судов

Рейс	Пункт отправления	Пункт назначения	Время прибытия	Прибыль
1	Порт *A*	Порт *C*	3	10
2	Порт *A*	Порт *C*	8	10
3	Порт *B*	Порт *D*	3	3
4	Порт *B*	Порт *C*	6	4

продуктов. Нескольким вашим клиентам потребовалось осуществить перевозку товара. Поскольку груз скоропортящийся, клиенты указывают точную дату перевозки. Если ваша компания не сможет перевезти груз в назначенный срок, то это будет сделано конкурирующей компанией.

Ниже описаны четыре рейса (в каждом из которых суда загружаются полностью). В табл. 2.1 содержится информация о каждом рейсе. Первую строку таблицы можно интерпретировать следующим образом: в порту *A* имеется груз, который следует доставить в порт *C* на третий день (считая день выгрузки). Прибыль, получаемая от каждой перевозки, определяется доходами и эксплуатационными расходами, непосредственно связанными с данной перевозкой. Помимо эксплуатационных расходов компания платит постоянную сумму (5 единиц) за то, чтобы ввести судно в эксплуатацию. Следует также учитывать выплату компанией постоянной суммы, включающей накладные расходы и расходы, связанные с наймом новых членов экипажей судов.

Время транспортировки (включая время на погрузку и разгрузку) определяется следующим образом:

Пункт отправления \ Пункт назначения	*C*	*D*
A	3	2
B	2	3

Время, требуемое для возвращения судна (когда оно загружено только балластом), определяется следующим образом:

Пункт назначения \ Пункт отправления	*A*	*B*
C	2	1
D	1	2

При рассмотрении данной задачи возникает несколько вопросов. А именно, какие перевозки нам следует выполнять и сколько судов для этого потребуется? Сколько судов потребуется для того, чтобы выполнить все перевозки в заданные сроки? Для того чтобы ответить на эти вопросы, необходимо по-

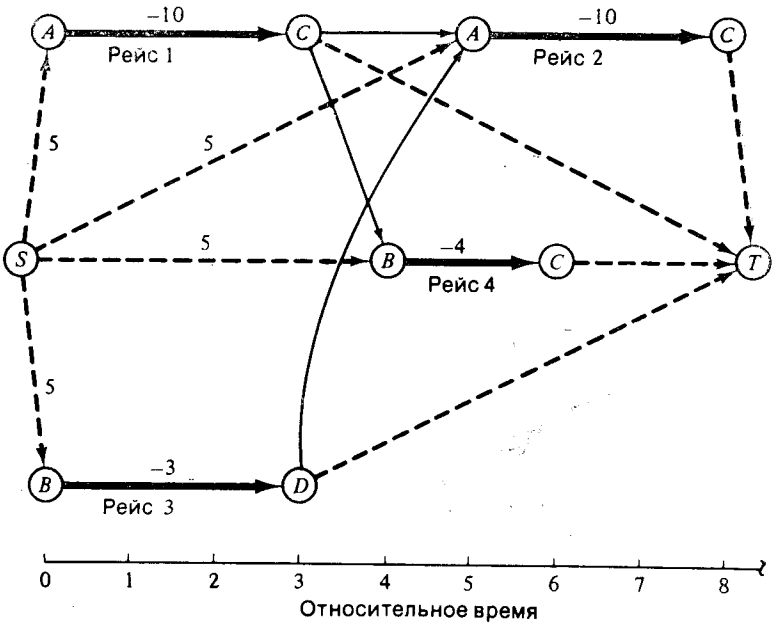


Рис. 2.8. Задача составления расписания движения грузового судна.

Жирными стрелками изображены маршруты загруженного судна, следующего в один из портов (узел). Сплошными стрелками изображены маршруты разгруженного судна, следующего в один из портов для погрузки. Пунктирные стрелки вводятся для обозначения затрат, связанных с вводом судов в эксплуатацию (узел S) и окончанием их работы (узел T).

строить сеть, описывающую всевозможные расписания движения судов. Каждый узел данной сети соответствует некоторому порту и некоторому моменту времени. При данных обозначениях для каждого порта и момента прихода в него загруженного судна в сети имеется соответствующий им узел. То же справедливо и для каждого порта и момента начала погрузки в нем судна.

Для удобства построения сети в нижней части рис. 2.8 изображена временна́я шкала. Жирные дуги соответствуют четырем рейсам, рассмотренным выше. Пропускная способность жирных дуг равна 1. Все остальные дуги имеют бесконечную пропускную способность. Числа, приписанные дугам, представляют собой ненулевые стоимости.

При завершении порожнего рейса судно может быть загружено и отправлено в какой-нибудь другой порт. Поскольку время, необходимое для того, чтобы судно возвратилось из порта C в порт B, равно одному дню, в сеть включена дуга, соответствующая выходу судна из порта C в момент времени 3 и заходу его в порт B в момент времени 4. Остальные дуги, соединяющие конец одной жирной дуги, соответствующей транспортированию товара, с началом другой жирной дуги, изображают другие маршруты разгруженного судна, следующего в порт для того, чтобы загрузиться и совершить очередную перевозку.

Для завершения построения сети вводятся два дополнительных узла. Источник S помещается в начало временной шкалы для обозначения начальной точки маршрута судов, а сток T — в конец шкалы для обозначения конечного пункта их движения. Пунктирные дуги, исходящие из S, соответствуют первому использованию судна. Дуги, заходящие в T, соответствуют окончанию эксплуатации судна.

Каждая цепь из источника в сток соответствует допустимому расписанию движения одного судна. Ответы на поставленные выше вопросы можно получить, решая задачу нахождения потока минимальной стоимости. Для того чтобы были учтены расходы, связанные с первым использованием судна, каждой дуге, исходящей из S, приписана стоимость, равная 5 единицам. Стоимость каждой дуги, соответствующей транспортировке товара, равна взятой со знаком минус величине прибыли, получаемой от данной перевозки. Все остальные дуги имеют нулевую стоимость. Пропускная способность каждой дуги, соответствующей транспортировке товара, равна 1, поскольку существует только один рейс данного типа. Все остальные дуги имеют бесконечную пропускную способность.

Если количество используемых судов равно M, то поток, величина которого также равна M, а стоимость минимальная, будет определять оптимальное расписание движения судов и перевозки, которые следует производить. Если количество судов выбирается, исходя из максимальной прибыли, то задача может быть решена путем определения зависимости прибыли от числа судов. Для нашего примера такая зависимость сведена в табл. 2.2.

Расписание движения судов может быть составлено с помощью сети или непосредственно из исходных данных. В рассматриваемом примере второе судно, выполняющее рейсы 3 и 2, начинает грузиться в порту B в нулевой момент времени для выполнения рейса 3. В порту D его разгрузка заканчивается в момент времени 3. Затем судно следует разгруженным в порт A и в момент времени 5 загружается там для выполнения рей-

Таблица 2.2. Решения, зависящие от параметра, для задачи составления расписания движения грузовых судов

Количество судов	Расписание перевозок	Затраты	Прибыль
1	Судно 1: 1, 2	−15	15
2	Судно 1: 1, 4	−9	
	Судно 2: 3, 2	−8	17

са 2. В порту C его разгрузка заканчивается в момент времени 8.

Методика решения рассмотренной здесь задачи используется также для определения оптимального маршрута движения автолавки при заданном множестве всех путей, ведущих к пунктам сбыта продукции. Затраты связаны с выполнением порожних рейсов, когда автолавка возвращается незагруженной в начальную точку маршрута.

2.1.9. МОДЕЛЬ ПРОИЗВОДСТВЕННОГО ПЛАНИРОВАНИЯ (СМИТА И ДЖОНСОНА)

Предпринимателю необходимо удовлетворять спрос на продукцию в течение T периодов [49]. Обозначим величины ожидаемого спроса в рассматриваемых периодах через $d_1, d_2,..., d_T$, где d_j — ожидаемый спрос в j-м периоде. Предположим, что предприниматель может получать продукцию из L различных пунктов производства, или пунктов снабжения. Пусть B_{ij} — максимальный объем продукции, которая может быть получена из i-го пункта снабжения в j-й период, а x_{ij} ($i=1, 2,..., L$, $j=1,...,T$) — объем продукции, действительно получаемой из i-го пункта снабжения в j-й период. Таким образом, x_{ij} — искомые значения переменных. Пусть c_{ij} — затраты, необходимые для получения единицы продукции из i-го пункта снабжения в j-й период, а c_s — издержки хранения единицы продукции в течение одного периода. Предполагается, что вместимость товарных складов достаточно большая и поэтому можно не накладывать ограничений на объем хранимой в них продукции. И наконец, через I_0 обозначим запас продукции, имеющийся в начальный момент периода планирования.

Для математического описания данной модели будет полезно иметь выражение объема запасов продукции на конец каждого периода t. Обозначая эту величину через I_t, имеем $I_t=I_0+\Sigma^t_{j=1}\Sigma^L_{i=1}x_{ij}-\Sigma^t_{j=1}d_j$, $t=1, 2, ..., T$. Здесь следует отметить, что если величины x_{ij} выбраны таким образом, что $I_t\geqslant 0$ при всех t, то спрос будет удовлетворяться в каждый период.

Теперь задачу можно сформулировать следующим образом: найти неотрицательные величины x_{ij}, при которых функция

$$Z=\sum_{j=1}^{T}\sum_{i=1}^{L}c_{ij}x_{ij}+c_s\sum_{t=1}^{T}\left(I_0+\sum_{j=1}^{t}\sum_{i=1}^{L}-\sum_{j=1}^{t}d_j\right) \qquad (2.18)$$

принимает минимальное значение. Первое слагаемое в целевой функции (2.18) представляет собой затраты, необходимые на получение продукции из пунктов снабжения, второе слагаемое соответствует издержкам $c_s\Sigma^T{}_{t=1}I_t$ хранения продукции за весь период планирования.

Кроме того, должны выполняться ограничения двух типов. Ограничения первого типа гарантируют, что поставки продукции не превысят предложения. Таким образом, мы имеем LT ограничений вида

$$x_{ij}\leqslant B_{ij} \text{ для всех } i=1,2,\ldots,L \text{ и } j=1,2,\ldots,T. \qquad (2.19)$$

Второй тип ограничений гарантирует удовлетворение спроса в каждом периоде. Мы имеем T ограничений вида

$$I_0+\sum_{j=1}^{t}\sum_{i=1}^{L}x_{ij}-\sum_{j=1}^{t}d_j\geqslant 0 \text{ для всех } t=1,2,\ldots,T. \qquad (2.20)$$

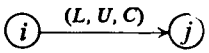
Остановимся на следующем примере. Будем рассматривать три периода и предположим, что в каждом периоде продукция может производиться как в рабочее, так и в сверхурочное время. Соответствующие значения приведены в табл. 2.3.

Таблица 2.3. Пример задачи производственного планирования

Период	Объем выпускаемой продукции (в единицах)		Стоимость единицы продукции (в долл.)		Ожидаемый спрос
	В рабочее время	В сверхурочное время	В рабочее время	В сверхурочное время	
1	100	20	14	18	60
2	100	10	17	22	80
3	60	20	17	22	140

Для того чтобы свести данную задачу к потоковой задаче, следует определить компоненты сети. Будем предполагать, что узлы сети соответствуют источникам снабжения продукции, периодам, в течение которых возникает необходимость в ее производстве, и спросу на продукцию в эти периоды. Можно указать семь возможных источников снабжения продукции. Она может быть: а) взята из начальных запасов; б) произведена в каждом из трех производственных периодов в рабочее время; в) произведена в каждом из трех производственных периодов в сверхурочное время. Дуги сети будут иметь четыре

характеристики: 1) направление допустимого потока; 2) минимальную величину требуемого потока по дуге; 3) максимальную величину допустимого потока по дуге; 4) стоимость единицы потока по дуге. Дуга, которая соединит пару узлов i и j, графически будет обозначаться следующим образом:



где L — минимальная величина допустимого потока, U — максимально допустимый поток, C — стоимость единицы потока. Задача сетевой оптимизации заключается в нахождении потока минимальной стоимости, гарантирующего удовлетворение спроса в периодах 1, 2 и 3 ограниченным предложением источников снабжения.

На рис. 2.9 изображена сеть для данной задачи. Рассмотрим дугу (I, P_1). Параметр (0, 15, 0) дуги указывает на то, что в первом периоде продукцию из начального запаса можно либо не использовать совсем $(L=0)$, либо использовать не более 15 ее единиц $(U=15)$, причем стоимость продукции нулевая $(C=0)$. Аналогично параметр дуги (I, P_2) указывает на то, что во втором периоде начальный запас объемом 15 единиц также может быть не использован, однако стоимость единицы продукции, если она используется, равна 1 долл. $(C=1)$. Аналогичный смысл имеют параметры всех дуг, ведущих из источников I, R_1, O_1, R_2, O_2,, R_3, O_3 к узлам, представляющим периоды (P_1, P_2, P_3). И наконец, благодаря введению дуг (P_1, D_1), (P_2, D_2) и (P_3, D_3) предложение удовлетворяет спрос. Например, минимальная величина требуемого потока по дуге (P_2, D_2) равна 80 единицам $(L=80)$, максимально допустимый поток также равен 80 единицам $(U=80)$, а стоимость единицы потока равна нулю $(C=0)$. Следовательно, по дуге (P_2, D_2) в узел D_2 поступает 80 единиц потока. Дуги (P_1, D_1) и (P_3, D_3) накладывают аналогичные требования на объем продукции, выпускаемой в первом и третьем периодах соответственно.

Данная задача может быть решена как задача нахождения потока минимальной стоимости с использованием алгоритма решения транспортной задачи, описанного в разд. 2.10, или алгоритма дефекта, описанного в гл. 3. Полное описание задач подобного класса содержится в гл. 3.

2.1.10. ЗАКЛЮЧЕНИЕ

Мы остановились на рассмотренных выше моделях для того, чтобы показать, насколько разнообразными являются задачи, решение которых может быть получено с помощью сете-

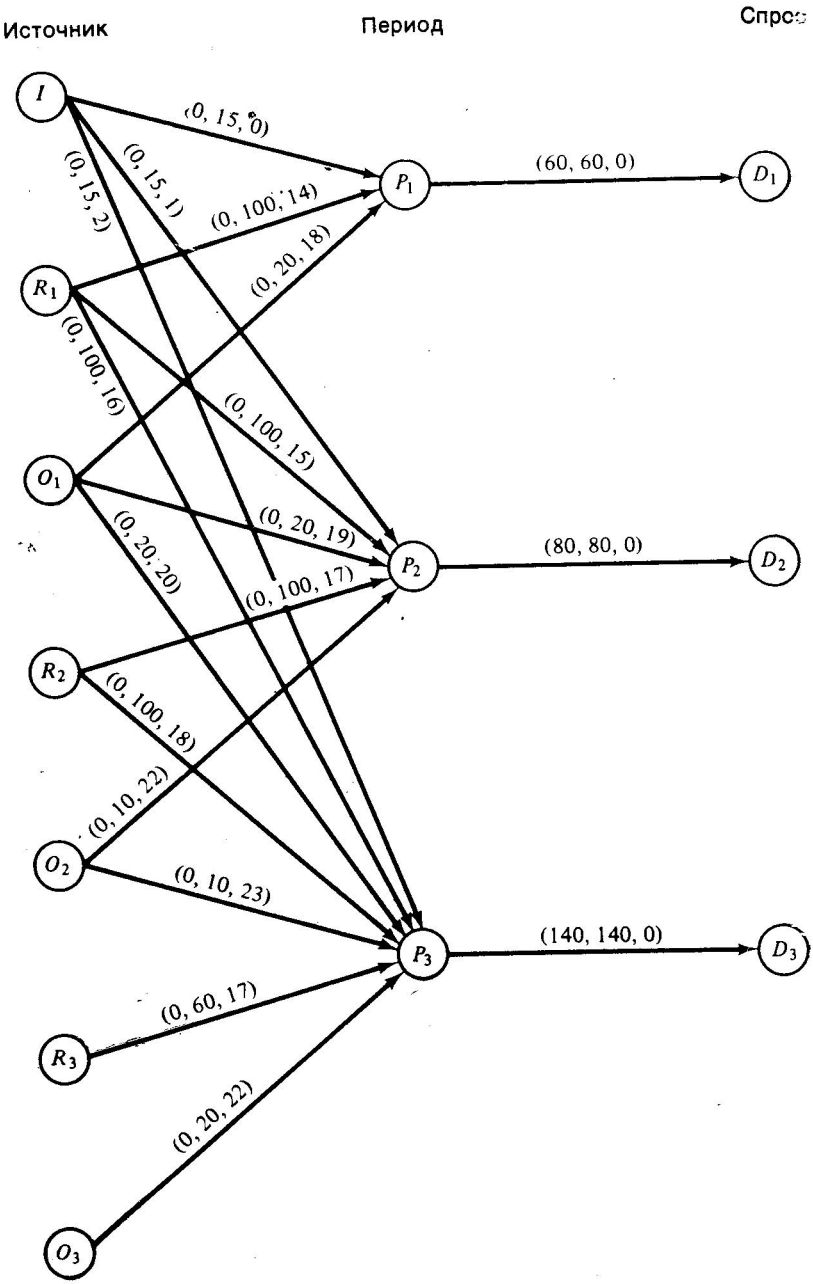


Рис. 2.9. Модель производственного планирования

вого моделирования. Основное внимание было уделено потоковым моделям. Такие модели могут быть как детерминированными, так и стохастическими, имеющими детерминированный аналог. Используя статические модели, мы определили понятие работы. Динамические модели возникали главным образом в тех случаях, когда узлы соответствовали временным периодам или комбинациям момент времени-место расположения. Модели подобного типа являются весьма общими и могут использоваться даже в том случае, когда размерность задачи не позволяет воспользоваться традиционными методами линейного программирования.

Особенность задач нахождения кратчайшей цепи и однопродуктовых потоковых задач в том, что их решение сводится к легко осуществляемому поиску некоторых цепей сети. Несомненно, это способствовало тому, что данные модели получили широкое распространение.

Рассмотрение поставленных выше задач имело целью познакомить читателя с сетевыми моделями. Эти задачи значительно проще, чем те, которые реально возникают на практике. Однако их изучение полезно по той причине, что они являются несложными для понимания и решаются просто. Кроме того, эти модели позволяют наглядно изобразить решение каждой задачи. В последующих разделах мы остановимся на методах решения этих и некоторых других задач. Будут также описаны соответствующие вычислительные алгоритмы.

2.2. ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ И ПОТОКИ В СЕТЯХ

Основной результат данного раздела заключается в следующем: показано, что большинство потоковых задач, рассматриваемых в последующих разделах книги, могут быть сформулированы как задачи линейного программирования (ЛП). Возникает естественный вопрос: почему мы не ограничиваемся сведением этих задач к задачам ЛП? Во-первых, специфика структуры потоковых моделей, описывающих реально существующие системы, позволяет находить более эффективные алгоритмы поиска решения, чем симплексный метод. Во-вторых, эти специальные алгоритмы часто позволяют решать задачи ЛП большой размерности за время, меньшее среднего времени решения подобных задач. В-третьих, именно тот факт, что мы можем описать математические зависимости взаимосвязями узлов и дуг сети, нередко помогает нам при постановке и решении практических задач.

Особое внимание следует уделить изучению специфики потоковых задач, сформулированных в виде задач ЛП, поскольку знание особенностей структуры сети играет главную роль в

повышении эффективности алгоритмов. В частности, целесообразно заранее определить, в каких случаях при применении методов ЛП будут получены целочисленные оптимальные решения. При этом может быть использовано несколько важных теорем, приводимых в данном разделе. Если заранее известно, что решение целочисленное, то для его поиска можно воспользоваться одной из следующих процедур: 1) специальным алгоритмом ЛП, позволяющим находить целочисленное оптимальное решение; 2) итеративными процедурами, непосредственно использующими сетевую структуру модели. Такие итеративные процедуры основаны на использовании аддитивной арифметики, а целочисленные решения, получаемые при их применении, соответствуют решениям задачи ЛП.

Излагаемый в данном разделе материал основан на результатах, полученных Гарфинкелем и Немхаузером [19]. Рассмотрим сеть с α источниками, β стоками и φ промежуточными узлами. Предположим, что однопродуктовый поток должен протекать из источников в стоки через промежуточные узлы. Каждая дуга сети характеризуется верхней границей U_{ij} потока f_{ij} через нее и стоимостью c_{ij} единицы потока. Сеть может быть представлена в виде ориентированного графа $G=(\mathbf{N}, \mathbf{A})$, где $\mathbf{N}=\mathbf{N}_\alpha \cup \mathbf{N}_\beta \cup \mathbf{N}_\varphi$, а $\mathbf{N}_\alpha$, $\mathbf{N}_\beta$ и $\mathbf{N}_\varphi$ — множества источников, стоков и промежуточных узлов соответственно. Задача нахождения потока минимальной стоимости, поставленная в разд. 2.1.4, может быть переформулирована математически следующим образом:

$$\text{минимизировать} \sum_i \sum_j c_{ij} f_{ij} \tag{2.21}$$

при условии, что

$$\sum_j f_{ij} - \sum_j f_{ji} \leqslant a_i, \quad i \in \mathbf{N}_\alpha, \tag{2.22}$$

$$\sum_j f_{ij} - \sum_j f_{ji} = 0, \quad i \in \mathbf{N}_\varphi, \tag{2.23}$$

$$\sum_j f_{ji} - \sum_j f_{ij} \geqslant b_i, \quad i \in \mathbf{N}_\beta, \tag{2.24}$$

$$0 \leqslant f_{ij} \leqslant U_{ij}, \quad (i, j) \in \mathbf{A}. \tag{2.25}$$

Данная задача состоит в минимизации стоимости потока в сети (см. (2.21)). Из неравенства (2.22) следует, что в нашем распоряжении находится не менее $\sum_i a_i$ единиц продукта, а согласно неравенствам (2.24), в стоки должно быть доставлено не менее $\sum_i b_i$ единиц. Соотношения (2.25) указывают на то, что

потоки по дугам ограничены. Равенства (2.23) представляют собой условия сохранения потока для каждого промежуточного узла (см. гл. 1). Частным случаем общей задачи, описываемой выражениями (2.21)—(2.25), является каждая из следующих четырех важных задач, постановка которых приводилась в разд. 2.1: 1) транспортная задача (разд. 2.1.5), 2) задача о назначениях (разд. 2.1.5), 3) задача о максимальном потоке (разд. 2.1.4) и 4) задача о кратчайшей цепи (разд. 2.1.1).

Случай 1. Транспортная задача. В случае когда поток течет непосредственно из источников в стоки, сеть не содержит промежуточных узлов. Если требуется, чтобы фиксированное предложение удовлетворяло заданный спрос при минимальных затратах, то возникает классическая транспортная задача с ограничениями, которая формулируется следующим образом:

$$\text{минимизировать} \sum_i \sum_j c_{ij} f_{ij} \tag{2.26}$$

при условии, что

$$\sum_j f_{ij} \leqslant a_i, \quad i \in \mathbf{N}_\alpha, \tag{2.27}$$

$$\sum_j f_{ji} \geqslant b_i, \quad i \in \mathbf{N}_\beta, \tag{2.28}$$

$$0 \leqslant f_{ij} \leqslant U_{ij}, \quad (i, j) \in \mathbf{A}. \tag{2.29}$$

Если на пропускные способности одной или нескольких дуг, соединяющих источники с источниками или стоки со стоками, не накладываются ограничения сверху, то возникает *задача о перевозках.*

Случай 2. Задача о назначениях. В случае когда поток по каждой дуге не ограничен сверху (т. е. $U_{ij}=\infty$ для $i, j \in \mathbf{N}$) и, кроме того, $a_i=1$ для $i \in \mathbf{N}$ и $b_i=1$ для $i \in \mathbf{N}_\beta$, то возникает задача о назначениях:

$$\text{минимизировать} \sum_i \sum_j c_{ij} f_{ij} \tag{2.30}$$

при условии, что

$$\sum_j f_{ij} \leqslant 1, \quad i \in \mathbf{N}_\alpha, \tag{2.31}$$

$$\sum_j f_{ji} \geqslant 1, \quad i \in \mathbf{N}_\beta, \tag{2.32}$$

$$f_{ij} \geqslant 0, \quad (i, j) \in \mathbf{A}. \tag{2.33}$$

Отметим, что если число стоков равно числу источников, то ограничения (2.31), (2.32) записываются в виде равенств. Не-

обходимое условие существования допустимого решения заключается в том, что число источников должно быть не меньше числа стоков.

Случай 3. Задача о максимальном потоке. Рассмотрим случай, когда имеется только один источник ($i=1$) и один сток ($i=n$), предложение не ограничено ($a_1=\infty$), а спрос нулевой ($b_n=0$). Если пропускная способность каждой дуги ограничена, а стоимость единицы потока по дуге равна $c_{in}=-1$ для $i\neq n$ и нулю в противном случае, то задача, рассмотренная в случае 1, переходит в задачу максимизации суммарной величины потока, втекающего в сток n:

$$\text{максимизировать} \sum_{i\neq n} f_{in} \tag{2.34}$$

при условии, что

$$\sum_i f_{ij} - \sum_l f_{ji} = 0, \quad i \neq 1, \quad i \neq n, \tag{2.35}$$

$$0 \leqslant f_{ij} \leqslant U_{ij}, \quad (i, j) \in \mathbf{A}. \tag{2.36}$$

Случай 4. Задача о кратчайшей цепи. Предположим, что параметр дуги c_{ij} соответствует стоимости единицы потока по дуге (i, j) или времени, необходимому для передачи единицы потока от узла i к узлу j. Как отмечалось в разд. 2.1.1, в данном случае возникает задача определения минимальной стоимости (времени) передачи единицы потока от источника к заданному стоку. Пусть $\alpha=1$, $\beta=1$, $\mathbf{N}_\alpha=\{1\}$ и $\mathbf{N}_\beta=\{n\}$. Предположим далее, что $a_1=1$ и $b_n=1$. Поскольку предложение составляет одну единицу, а ограничения на поток по дуге записываются в виде $0\leqslant f_{ij}\leqslant 1$, то они никогда не нарушаются и поэтому могут вообще не рассматриваться. Наконец, предположим, что сеть не содержит контуров отрицательного веса. Если все дуги сети ориентированные, то задача состоит в нахождении кратчайшей цепи из узла 1 (источника) в узел n (стока). Математически это может быть записано следующим образом:

$$\text{минимизировать} \sum_i \sum_j c_{ij} f_{ij} \tag{2.37}$$

при условии, что

$$\sum_{j\in \mathrm{N}} f_{1j} \leqslant 1, \tag{2.38}$$

$$\sum_{j\in \mathrm{N}} f_{jn} \geqslant 1, \tag{2.39}$$

$$\sum_j f_{ij} - \sum_j f_{ji} = 0, \quad i \neq 1, \quad i \neq n, \tag{2.40}$$

$$f_{ij} \geqslant 0, \quad (i, j) \in \mathbf{A}. \tag{2.41}$$

Следует отметить, что неравенства (2.38), (2.39) при подстановке в них значений f_{ij}, образующих оптимальное решение, переходят в равенства.

Как было показано выше, постановка широкого круга важных потоковых задач может быть записана в виде (2.21)—(2.25). Алгоритмы решения этих и некоторых других задач будут описаны ниже в данной главе. Сейчас наша цель заключается в том, чтобы показать, что если величины a_i, b_i и U_{ij} — целые, то любое базисное допустимое решение общей задачи (2.21)—(2.25) является целочисленным. Для доказательства этого факта введем два важных *определения.*

Определение 1. Квадратная целочисленная матрица, определитель которого равен 0, $+1$ или -1, называется *унимодулярной.*

Определение 2. Целочисленная матрица называется *абсолютно унимодулярной,* если все ее квадратные подматрицы унимодулярные.

Следующая система линейных неравенств (2.42) и (2.43) определяет область допустимых значений элементов неотрицательного вектора **F** (будем предполагать, что матрица **D** и вектор **B** являются целочисленными):

$$\mathbf{DF} \leqslant \mathbf{B}, \qquad \mathbf{F} \geqslant 0. \qquad (2.42), (2.43)$$

Данные определения используются в следующих двух теоремах.

Теорема 1. Если матрица **D** абсолютно унимодулярная, то *любое* базисное решение системы $\mathbf{DF}=\mathbf{B}$, $\mathbf{F}\geqslant 0$ является целочисленным.

Теорема 2. Рассмотрим произвольную целочисленную матрицу **D**. Следующие два условия являются эквивалентными:

А. Матрица **D** является абсолютно унимодулярной.

Б. Все базисные решения задачи линейного программирования с ограничениями $\mathbf{DF}\leqslant\mathbf{B}$, $\mathbf{F}\geqslant 0$ являются целочисленными.

Эти теоремы занимают центральное место в наших исследованиях, поскольку из них следует что если в задаче ЛП ограничения записываются в форме $\mathbf{DF}\leqslant\mathbf{B}$, $\mathbf{F}\geqslant 0$, где **D** — абсолютно унимодулярная матрица, а **B** — целочисленный вектор, то любое базисное решение (угловая точка) этой задачи является целочисленным. Особый интерес представляет оптимальное решение, которое также должно быть целочисленным. Мы покажем, что если величины a_i, b_i и U_{ij} — целые, то оптимальное решение потоковой задачи (2.21)—(2.25) является целочисленным.

Для простоты изложения рассмотрим сеть с одним источником ($\alpha=1$, $\mathbf{N}_\alpha=\{1\}$), одним стоком ($\beta=1$, $\mathbf{N}=\{6\}$) и шестью промежуточными узлами ($\varphi=6$, $\mathbf{N}_\varphi=\{2, 3, 4, 5, 7, 8\}$).

Графическое изображение данной сети дается на рис. 2.10. Задача (2.21)—(2.25) в рассматриваемом случае может быть сформулирована следующим образом:

$$\text{минимизировать} \sum_i \sum_j c_{ij} f_{ij}$$

при условии, что

$$\begin{gathered}
f_{12}+f_{14}+f_{18} \leqslant a_1, \\
f_{23}+f_{25}+f_{27}-f_{12}=0, \\
f_{43}+f_{45}+f_{47}-f_{14}=0, \\
f_{83}+f_{85}+f_{87}-f_{18}=0, \\
f_{36}-f_{23}-f_{43}-f_{83}=0, \\
f_{56}-f_{25}-f_{45}-f_{85}=0, \\
f_{76}-f_{27}-f_{47}-f_{87}=0, \\
f_{36}+f_{56}+f_{76} \geqslant b_6, \\
0 \leqslant f_{ij} \leqslant U_{ij}, \quad (i, j) \in \mathbf{A}.
\end{gathered}$$

Для того чтобы доказать, что любое допустимое базисное решение данной задачи является целочисленным, достаточно по-

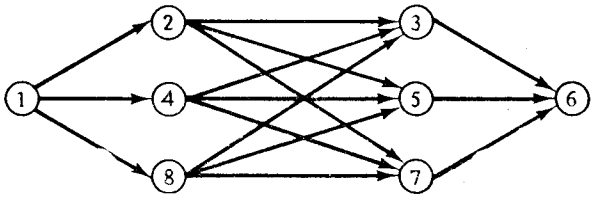


Рис. 2.10. Поток в сети.

казать, что матрица коэффициентов в общей задаче является абсолютно унимодулярной. Матрица ограничений может быть представлена в виде

$$\mathbf{D}=[\mathbf{Z}/\mathbf{I}], \tag{2.44}$$

где матрица **Z** состоит из коэффициентов при f_{ij} в равенствах, выражающих сохранение потока, и в неравенствах, соответствующих ограничениям на предложение и спрос, а **I** — единичная матрица, образованная из коэффициентов при f_{ij} в неравенствах, соответствующих ограничениям на потоки в дугах. Наша цель состоит в том, чтобы доказать, что матрица **D** является абсолютно унимодулярной.

Подматрицы **Z** и **I** матрицы **D**, определенной равенством (2.44), имеют следующий вид:

$$\mathbf{Z} = \begin{array}{c} \\ 1 \\ 2 \\ 4 \\ 8 \\ 3 \\ 5 \\ 7 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} f_{12} & f_{14} & f_{18} & f_{23} & f_{25} & f_{27} & f_{43} & f_{45} & f_{47} & f_{83} & f_{85} & f_{87} & f_{36} & f_{56} & f_{76} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix} \end{array},$$

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Отметим, что матрица **Z** совпадает с матрицей инциденций узлы-дуги (см. гл. 1). Для того чтобы показать, что матрица **D** является абсолютно унимодулярной, воспользуемся следующими теоремами.

Теорема 3 [28]. Целочисленная матрица M является абсолютно унимодулярной, если выполнены следующие условия:

а) каждый элемент матрицы равен 0, $+1$ или -1;

б) каждый столбец матрицы содержит не более двух ненулевых элементов;

в) строки матрицы **M** можно разбить на два непересекающихся множества $\mathbf{E}_1$ и $\mathbf{E}_2$ таким образом, что

1. Если некоторый столбец содержит два ненулевых элемента одного знака, то один из них входит в $\mathbf{E}_1$, а другой — в $\mathbf{E}_2$.

2. Если некоторый столбец содержит два ненулевых элемента с противоположными знаками, то оба они входят либо в $\mathbf{E}_1$, либо в $\mathbf{E}_2$.

Теорема 4 [19]. Если **M** — абсолютно унимодулярная матрица, а **I** — единичная матрица, то матрица $\mathbf{P}=[\mathbf{M}|\mathbf{I}]$, составленная из **M** и **I**, также является абсолютно унимодулярной.

Утверждение. Матрица, транспонированная по отношению к абсолютно унимодулярной матрице, также является абсолютно унимодулярной.

Из соотношений (2.21)—(2.25) или соответствующих соотношений рассматриваемого примера видно, что каждый столбец матрицы **Z** содержит только два ненулевых элемента: $+1$ и -1. Таким образом, условия (а) и (б) теоремы 3 выполнены. Условие (в) выполняется, когда $\mathbf{E}_1$ содержит все строки матрицы **Z**, а $\mathbf{E}_2$ — пустое множество. Следовательно, матрица **Z** является абсолютно унимодулярной. Согласно теореме 4[1], матрица [**Z**/**I**] также является абсолютно унимодулярной. Тогда из теорем 1 и 2 следует существование целочисленного оптимального решения.

Сформулированные выше математические утверждения несколько заслонили собой основной результат, полученный нами в данном разделе. Он заключается в том, что если задача ЛП может быть сформулирована в виде (2.21)—(2.25), а элементы матрицы ограничений удовлетворяют условиям абсолютной унимодулярности, то *оптимальное* решение исходной задачи является *целочисленным*. На практике задачи ЛП редко решаются обычным симплексным методом. Значительно чаще используются модифицированный симплексный метод или эффективные специальные потоковые алгоритмы решения сетевой задачи, эквивалентной исходной задаче ЛП. Это говорит о том, что большое число задач ЛП может решаться как потоковые задачи. В тех случаях, когда это удается сделать, как правило, становятся очевидными простота сетевых методов и эффективность соответствующих алгоритмов. Поэтому большое значение имеет изучение вопросов, связанных со сведением некоторых практических задач к сетевым задачам и разработкой вычислительных алгоритмов поиска решений этих задач. Данной теме посвящены последующие разделы книги.

2.3. ЗАДАЧА О КРАТЧАЙШЕЙ ЦЕПИ. АЛГОРИТМ ДЕЙКСТРЫ

Исходя из применения на практике, одной из наиболее важных потоковых задач является задача нахождения цепи из источника в сток, минимизирующей стоимость (время) прохождения потока заданной величины по данной цепи. Предположим, что каждой дуге (i, j) ориентированной сети поставлено в соответствие некоторое число c_{ij}, называемое обобщенной стоимостью дуги. Фиктивным, или «бесплатным», дугам приписывается стоимость $c_{ij}=0$, а каждой паре узлов (i, j), для которых не существует дуги, соединяющей их, приписыва-

[1] и утверждению, сформулированному после теоремы 4. — *Прим. перев.*

ется стоимость $c_{ij}=\infty$. Задача, решаемая в данном разделе, состоит в нахождении в заданной сети такой цепи из источника s в сток t, для которой стоимость прохождения единицы потока по этой цепи минимальна. Математически эта задача может быть записана как следующая задача ЛП:

$$\text{минимизировать} \sum_i \sum_j c_{ij} f_{ij} \tag{2.45}$$

при условии, что

$$\sum_j f_{sj} - \sum_j f_{js} = 1, \tag{2.46}$$

$$\sum_j f_{ij} - \sum_j f_{ji} = 0, \quad i \neq s, \quad i \neq t, \tag{2.47}$$

$$\sum_j f_{tj} - \sum_j f_{jt} = -1, \tag{2.48}$$

$$f_{ij} \geqslant 0. \tag{2.49}$$

Согласно первому равенству, единица потока вытекает из источника. Второе равенство гарантирует сохранение данной единицы потока при протекании по сети. Согласно третьему равенству, единица потока втекает в сток. В качестве кратчайшей цепи может быть взята последовательность смежных дуг (i, j), для которых $f_{ij}=1$. Для решения сформулированной нами задачи линейного программирования мы воспользуемся специальным методом, известным под названием *алгоритма Дейкстры* [11]. Данный алгоритм был разработан с целью использовать специфику рассматриваемой модели.

Предположим, что стоимость каждой дуги, или расстояние между двумя узлами, выражается неотрицательным числом c_{ij}. Алгоритм основан на приписывании узлам либо *временных,* либо *постоянных пометок.* Первоначально каждому узлу, исключая источник, приписывается временная пометка, соответствующая длине кратчайшей дуги, ведущей из источника в данный узел. Источнику приписывается постоянная пометка, значение которой равно нулю. Каждому узлу, в который нельзя попасть непосредственно из источника, приписывается временная пометка ∞, а всем остальным узлам — временные пометки c_{sj}, $j \neq s$. Если определено, что узел принадлежит кратчайшей цепи, его пометка становится постоянной. Алгоритм основан на следующем простом факте: если известна кратчайшая цепь из узла s в узел j и узел k принадлежит этой цепи, то кратчайшая цепь из s в k является частью первоначальной цепи, оканчивающейся в узле k. Алгоритм начинает работать при

$j=s$. Затем величина j последовательно увеличивается на единицу, и при $j=t$ алгоритм завершает работу.

2.3.1. ИТЕРАТИВНАЯ ПРОЦЕДУРА

Для заданного узла j через δ_j будем обозначать оценку длины кратчайшей цепи из источника s в узел j. Если эта оценка не может быть улучшена, то соответствующее значение мы назовем «постоянной пометкой» и будем обозначать ее символом $\boxed{\delta_j}$. В противном случае назовем его «временной пометкой». Вначале процедуры постоянная пометка приписывается только источнику. Каждая другая пометка является временной и ее величина равна длине дуги, ведущей из источника в соответствующий узел. Для определения ближайшего к источнику узла выберем временную пометку с минимальным значением и объявим ее постоянной пометкой. Затем, до тех пор пока стоку не будет приписана постоянная пометка, необходимо выполнить следующие две процедуры:

1. Рассмотреть оставшиеся узлы с временной пометкой. Сравнить величину каждой *временной пометки* с *суммой* величины последней из *постоянных пометок* и *длины* дуги, ведущей из соответствующего постоянно помеченного узла в рассматриваемый узел. Минимальная из двух сравниваемых величин определяется как новая временная пометка рассматриваемого узла. Отметим, что если величина старой временной пометки меньше второй из сравниваемых величин, то пометка остается прежней.

2. Среди временных пометок выбрать ту, значение которой минимально, и объявить ее постоянной пометкой. Если при этом постоянная пометка приписывается узлу t, то алгоритм завершает работу. В противном случае перейти на шаг 1. Данная процедура может быть выполнена с помощью таблицы решения, в которой столбцы соответствуют узлам сети, строки — шагам итеративного процесса, а ее элементы — постоянным и временным пометкам.

2.3.2. ПРИМЕР, ИЛЛЮСТРИРУЮЩИЙ РАБОТУ АЛГОРИТМА ДЕЙКСТРЫ

В данном разделе мы продемонстрируем работу алгоритма Дейкстры на примере, изображенном на рис. 2.11. Узел s здесь является источником, t — стоком, а числа c_{ij}, приписанные дугам, соответствуют их длине или стоимости.

Работа алгоритма начинается с того, что источнику s приписывается постоянная пометка $\boxed{0}$, а узлам $j=1,2,...,6$, t — временные пометки $\delta_j=c_{sj}$. Таким образом, $\delta_1=0$ и $\delta_j=\infty$ для $j=2,...,t$. Поскольку значение $\delta_1=0$ является минимальным среди всех временных пометок, узлу 1 приписывается постоянная пометка $\boxed{0}$.

Узлы 2 и 3 непосредственно связаны с узлом 1, последним из постоянно помеченных узлов. Отметим, что $\delta_1+c_{12}=0+3<\infty$

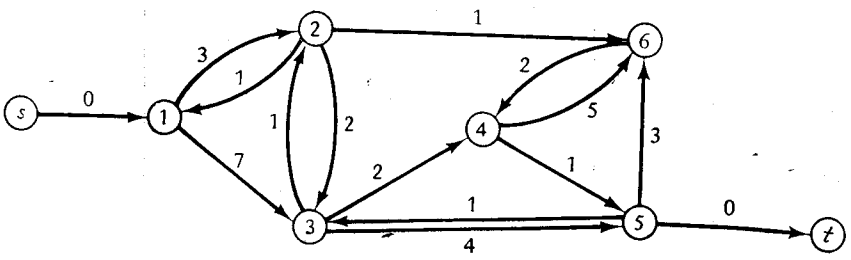


Рис. 2.11. Сетевая модель для иллюстрации алгоритма Дейкстры.

и $\delta_1+c_{13}=0+7<\infty$. Поэтому узлам 2 и 3 приписываются новые временные пометки $\delta_2=3$ и $\delta_3=7$ соответственно. Поскольку $\delta_2<\delta_3$, узлу 2 приписывается постоянная пометка $\delta_2=\boxed{3}$.

Узлы 3 и 6 непосредственно связаны с узлом 2. Кроме того, $\delta_2+c_{23}=3+2=5<7$ и $\delta_2+c_{26}=3+1=4<\infty$. Поэтому узлам 3 и 6 приписываются новые временные пометки $\delta_3=5$ и $\delta_6=4$ соответственно. Поскольку $\delta_6<\delta_3$, узел 6 становится помеченным постоянно, т. е. $\delta_6=\boxed{4}$.

На данном шаге временными являются пометки $\delta_3=5$ и $\delta_4=\delta_5=\delta_t=\infty$. Последним узлом, которому была приписана постоянная пометка, является узел 6. Он непосредственно связан только с узлом 4. Отметим, что $\delta_6+c_{64}=4+2=6<\infty$. Следовательно, узлу 4 приписывается новая временная пометка $\delta_4=6$. Поскольку $\delta_3=\min\{\delta_3, \delta_4, \delta_5, \delta_t\}$, узлу 3 приписы-

вается постоянная пометка $\delta_3 = \boxed{5}$. Аналогично проведем дальнейшие вычисления. Алгоритм заканчивает работу, когда узлу t приписывается постоянная пометка. Промежуточные результаты, полученные при решении данной задачи, приводятся в табл. 2.4.

Таблица 2.4. Результаты вычислений в задаче, решенной с помощью алгоритма Дейкстры

Шаг \ Узел	s	1	2	3	4	5	6	t
0	[0]	∞	∞	∞	∞	∞	∞	∞
1	[0]	0	∞	∞	∞	∞	∞	∞
2	[0]	[0]	∞	∞	∞	∞	∞	∞
3	[0]	[0]	3	7	∞	∞	∞	∞
4	[0]	[0]	[3]	7	∞	∞	∞	∞
5	[0]	[0]	[3]	5	∞	∞	4	∞
6	[0]	[0]	[3]	5	∞	∞	[4]	∞
7	[0]	[0]	[3]	5	6	∞	[4]	∞
8	[0]	[0]	[3]	[5]	6	∞	[4]	∞
9	[0]	[0]	[3]	[5]	6	9	[4]	∞
10	[0]	[0]	[3]	[5]	[6]	9	[4]	∞
11	[0]	[0]	[3]	[5]	[6]	7	[4]	∞
12	[0]	[0]	[3]	[5]	[6]	[7]	[4]	∞
13	[0]	[0]	[3]	[5]	[6]	[7]	[4]	7
14	[0]	[0]	[3]	[5]	[6]	[7]	[4]	[7]

Как видно из табл. 2.4, узлу t приписывается постоянная пометка $\delta_t = \boxed{7}$. Следовательно, длина кратчайшей цепи из узла s в узел t равна 7. Эта цепь состоит из дуг, для каждой из которых разность между значениями постоянных пометок ее концевых узлов равна длине этой дуги. Иными словами, если $\boxed{\delta_i}$ и $\boxed{\delta_j}$ — постоянные пометки узлов i и j соответственно, то условие, при выполнении которого эти узлы принадлежат кратчайшей цепи, может быть записано следующим образом:

$$\delta_j = \delta_i + c_{ij}. \qquad (2.50)$$

Соотношение (2.50) можно использовать рекурсивно, двигаясь от узла t к узлу s. Определив узел, непосредственно предшествующий t в кратчайшей цепи, будем повторять данную процедуру до тех пор, пока не достигнем узла s. Можно показать, что кратчайшая цепь в рассмотренной нами сети образуется последовательностью узлов s—1—2—6—4—5—t.

2.3.3. СВЕДЕНИЕ ЗАДАЧИ О ПОКУПКЕ АВТОМОБИЛЯ К ЗАДАЧЕ О КРАТЧАЙШЕЙ ЦЕПИ

При решении вопроса о том, когда следует производить покупку нового автомобиля, необходимо учитывать его стоимость и возрастающие эксплуатационные расходы. В данном разделе мы воспользуемся алгоритмом поиска кратчайшей цепи для

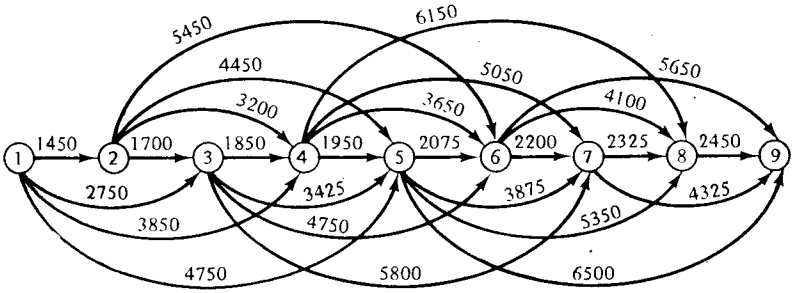


Рис. 2.12. Покупка нового автомобиля.

решения задачи, в которой требуется определить, как часто в течение восьмилетнего периода следует заменять автомобиль, чтобы минимизировать при этом общие затраты. Предполагается, что решение о покупке нового автомобиля может приниматься в начале каждого года, исходя из затрат на его приобретение, эксплуатационных расходов на период, в течение которого автомобиль будет использоваться, и ликвидационной стоимости машины в момент, когда она заменяется на новую. Предположим, что в начальный момент человек не имеет автомобиля и что он собирается покупать новый автомобиль по крайней мере каждые 4 года.

На рис. 2.12 изображена сеть для рассматриваемой задачи. Началу каждого года соответствует узел. Если покупка автомобиля производится в начале i-го года, а в начале j-го года автомобиль заменяется на новый, то данному варианту замены соответствует дуга (i, j). Пусть общие затраты в данном слу-

чае равны c_{ij}. Тогда значение параметра дуги (i, j) вычисляется по формуле

$$c_{ij} = P_i + \sum_{k=1}^{j-1} m_k - S_j, \qquad (2.51)$$

где P_i — стоимость автомобиля в начале i-го года, m_k — эксплуатационные расходы в течение k-го года, S_j — ликвидационная стоимость автомобиля в начале j-го года. На рис. 2.12 числа, приписанные дугам (i, j), соответствуют планируемым затратам c_{ij}. Вследствие инфляции, возрастания стоимости автомобиля и увеличения расходов на его техническое обслуживание общие затраты в различных периодах одинаковой длины не являются постоянными. Оптимальному решению данной задачи соответствует кратчайший путь из узла $s=1$ в узел $t=9$. Эту цепь можно рассматривать как цепь, минимизирующую стоимость единицы потока из узла 1 в узел 9 при условии, что стоимость единицы потока по дуге (i, j) равна c_{ij}. Результаты вычислений, проведенных при поиске кратчайшей цепи по алгоритму Дейкстры, даны в табл. 2.5. Для минимизации общих затрат покупку автомобиля следует производить в начале первого, пятого и девятого годов. Общие затраты при этом составляют 11 250 долл.

Таблица 2.5. Результаты вычислений

Шаг \ Узел	1	2	3	4	5	6	7	8	9
0	0	∞	∞	∞	∞	∞	∞	∞	∞
1	0	1450	2750	3850	4750	∞	∞	∞	∞
2	0	1450	2750	3850	4750	∞	∞	∞	∞
3	0	1450	2750	3850	4750	6900	∞	∞	∞
4	0	1450	2750	3850	4750	6900	∞	∞	∞
5	0	1450	2750	3850	4750	6900	8550	∞	∞
6	0	1450	2750	3850	4750	6900	8550	∞	∞
7	0	1450	2750	3850	4750	6900	8550	10 000	∞
8	0	1450	2750	3850	4750	6900	8550	10 000	∞
9	0	1450	2750	3850	4750	6825	8550	10 000	11 250
10	0	1450	2750	3850	4750	6825	8550	10 000	11 250
11	0	1450	2750	3850	4750	6825	8550	10 000	11 250
12	0	1450	2750	3850	4750	6825	8550	10 000	11 250
13	0	1450	2750	3850	4750	6825	8550	10 000	11 250
14	0	1450	2750	3850	4750	6825	8550	10 000	11 250
15	0	1450	2750	3850	4750	6825	8550	10 000	11 250

2.3.4. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ ДЕЙКСТРЫ

Назначение: нахождение кратчайшей цепи из источника в любой другой узел сети.

Локализация: подпрограмма DIJKA в пакете сетевой оптимизации.

Ограничения: программа обрабатывает сети, содержащие до 50 узлов и 50 дуг. Размеры сети можно увеличить, изменив границы массивов в операторах размерности, записанных в подпрограмме DIJKA и в основной программе.

Входные данные:

Набор 1. Одна карта с именем алгоритма в формате (A4).

Набор 2. Одна карта с числом узлов и числом дуг в сети в формате (2I10).

Набор 3. Общее число карт в данном наборе равно числу дуг в сети. С каждой карты считываются следующие величины: 1) номер начального узла дуги; 2) номер конечного узла дуги; 3) длина дуги.

Формат (4X, I6, I10, F10.2).

Набор 4. Данный набор состоит из одной карты, содержащей слово 'PEND' в формате (A4), которая указывает конец задачи.

Набор 5. Данный набор состоит из одной карты, содержащей слово 'EXIT' в формате (A4), которая указывает конец входных данных.

Составные части программы. Программа состоит из следующих подпрограмм: DIJKA (ввод и вычисления); TRACED (печать узлов кратчайшей цепи и ее длины).

Используемые переменные: I — начальный узел дуги, J — конечный узел дуги, VAL — длина дуги, D — рабочий массив длин дуг, OPTPAT — массив оптимального решения.

Используемый метод: данный алгоритм основан на методе, описанном в разд. 2.3.1.

Литература: [11].

2.3.5. ЗАДАЧА, СВЯЗАННАЯ С ТРАНСПОРТИРОВКОЙ НЕФТИ (ФИЛЛИПС)

Недавно компанией Black Gold Petroleum было найдено крупное месторождение нефти на северном склоне Аляски. Для его разработки возникла необходимость в строительстве нефтепровода, соединяющего северную часть полуострова с одним из шести возможных пунктов потребления на территории Соединенных Штатов. Для сооружения нефтепровода потребуется семь насосных станций, расположенных между подземным нефтехранилищем на северном склоне Аляски и пункта-

ми потребления. Каждая подстанция может быть построена на любом из имеющихся участков. Однако между некоторыми участками строительство нефтепровода не может проводиться в силу таких причин, как сложные физико-географические условия местности (наличие гор, озер), невозможность арендовать землю, ограничения, возникающие из-за необходимости охраны природы. Для любой пары насосных станций извест-

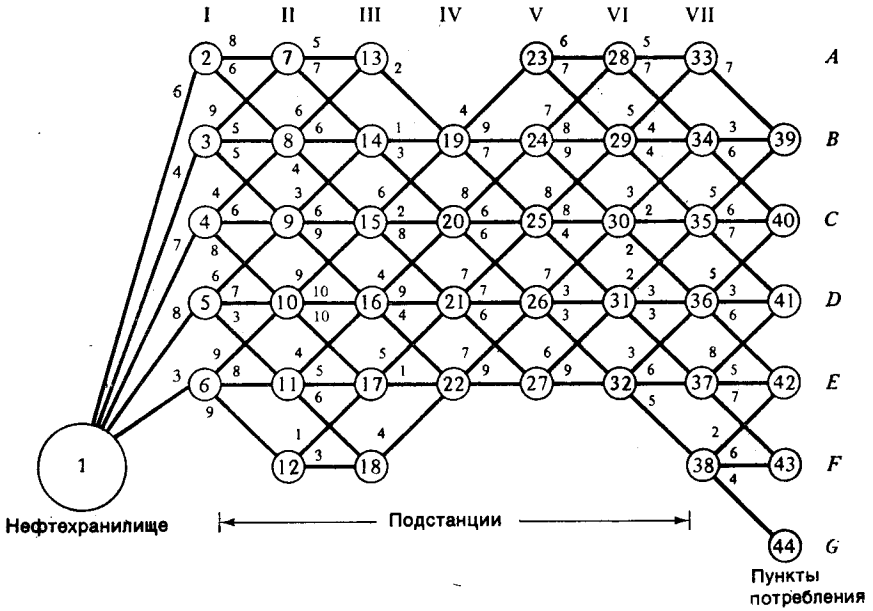


Рис. 2.13. Распределительная сеть.

ны затраты на строительство соединяющего их нефтепровода, которые зависят от местоположения каждой из них. Задача заключается в определении допустимого расположения насосных станций, при котором общие затраты на строительство нефтепровода минимальны. На рис. 2.13 изображена распределительная сеть для данной задачи. Каждый участок здесь представлен узлом, а каждая пара узлов, соответствующих участкам, между которыми может быть проложен нефтепровод, соединена дугой с приписанной ей стоимостью данного участка нефтепровода.

На практике при решении потоковых задач вычисления целесообразно проводить на ЭВМ. С этой целью на языке ФОРТРАН АНСИ была написана специальная программа, которая имеет стандартные форматы входных и выходных дан-

ных. Возможности использования программы и результаты ее работы иллюстрируются с помощью приведенного выше примера.

Длина, или стоимость, кратчайшей цепи из узла 1 в узел 44 равна 33. Оптимальный план размещения подстанций описывается следующей последовательностью узлов (данное решение получено с использованием программы, описанной в разд. 2.3.4):

Из узла 1 в узел 3 (расстояние равно 4,00).
Из узла 3 в узел 9 (расстояние равно 5,00).
Из узла 9 в узел 14 (расстояние равно 3,00).
Из узла 14 в узел 20 (расстояние равно 3,00).
Из узла 20 в узел 26 (расстояние равно 6,00).
Из узла 26 в узел 32 (расстояние равно 3,00).
Из узла 32 в узел 38 (расстояние равно 5,00).
Из узла 38 в узел 44 (расстояние равно 4,00).

Аналогично можно найти кратчайшую цепь из начального узла 1 в любой из конечных узлов 39, 40, 41, 42 и 43.

2.4. ЗАДАЧА О МНОГОПОЛЮСНОЙ КРАТЧАЙШЕЙ ЦЕПИ

В настоящем разделе рассматривается задача нахождения кратчайших цепей между всеми парами узлов сети $G=(\mathbf{N}, \mathbf{A})$. Если длина каждой дуги соответствует расстоянию или стоимости единицы потока по этой дуге, то кратчайшей цепью между двумя произвольными узлами является цепь, стоимость единицы потока по которой минимальна. Дуги из множества **A** могут быть ориентированными и неориентированными. Однако поскольку направление потока в неориентированных дугах нельзя определить заранее, то каждую такую дугу следует заменить двумя ориентированными дугами с противоположными направлениями и длинами, равными длине неориентированной дуги.

Предполагается, что длины дуг могут быть как положительными, так и отрицательными. Однако длина, или стоимость, каждого цикла или контура должна быть неотрицательной.

Алгоритм, описанный в настоящем разделе, первоначально был разработан Флойдом [12]. Мы предлагаем усовершенствованный вариант алгоритма, принадлежащий Ху [31]. Пусть $\mathbf{N}=\{1, 2,...,n\}$ — множество узлов, а c_{ij} — длина, или количественный параметр, дуги (i, j), направленной от узла i к узлу j. Обозначим через d^*_{ik} длину кратчайшей цепи из узла i в узел k.

Предположим, что величина d_{ik} представляет наилучшую оценку длины кратчайшей цепи, соединяющей узлы i и k. Мы знаем, что либо длина $\bar{d}_{ik}=d_{ij}+d_{jk}$ каждой кратчайшей цепи,

проходящей через промежуточный узел j, превосходит величину d_{ik}, либо текущая длина кратчайшей цепи равна $\bar{d}_{ik}$. Однако если длина $\bar{d}_{ik}$ новой цепи меньше d_{ik}, то текущее значение d_{ik} следует заменить на $\bar{d}_{ik}$. Алгоритм Флойда работает следующим образом. Первоначально за длину кратчайшей цепи между двумя произвольными узлами i и k принимается длина дуги (i, k), соединяющей эти узлы. Затем последовательно

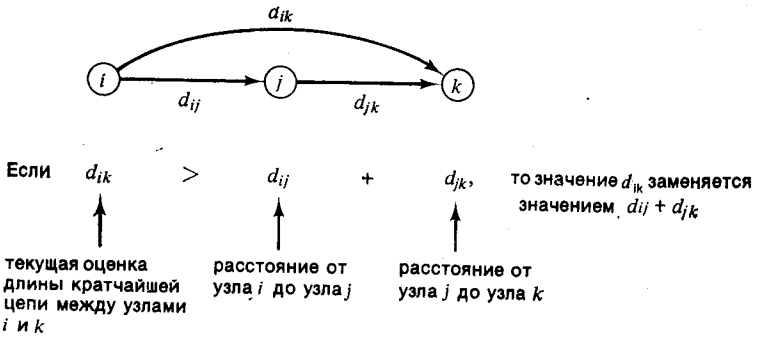


Рис. 2.14. Пример процедуры сравнения, выполняемой при работе алгоритма Флойда.

проверяются всевозможные промежуточные узлы, расположенные между i и k. Если длина цепи, проходящей через некоторый промежуточный узел, меньше текущего значения d_{ik}, то переменной d_{ik} присваивается новое значение. Данная процедура повторяется для всевозможных пар узлов, пока не будут получены все значения d^*_{ik}.

Для любых трех различных узлов i, j и k сформулированные выше условия могут быть записаны в виде неравенства

$$d_{ij} + d_{jk} \geqslant d_{ik}. \quad i \neq j \neq k, \tag{2.52}$$

поскольку в противном случае кратчайшая цепь из узла i в узел k должна содержать узел j и тогда величина d_{ik} не была бы равной длине кратчайшей цепи. В алгоритме Флойда начальным значением переменной d_{ik} является величина c_{ik}, а затем данная оценка последовательно улучшается до тех пор, пока не будет найдена кратчайшая цепь между узлами i и k. Рис. 2.14 иллюстрирует выполнение процедуры сравнения для пары узлов i и k.

Алгоритм Флойда позволяет решать задачу о многополюсной кратчайшей цепи (пути) для сети из n узлов за n итераций. Для того чтобы перейти к формальному описанию итеративной процедуры поиска решения, обозначим символом d^j_{ik} оценку длины кратчайшей цепи из узла i в узел k, полученную на j-й итерации.

Способ получения на j-й итерации величины d^j_{ik} может быть описан с помощью следующей трехместной операции:

$$d_{ik}{}^j = \min [d_{ik}{}^{j-1};\ d_{ij}{}^{j-1} + d_{jk}{}^{j-1}], \quad i \neq j \neq k. \tag{2.53}$$

Если трехместную операцию, определенную выражением (2.53), выполнять с данной парой узлов i и k и всеми узлами j $(i \neq j \neq k)$ в порядке возрастания номеров j, то значение d^n_{ik}, полученное на последней итерации, будет равно длине кратчайшей цепи из узла i в узел k.

Мы воспользуемся матричным методом, который позволит нам запоминать длины кратчайших цепей и их дуги. На каждой итерации алгоритма строятся две матрицы. Первая из них, называемая *матрицей длин кратчайших путей*, содержит текущие оценки длин кратчайших цепей, т. е. на j-й итерации данная матрица определяется как $\mathbf{D}^j = [d^j_{ik}]$. Алгоритм начинает работу при $\mathbf{D}^0 = [d^0_{ik}]$, где $d^0_{ik} = c_{ik}$. Затем, выполняя трехместную операцию (2.53) со всеми элементами матрицы $\mathbf{D}^0$, получаем $\mathbf{D}^1$ и т. д. до тех пор, пока для каждой пары узлов не будет выполнен критерий оптимальности (2.52)

Вторая матрица, называемая *матрицей маршрутов*, служит для нахождения промежуточных узлов (если таковые имеются) кратчайших цепей. На j-й итерации она определяется как $\mathbf{R}^j = [r^j_{ik}]$, где r^j_{ik} — первый промежуточный узел кратчайшей цепи из i в k, выбираемый среди узлов множества $\{1, 2, ..., j\}$ $(i \neq j \neq k)$. Алгоритм начинает работу при $\mathbf{R}^0 = [r^0_{ik}]$, где $r^0_{ik} = k$. На j-й итерации узел r^j_{ik} может быть получен из следующего соотношения:

$$r^j_{ik} = \begin{cases} j, & \text{если } d_{ik}{}^{j-1} > d_{ij}{}^{j-1} + d_{jk}{}^{j-1}, \\ r_{ik}{}^{j-1} & \text{в противном случае.} \end{cases} \tag{2.54}$$

После построения матриц $\mathbf{D}^0$ и $\mathbf{R}^0$ нужно для каждого $j = 1, 2, ..., n$, используя для вычислений элементы матрицы $\mathbf{D}^{j-1}$, полученной на $(j-1)$-й итерации, выполнить следующую процедуру:

Шаг 1. Вычеркнуть элементы j-й строки и j-го столбца. Назовем эти множества элементов *базовой строкой* и *базовым столбцом* соответственно.

Шаг 2. Каждый элемент $d_{ik}{}^{j-1}$ $(k \neq j)$ матрицы расстояний (начиная с первого элемента, т. е. расположенного в левом верхнем углу матрицы), который не принадлежит ни базовой строке, ни базовому столбцу, сравнить с суммой элементов $d_{jk}{}^{j-1}$ и $d_{ij}{}^{j-1}$ базовой строки и базового столбца соответственно.

Если выполняется неравенство $d_{ij}{}^{j-1} + d_{jk}{}^{j-1} \geqslant d_{ik}{}^{j-1}$, то выбрать новые значения для i и k, перейти к следующему эле-

менту d_{ik}^{j-1} и снова выполнить шаг 2. Если $d_{ik}^{j-1} > d_{ij}^{j-1} + d_{jk}^{j-1}$, то элементу d_{ik}^{j-1} *матрицы длин кратчайших путей* присвоить значение $d_{ik}^{j} = d_{ij}^{j-1} + d_{jk}^{j-1}$, а соответствующему элементу *матрицы маршрутов* — значение, равное j. После просмотра всех элементов вновь перейти к выполнению шага 1 при $j = j+1$. При $j = n$ будут построены матрица расстояний и матрица маршрутов, представляющие решение задачи.

Согласно (2.53), при получении $\mathbf{D}^j$ из $\mathbf{D}^{j-1}$ трехместную операцию, позволяющую установить факт принадлежности базового узла кратчайшей цепи, следует выполнять только с элементами матрицы $\mathbf{D}^{j-1}$, не принадлежащими ни базовой строке, ни базовому столбцу.

Отметим, что при исследовании каждого элемента d_{ik}^{j-1} $(i,\ j \neq k)$ может быть использована процедура, похожая на симплекс-метод. После того как выбран элемент d_{ik}^{j-1}, замену производить не следует, если $d_{ik}^{j-1} = \infty$ и одно из двух значений d_{ij}^{j-1} или d_{jk}^{j-1} равно ∞. Если $d_{ik}^{j-1} \neq \infty$ и одно из двух значений d_{ij}^{j-1} или d_{jk}^{j-1} превосходит d_{ik}^{j-1}, то замену также производить не следует. Поэтому необходимо рассмотреть лишь случай, когда $d_{ik}^{j-1} \neq \infty$ и оба значения d_{ij}^{j-1} и $d_{jk}^{j-1} \neq \infty$ меньше d_{ik}^{j-1}. Таким образом, для выполнения каждой итерации необходимо руководствоваться следующими правилами, позволяющими упростить вычисления:

1. $d_{ij}^{j-1} = \infty$, т. е. i-й элемент базового столбца равен ∞. Из (2.53) следует, что в данном случае значение d_{ik}^{j} должно оставаться равным d_{ik}^{j-1}. Поэтому трехместную операцию выполнять не нужно.

2. $d_{jk}^{j-1} = \infty$, т. е. j-й элемент базовой строки равен ∞. Из (2.53) следует, что в данном случае значение d_{ik}^{j} должно оставаться равным d_{ik}^{j-1}. Поэтому трехместную операцию выполнять не нужно.

Рассматриваемый алгоритм может быть описан в виде последовательности двух шагов вычислений. Начальные значения $\mathbf{D}^0$ и $\mathbf{R}^0$ для итеративного выполнения этих шагов определяются по формулам $\mathbf{D}^0 = [c_{ik}]$, $\mathbf{R}^0 = [k]$, а параметру j присваивается начальное значение, равное нулю. Данный параметр используется для обозначения базового узла на каждой итерации.

Шаг 1. $j = j+1$. Определить узел j как базовый узел и вычеркнуть базовую строку и базовый столбец матрицы $\mathbf{D}^{j-1}$. Вычеркнуть также строки и столбцы матрицы $\mathbf{D}^{j-1}$, содержащие те элементы, равные ∞, которые принадлежат базовой строке или базовому столбцу.

Шаг 2. Сравнить каждый невычеркнутый элемент d_{ik}^{j-1} с суммой следующих двух элементов:

а) элемент d_{ij}^{j-1}, расположенный на пересечении базового столбца и строки, содержащей d_{ik}^{j-1};

б) элемент d_{jk}^{j-1}, расположенный на пересечении базовой строки и столбца, содержащего d_{ik}^{j-1}.

Если сумма этих двух элементов меньше d_{ik}^{j-1}, то положить $d_{ik}^{j}=d_{ij}^{j-1}+d_{jk}^{j-1}$, $r_{ik}^{j}=j$. В противном случае положить $d_{ik}^{j}=d_{ik}^{j-1}$, $r_{ik}^{j}=r_{ik}^{j-1}$. После того как будут рассмотрены все невычеркнутые элементы матрицы $\mathbf{D}^{j-1}$, проверить выполнение условия $j=n$. Если оно выполнено, то работа алгоритма завершается, в противном случае перейти к шагу 1.

2.4.1. ПРИМЕР ЗАДАЧИ О МНОГОПОЛЮСНОЙ КРАТЧАЙШЕЙ ЦЕПИ

Для иллюстрации работы алгоритма Флойда найдем кратчайшую цепь между всеми парами узлов сети, изображенной на рис. 2.15. Начальные значения элементов матрицы длин кратчайших путей и матрицы маршрутов могут быть выбраны так, как это показано ниже. Поскольку $n=\infty$, число итераций в алгоритме также равно 8.

$$\mathbf{D}^0=\begin{array}{c} \\ 1\\2\\3\\4\\5\\6\\7\\8\end{array}\begin{array}{c}\begin{array}{cccccccc}1&2&3&4&5&6&7&8\end{array}\\ \begin{bmatrix}0&9&\infty&3&\infty&\infty&\infty&\infty\\9&0&2&\infty&7&\infty&\infty&\infty\\\infty&2&0&2&4&8&6&\infty\\3&\infty&2&0&\infty&\infty&5&\infty\\\infty&7&4&\infty&0&10&\infty&\infty\\\infty&\infty&8&\infty&10&0&7&\infty\\\infty&\infty&6&5&\infty&7&0&\infty\\\infty&\infty&\infty&\infty&9&12&10&0\end{bmatrix}\end{array},$$

$$\mathbf{R}^0=\begin{array}{c} \\ 1\\2\\3\\4\\5\\6\\7\\8\end{array}\begin{array}{c}\begin{array}{cccccccc}1&2&3&4&5&6&7&8\end{array}\\ \begin{bmatrix}1&2&3&4&5&6&7&8\\1&2&3&4&5&6&7&8\\1&2&3&4&5&6&7&8\\1&2&3&4&5&6&7&8\\1&2&3&4&5&6&7&8\\1&2&3&4&5&6&7&8\\1&2&3&4&5&6&7&8\\1&2&3&4&5&6&7&8\end{bmatrix}\end{array}$$

Итерация 1. Узел $j=1$ определяется как базовый. Следовательно, в матрице $\mathbf{D}^0$ можно вычеркнуть первую строку и первый столбец. Кроме того, столбцы 3, 5, 6, 7 и 8 содержат элементы, равные ∞ и принадлежащие базовой строке, а строки 3, 5,

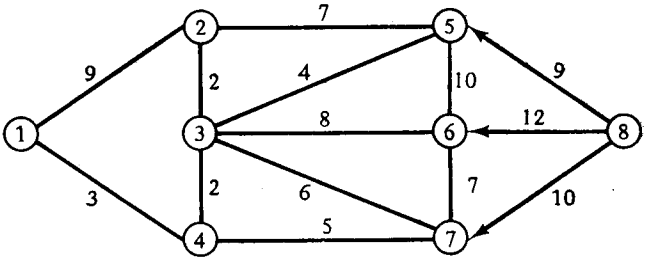


Рис. 2.15. Сеть в задаче о многополюсной кратчайшей цепи.

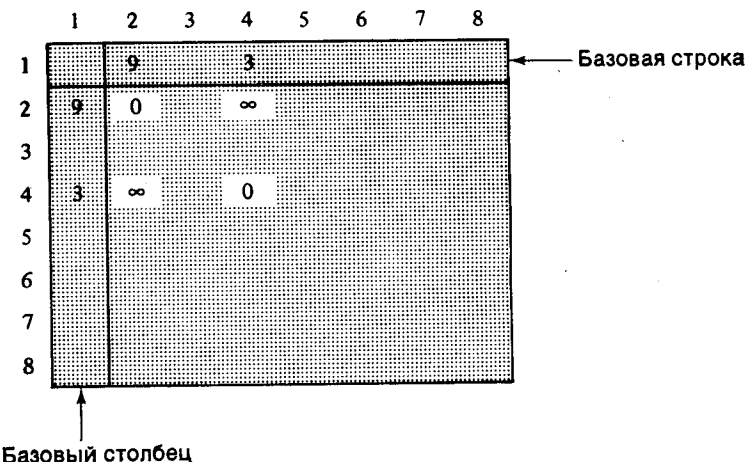


Рис. 2.16. Элементы, исследуемые (с помощью трехместной операции) на первой итерации.

6, 7 и 8 содержат элементы, равные ∞ и принадлежащие базовому столбцу. Поэтому, для того чтобы определить, приведет ли использование узла 1 к более коротким цепям, следует исследовать (с помощью трехместной операции) лишь элементы матрицы $\mathbf{D}^0$, изображенные на рис. 2.16.

Поскольку диагональные элементы матрицы $\mathbf{D}^0$ можно не рассматривать, необходимо исследовать лишь оценки d^0_{24} и d^0_{42}. Применение трехместной операции дает следующие результаты:

$$d^1_{24} = \min [d^0_{24};\ d^0_{21} + d^0_{14}] = \min [\infty;\ 9+3] = 12,$$

$$d^1_{42} = \min [d^0_{42};\ d^0_{41} + d^0_{12}] = \min [\infty;\ 3+9] = 12.$$

Отметим, что оценки $d^1{}_{24}=12$ и $d^1{}_{42}=12$ лучше оценок $d^0{}_{24}=\infty$ и $d^0{}_{42}=\infty$ соответственно. Поскольку использование базового узла приводит к более коротким цепям из узла 2 в узел 4 и из узла 4 в узел 2, то полагаем $r^1{}_{24}=1$ и $r^1{}_{42}=1$. Все остальные элементы матриц $\mathbf{D}^0$ и $\mathbf{R}^0$ остаются без изменения. Теперь мы можем выписать матрицы $\mathbf{D}^1$ и $\mathbf{R}^1$:

$$\mathbf{D}^1=\begin{array}{c} \\ 1\\2\\3\\4\\5\\6\\7\\8 \end{array}\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}\\ \begin{bmatrix} 0 & 9 & \infty & 3 & \infty & \infty & \infty & \infty \\ 9 & 0 & 2 & 12 & 7 & \infty & \infty & \infty \\ \infty & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & 12 & 2 & 0 & \infty & \infty & 5 & \infty \\ \infty & 7 & 4 & \infty & 0 & 10 & \infty & \infty \\ \infty & \infty & 8 & \infty & 10 & 0 & 7 & \infty \\ \infty & \infty & 6 & 5 & \infty & 7 & 0 & \infty \\ \infty & \infty & \infty & \infty & 9 & 12 & 10 & 0 \end{bmatrix} \end{array},$$

$$\mathbf{R}^1=\begin{array}{c} \\ 1\\2\\3\\4\\5\\6\\7\\8 \end{array}\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}\\ \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 1 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 1 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix} \end{array}.$$

Итерация 2. Определяем узел $j=2$ как базовый и вычеркиваем в матрице $\mathbf{D}^1$ вторую строку и второй столбец. Кроме того, столбцы 6, 7 и 8 содержат элементы, равные ∞ и принадлежащие базовой строке, а строки 6, 7 и 8 содержат элементы, равные ∞ и принадлежащие базовому столбцу. Поэтому, для того чтобы определить, приведет ли использование узла 2 к более коротким цепям, следует исследовать (с помощью трехместной операции) лишь элементы матрицы $\mathbf{D}^1$, изображенные на рис. 2.17.

Поскольку диагональные элементы матрицы $\mathbf{D}^1$ можно не рассматривать, то необходимо исследовать элементы $d^1{}_{13}$, $d^1{}_{14}$, $d^1{}_{15}$. $d^1{}_{31}$, $d^1{}_{34}$, $d^1{}_{35}$, $d^1{}_{41}$, $d^1{}_{43}$, $d^1{}_{45}$, $d^1{}_{51}$, $d^1{}_{53}$ и $d^1{}_{54}$. Нетрудно про-

верить, что улучшены могут быть только оценки d^1_{13}, d^1_{15}, d^1_{31}, d^1_{45}, d^1_{51} и d^1_{54}. Новые оценки получаются следующим образом:

$$d^2_{13} = \min[d^1_{13};\ d^1_{12}+d^1_{23}] = \min[\infty;\ 9+2] = 11,$$
$$d^2_{15} = \min[d^1_{15};\ d^1_{12}+d^1_{25}] = \min[\infty;\ 9+7] = 16,$$
$$d^2_{31} = \min[d^1_{31};\ d^1_{32}+d^1_{21}] = \min[\infty;\ 2+9] = 11,$$
$$d^2_{45} = \min[d^1_{45};\ d^1_{42}+d^1_{25}] = \min[\infty;\ 12+7] = 19,$$
$$d^2_{51} = \min[d^1_{51};\ d^1_{52}+d^1_{21}] = \min[\infty;\ 7+9] = 16,$$
$$d^2_{54} = \min[d^1_{54};\ d^1_{52}+d^1_{24}] = \min[\infty;\ 7+12] = 19.$$

Таким образом, $r^2_{13}=r^2_{15}=r^2_{31}=r^2_{45}=r^2_{51}=r^2_{54}=2$ и $r^2_{ik}=r^1_{ik}$ для всех элементов, таких, что $d^2_{ik}=d^1_{ik}$. Новые матрицы $\mathbf{D}^2$ и $\mathbf{R}^2$ имеют вид

$$\mathbf{D}^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{bmatrix} 0 & 9 & 11 & 3 & 16 & \infty & \infty & \infty \\ 9 & 0 & 2 & 12 & 7 & \infty & \infty & \infty \\ 11 & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & 12 & 2 & 0 & 19 & \infty & 5 & \infty \\ 16 & 7 & 4 & 19 & 0 & 10 & \infty & \infty \\ \infty & \infty & 8 & \infty & 10 & 0 & 7 & \infty \\ \infty & \infty & 6 & 5 & \infty & 7 & 0 & \infty \\ \infty & \infty & \infty & \infty & 9 & 12 & 10 & 0 \end{bmatrix} \end{array},$$

$$\mathbf{R}^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 4 & 2 & 6 & 7 & 8 \\ 1 & 2 & 3 & 1 & 5 & 6 & 7 & 8 \\ 2 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 1 & 3 & 4 & 2 & 6 & 7 & 8 \\ 2 & 2 & 3 & 2 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix} \end{array}.$$

Выполняя аналогичные вычисления на итерациях 3, 4, 5, 6, 7, 8, мы получим результаты, приведенные в табл. 2.6. Нетрудно проверить, что оценки, полученные на итерациях 6, 7 и 8, не могут быть улучшены. Следовательно, оптимальное реше-

Таблица 2.6. Результаты вычислений для модельного примера

Итерация j	D^j	R^j
3	$\begin{bmatrix} 0 & 9 & 11 & 3 & 15 & 19 & 17 & \infty \\ 9 & 0 & 2 & 4 & 6 & 10 & 8 & \infty \\ 11 & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & 4 & 2 & 0 & 6 & 10 & 5 & \infty \\ 15 & 6 & 4 & 6 & 0 & 10 & 10 & \infty \\ 19 & 10 & 8 & 10 & 10 & 0 & 7 & \infty \\ 17 & 8 & 6 & 5 & 10 & 7 & 0 & \infty \\ \infty & \infty & \infty & \infty & 9 & 12 & 10 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 2 & 4 & 2 & 2 & 2 & 8 \\ 1 & 2 & 3 & 3 & 3 & 3 & 3 & 8 \\ 2 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 3 & 3 & 4 & 3 & 3 & 7 & 8 \\ 2 & 3 & 3 & 3 & 5 & 6 & 3 & 8 \\ 3 & 3 & 3 & 3 & 5 & 6 & 7 & 8 \\ 3 & 3 & 3 & 4 & 3 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$
4	$\begin{bmatrix} 0 & 7 & 5 & 3 & 9 & 13 & 8 & \infty \\ 7 & 0 & 2 & 4 & 6 & 10 & 8 & \infty \\ 5 & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & 4 & 2 & 0 & 6 & 10 & 5 & \infty \\ 9 & 6 & 4 & 6 & 0 & 10 & 10 & \infty \\ 13 & 10 & 8 & 10 & 10 & 0 & 7 & \infty \\ 8 & 8 & 6 & 5 & 10 & 7 & 0 & \infty \\ \infty & \infty & \infty & \infty & 9 & 12 & 10 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 4 & 4 & 4 & 4 & 4 & 4 & 8 \\ 4 & 2 & 3 & 3 & 3 & 3 & 3 & 8 \\ 4 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 3 & 3 & 4 & 3 & 3 & 7 & 8 \\ 4 & 3 & 3 & 3 & 5 & 6 & 3 & 8 \\ 4 & 3 & 3 & 3 & 5 & 6 & 7 & 8 \\ 4 & 3 & 3 & 4 & 3 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$
5	$\begin{bmatrix} 0 & 7 & 5 & 3 & 9 & 13 & 8 & \infty \\ 7 & 0 & 2 & 4 & 6 & 10 & 8 & \infty \\ 5 & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & 4 & 2 & 0 & 6 & 10 & 5 & \infty \\ 9 & 6 & 4 & 6 & 0 & 10 & 10 & \infty \\ 13 & 10 & 8 & 10 & 10 & 0 & 7 & \infty \\ 8 & 8 & 6 & 5 & 10 & 7 & 0 & \infty \\ 18 & 15 & 13 & 15 & 9 & 12 & 10 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 4 & 4 & 4 & 4 & 4 & 4 & 8 \\ 4 & 2 & 3 & 3 & 3 & 3 & 3 & 8 \\ 4 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 3 & 3 & 4 & 3 & 3 & 7 & 8 \\ 4 & 3 & 3 & 3 & 5 & 6 & 3 & 8 \\ 4 & 3 & 3 & 3 & 5 & 6 & 7 & 8 \\ 4 & 3 & 3 & 4 & 3 & 6 & 7 & 8 \\ 5 & 5 & 5 & 5 & 5 & 6 & 7 & 8 \end{bmatrix}$
6	Остается неизменной	Остается неизменной
7	Остается неизменной	Остается неизменной
8	Остается неизменной	Остается неизменной

ние соответствует матрицам $\mathbf{D}^5$ и $\mathbf{R}^5$. Для иллюстрации результатов, приведенных в табл. 2.6, рассмотрим кратчайшую цепь из узла 1 в узел 5. Длина этой цепи равна $d^5_{15}=9$. Для того чтобы найти соответствующую последовательность узлов, обратимся к матрице $\mathbf{R}^5$. Поскольку значение r^5_{15} равно 4, то узел 4 является первым промежуточным узлом в кратчайшей цепи из узла 1 в узел 5. Затем, для того чтобы найти узел, следующий за узлом 4 в цепи, ведущей к узлу 5, мы определяем значение r^5_{45}. Данное значение равно 3. Точно так же, поскольку $r^5_{35}=5$, узел 5 следует за узлом 3. Следовательно, кратчайшая цепь из узла 1 в узел 5 определяется последовательностью узлов 1, 4, 3, 5.

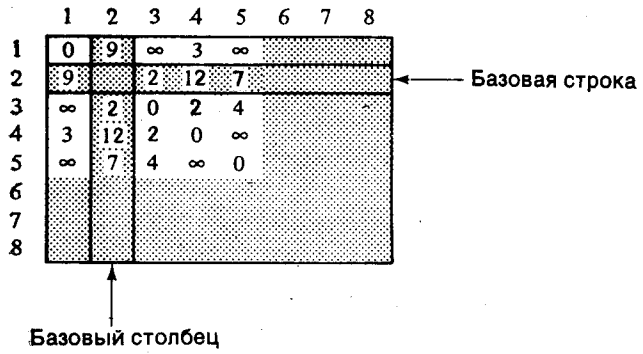


Рис. 2.17. Элементы, исследуемые (с помощью трехместной операции) на второй итерации.

2.4.2. ПРИМЕНЕНИЕ ЗАДАЧИ О МНОГОПОЛЮСНОЙ КРАТЧАЙШЕЙ ЦЕПИ ПРИ ПРОЕКТИРОВАНИИ СИСТЕМЫ ДОСТАВКИ ПОЧТЫ

Крупное учреждение планирует разработать систему внутренней доставки почты, основанную на использовании линий пневматической связи, для распределения корреспонденции между восьмью отделами. Некоторые отделы будут только отсылать поступающую корреспонденцию в другие отделы, не имея при этом возможности получать почту. Все остальные отделы смогут отправлять и получать почту без каких-либо ограничений.

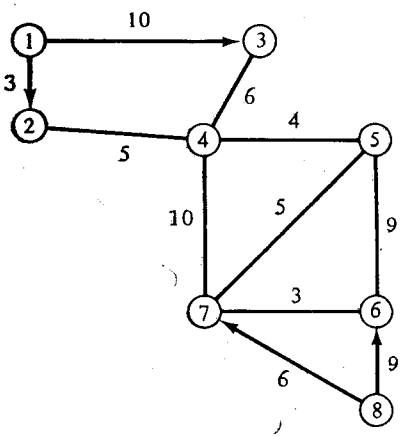


Рис. 2.18. Система доставки почты.

Расположение линий, изображенное на рис. 2.18, было установлено заранее. Каждый отдел представлен узлом, а каждая линия — дугой. Ориентированные дуги соответствуют распределительным звеньям, которые могут быть использованы для посылки почты только в указанном направлении. Числа, приписанные дугам, равны расстояниям между соответствующими отделами.

Для того чтобы каждый отдел при пересылке почты в другой отдел смог бы определить оптимальный путь, необходимо

заготовить таблицу, указывающую кратчайшую цепь между каждой парой отделов. Такая таблица может быть построена с помощью алгоритма Флойда. В рассматриваемой задаче (рис. 2.18) начальные значения элементов матрицы длин кратчайших путей и матрицы маршрутов могут быть выбраны следующим образом:

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 3 & 10 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & 5 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & 5 & 6 & 0 & 4 & \infty & 10 & \infty \\ \infty & \infty & \infty & 4 & 0 & 9 & 5 & \infty \\ \infty & \infty & \infty & \infty & 9 & 0 & 3 & \infty \\ \infty & \infty & \infty & 10 & \infty & 3 & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 9 & 6 & 0 \end{bmatrix},$$

$$\mathbf{R}^0 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}.$$

Читателю предлагается проверить, что оптимальное решение соответствует следующим матрицам:

$$\mathbf{D}^8 = \begin{bmatrix} 0 & 3 & 10 & 8 & 12 & 20 & 17 & \infty \\ \infty & 0 & 11 & 5 & 9 & 17 & 14 & \infty \\ \infty & 11 & 0 & 6 & 10 & 18 & 15 & \infty \\ \infty & 5 & 6 & 0 & 4 & 12 & 10 & \infty \\ \infty & 9 & 10 & 4 & 0 & 8 & 5 & \infty \\ \infty & 17 & 18 & 12 & 9 & 0 & 3 & \infty \\ \infty & 15 & 15 & 19 & 15 & 3 & 0 & \infty \\ \infty & 20 & 21 & 15 & 11 & 9 & 6 & 0 \end{bmatrix},$$

$$\mathbf{R}^8 = \begin{bmatrix} 1 & 2 & 3 & 2 & 4 & 7 & 5 & 8 \\ 1 & 2 & 4 & 4 & 4 & 7 & 5 & 8 \\ 1 & 4 & 3 & 4 & 4 & 7 & 5 & 8 \\ 1 & 2 & 3 & 4 & 5 & 7 & 5 & 8 \\ 1 & 4 & 4 & 4 & 5 & 7 & 7 & 8 \\ 1 & 7 & 7 & 7 & 7 & 6 & 7 & 8 \\ 1 & 5 & 5 & 5 & 5 & 6 & 7 & 8 \\ 1 & 7 & 7 & 7 & 7 & 6 & 7 & 8 \end{bmatrix}.$$

Длина кратчайшей цепи из узла 8 в узел 2 равна $d^8{}_{82}=20$. Для определения узлов, соответствующих этой цепи, обратимся к матрице $\mathbf{R}^8$. Нетрудно заметить, что $r^8{}_{82}=7$, $r^8{}_{72}=5$, $r^8{}_{52}=4$ и $r^8{}_{42}=2$. Следовательно, кратчайшая цепь из узла 8 в узел 2 определяется последовательностью узлов 8, 7, 5, 4, 2.

2.4.3. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ О МНОГОПОЛЮСНОЙ КРАТЧАЙШЕЙ ЦЕПИ

Назначение: решение задачи о многополюсной кратчайшей цепи.

Локализация: подпрограмма MULTSH в пакете сетевой оптимизации.

Ограничения: программа позволяет обрабатывать сети, содержащие до 50 узлов и 50 дуг. Размеры сети можно увеличить, изменив границы массивов в операторах размерности, записанных в подпрограмме MULTSH и в основной программе.

Входные данные:

Набор 1. Одна карта с именем алгоритма MTSC в формате (A4).

Набор 2. Одна карта с числом узлов и числом дуг в сети в формате (2I10).

Набор 3. Общее число карт в данном наборе равно числу дуг в сети. С каждой карты считываются следующие величины: 1) номер начального узла дуги; 2) номер конечного узла дуги; 3) длина дуги.

Формат (4X, I6, I10, F10.2).

Набор 4. Данный набор состоит из одной карты, содержащей слово 'PEND' в формате (A4), которая указывает конец задачи.

Набор 5. Данный набор состоит из одной карты, содержащей слово 'EXIT' в формате (A4), которая указывает конец входных данных.

Составные части программы. Программа состоит из следующих подпрограмм: MULTSH (ввод и вычисления); PRNTRX (печать исходной матрицы); PRNTRY (печать оптимального решения).

Используемые переменные: I — начальный узел дуги, J — конечный узел дуги, B — длина дуги, D — рабочий массив длин дуг.

Используемый метод: данный алгоритм основан на методе, в котором в качестве начальной оценки длины кратчайшей цепи из узла I в узел K выбирается длина дуги из I в K. На J-й итерации производится сравнение длины цепи из I в K, содержащей промежуточные узлы из множества {1, 2, ..., J}, с длиной цепи, проходящей через узел J с промежуточными узлами из множества {1, 2, ..., J—1}. Если использование узла J позволяет получить более короткую цепь, то длина последней принимается за новую оценку длины кратчайшей цепи.

Литература: [31].

2.4.4. СОСТАВЛЕНИЕ МАРШРУТА ПЕРЕГОНА ВАГОНОВ

Среди общих расходов на транспортировку груза, поступающего в порт Хьюстон и вывозимого из него, значительную долю составляют затраты на перегон вагонов из одного пункта портового комплекса в другой. В сети, изображенной на рис. 2.19, узел 1 соответствует станции прибытия поездов, из которой вагоны прибывающих составов перегоняются к различным пунктам назначения в портовом комплексе, обозначенным остальными узлами сети. Числа, приписанные дугам, представляют собой затраты на перегон вагонов по соответствующей ветке.

Рассмотрим следующий пример. Поезд состоит из 200 вагонов, 100 из которых хопперы, груженные зерном. Хопперы с зерном должны быть доставлены к элеваторам, расположенным в узлах 3, 6 и 10. После разгрузки они могут быть либо доставлены на заводы, производящие удобрения (узлы 2, 5 и 8), либо перевезены на станцию отправления (узел 11). Остальные типы вагонов, такие, как крытые товарные вагоны, вагоны-платформы и цистерны, следует доставить в пункты 4, 7 и 9, разгрузить их там и перегнать на станцию отправления.

Применение алгоритма решения задачи о многополюсной кратчайшей цепи к транспортной сети, изображенной на рис. 2.19, позволит минимизировать затраты на перегон вагонов из каждого пункта портового комплекса в любой другой пункт. Например, решение данной задачи позволит определить оптимальный путь перевозки хопперов, соединяющий станцию прибытия, элеваторы, заводы, производящие удобрения, и станцию отправки. Данная задача была решена с помощью про-

ЗАДАЧА О МНОГОПОЛЮСНОЙ КРАТЧАЙШЕЙ ЦЕПИ

ИСХОДНАЯ МАТРИЦА

	1	2	3	4	5
1	0.0	4.00	3.00	6.00	9999999.00
2	4.00	0.0	9999999.00	5.00	3.00
3	3.00	9999999.00	0.0	4.00	9999999.00
4	6.00	5.00	4.00	0.0	2.00
5	9999999.00	3.00	9999999.00	2.00	0.0
6	9999999.00	9999999.00	6.00	5.00	9999999.00
7	9999999.00	9999999.00	9999999.00	2.00	2.00
8	9999999.00	9999999.00	9999999.00	9999999.00	4.00
9	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
10	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
11	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00

	6	7	8	9	10
1	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
2	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
3	6.00	9999999.00	9999999.00	9999999.00	9999999.00
4	5.00	2.00	9999999.00	9999999.00	9999999.00
5	9999999.00	2.00	4.00	9999999.00	9999999.00
6	0.0	1.00	9999999.00	2.00	5.00
7	1.00	0.0	2.00	5.00	9999999.00
8	9999999.00	2.00	0.0	2.00	9999999.00
9	2.00	5.00	2.00	0.0	3.00
10	5.00	9999999.00	9999999.00	3.00	0.0
11	9999999.00	9999999.00	7.00	8.00	4.00

	11
1	9999999.00
2	9999999.00
3	9999999.00
4	9999999.00
5	9999999.00
6	9999999.00
7	9999999.00
8	7.00
9	8.00
10	4.00
11	0.0

МАТРИЦА КРАТЧАЙШИХ РАССТОЯНИЙ

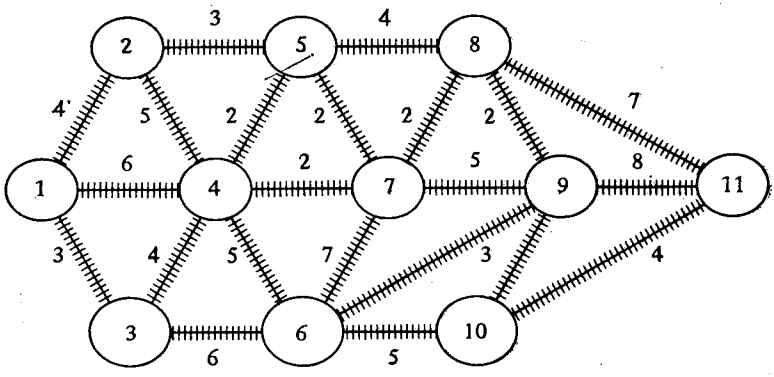
	1	2	3	4	5
1	0.0	4.00	3.00	6.00	7.00
2	4.00	0.0	7.00	5.00	3.00
3	3.00	7.00	0.0	4.00	6.00
4	6.00	5.00	4.00	0.0	2.00
5	7.00	3.00	6.00	2.00	0.0
6	9.00	6.00	6.00	3.00	3.00
7	8.00	5.00	6.00	2.00	2.00
8	10.00	7.00	8.00	4.00	4.00
9	11.00	8.00	8.00	5.00	5.00
10	14.00	11.00	11.00	8.00	8.00
11	17.00	14.00	15.00	11.00	11.00

	6	7	8	9	10	11
1	9.00	8.00	10.00	11.00	14.00	17.00
2	6.00	5.00	7.00	8.00	11.00	14.00
3	6.00	6.00	8.00	8.00	11.00	15.00
4	3.00	2.00	4.00	5.00	8.00	11.00
5	3.00	2.00	4.00	5.00	8.00	11.00
6	0.0	1.00	3.00	2.00	5.00	9.00
7	1.00	0.0	2.00	3.00	6.00	9.00
8	3.00	2.00	0.0	2.00	5.00	7.00
9	2.00	3.00	2.00	0.0	3.00	7.00
10	5.00	6.00	5.00	3.00	0.0	4.00
11	9.00	9.00	7.00	7.00	4.00	0.0

МАТРИЦА УЗЛОВ, ВХОДЯЩИХ В КРАТЧАЙШИЕ ЦЕПИ

	1	2	3	4	5	6	7	8	9	10	11
1	1	2	3	4	2	3	4	4	3	3	4
2	1	2	1	4	5	5	5	5	5	5	5
3	1	1	3	4	4	6	4	4	6	6	4
4	1	2	3	4	5	7	7	7	7	7	7
5	2	2	4	4	5	7	7	8	7	7	8
6	3	7	3	7	7	6	7	7	9	10	10
7	4	5	4	4	5	6	7	8	6	6	8
8	7	5	7	7	5	7	7	8	9	9	11
9	6	6	6	6	6	6	6	8	9	10	10
10	6	6	6	6	6	6	6	9	9	10	11
11	8	8	8	8	8	10	8	8	10	10	11

граммы, описанной в разд. 2.4.3. Результаты решения приводятся на с. 76—77. Длины дуг, не содержащихся в сети, изображенной на рис. 2.19, были взяты равными 9999999,00.



Узлы	Пункты портового комплекса
1	Станция прибытия
3, 6, 10	Элеваторы
2, 5, 8,	Заводы производящие удобрения
4, 7, 9,	Другие пункты портового комплекса
11	Станция отправления

Рис. 2.19. Транспортная сеть.

2.5. ЗАДАЧИ О КРАТЧАЙШЕМ ПУТИ С ФИКСИРОВАННЫМИ ПЛАТЕЖАМИ

В предыдущих разделах были даны алгоритмы, которые могут быть использованы для определения цепи минимальной стоимости из заданного источника s в заданный сток t. Кроме того, алгоритм Дейкстры позволяет находить кратчайшую цепь из произвольного источника s в каждый другой узел сети. Эти алгоритмы, как и все другие алгоритмы поиска кратчайшей цепи, основаны на следующем утверждении: «Если кратчайшая цепь между узлами s и t проходит через узел k, то отрезок этой цепи, расположенный между s и k, образует кратчайшую цепь, соединяющую s и k. Кроме того, отрезок цепи, расположенный между k и t, также образует кратчайшую цепь между k и t».

Сейчас мы рассмотрим класс задач, для которых данное утверждение несправедливо. Такие задачи называются задачами о кратчайшем пути с фиксированными платежами, или просто *задачами с фиксированными платежами.*

Задачи с фиксированными платежами возникают в тех случаях, когда за прохождение через один или несколько заданных узлов сети взимается штраф или плата. В качестве примера можно рассмотреть задачу транспортировки товаров со складов в пункты потребления, когда за первое использование каждого промежуточного пункта распределительной сети взимается фиксированная плата. Другим примером является задача определения маршрута движения грузового судна, когда плата взимается за то, чтобы войти в некоторый порт. При этом пла-

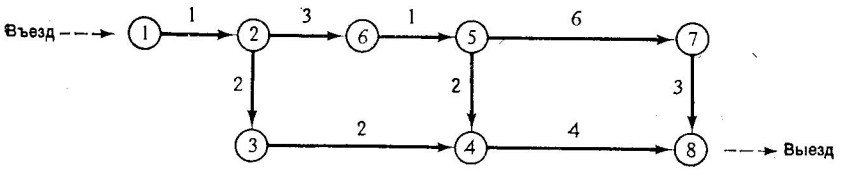


Рис. 2.20. Сеть со штрафами за выполнение поворота.

та в каждом порту может быть различна и зависит от размеров судна, вида перевозимого им груза и налога. Вообще данные задачи возникают в тех случаях, когда строятся, покупаются или сдаются на временное пользование сооружения, предназначенные для стоянки грузового транспорта.

Для решения сетевых задач с фиксированными платежами описанного выше класса был разработан ряд итеративных процедур, основанных на использовании множителей Лагранжа и (или) теории двойственности [23, 36, 37]. Хотя анализ подобного подхода выходит за рамки данной книги, хотелось бы рассмотреть один класс транспортных задач, которые могут быть просто решены с помощью методов, описанных в предыдущих разделах.

Во многих задачах транспортного планирования требуется определить оптимальный маршрут движения по сети улиц с односторонним и (или) двусторонним движением. При решении задач, связанных с транспортным потоком, обычно вводятся задержки при переезде через перекресток или «штрафы за выполнение поворота». Эти задержки и штрафы можно рассматривать как фиксированную плату. Штраф за выполнение поворота обычно зависит от направления движения при въезде на перекресток и направления движения при выезде с перекрестка. Предполагается, что за выполнение запрещенного поворота взимается бесконечно большая плата или штраф. Для иллюстрации численного метода решения данной задачи рассмотрим прямоугольную сеть, изображенную на рис. 2.20. Она

Таблица 2.7. Пути в задаче со штрафами за выполнение поворотов

Путь	Последовательность дуг пути	Затраты на прохождение по дуге	Штраф за выполнение поворота	Общие затраты	
P_1	(1, 2)	1	0	1	
	(2, 6)	3	0	3	
	(6, 5)	1	0	1	
	(5, 7)	6	0	6	
	(7, 8)	3	3	6	17
P_2	(1, 2)	1	0	1	
	(2, 6)	3	0	3	
	(6, 5)	1	0	1	
	(5, 4)	2	3	5	
	(4, 8)	4	3	7	17
P_3	(1, 2)	1	0	1	
	(2, 3)	2	3	5	
	(3, 4)	2	3	5	
	(4, 8)	4	0	4	15

представляет собой участок транспортной сети с перекрестками. Въезд на участок осуществляется через узел 1, а выезд — через узел 8. Числа, приписанные дугам, представляют собой плату за проезд по соответствующим улицам. Кроме того, предполагается, что за выполнение каждого поворота взимается штраф, равный 3. Требуется определить наиболее экономный путь из узла 1 в узел 8. Мы воспользуемся следующей теоремой.

Теорема [18]. В сети со штрафами за выполнение поворотов кратчайший путь из узла s в узел t, проходящий через промежуточный узел k, может не содержать в себе кратчайший путь из s в k или кратчайший путь из k в s.

Проиллюстрируем данную теорему на примере транспортной сети, изображенной на рис. 2.20. Перебирая всевозможные пути и выбирая среди них тот, для которого общие затраты минимальны, нетрудно получить наиболее экономный путь. Соответствующие результаты приведены в табл. 2.7. Из этих результатов видно, что наиболее экономным путем из узла 1 в узел 8 является путь P_3, общая стоимость которого равна 15. Отметим, что при движении от узла 1 к узлу 4 вдоль пути P_3 общие затраты составляют 11. В то же время, рассматривая пути P_1 и P_2, можно заключить, что между узлами 1 и 4 существует путь меньшей стоимости, равный 10. Он является частью пути P_2 и состоит из дуг (1, 2), (2, 6), (6, 5) и (5, 4). Поэтому данный путь является кратчайшим путем из узла 1 в узел 4, и в то же время он не принадлежит кратчайшему пути, соединяющему узлы 1 и 8. Следовательно, для решения за-

дачи о кратчайшем пути на сети со штрафами за выполнение поворотов нельзя непосредственно воспользоваться обычными алгоритмами поиска кратчайших путей. Однако исходная зада-

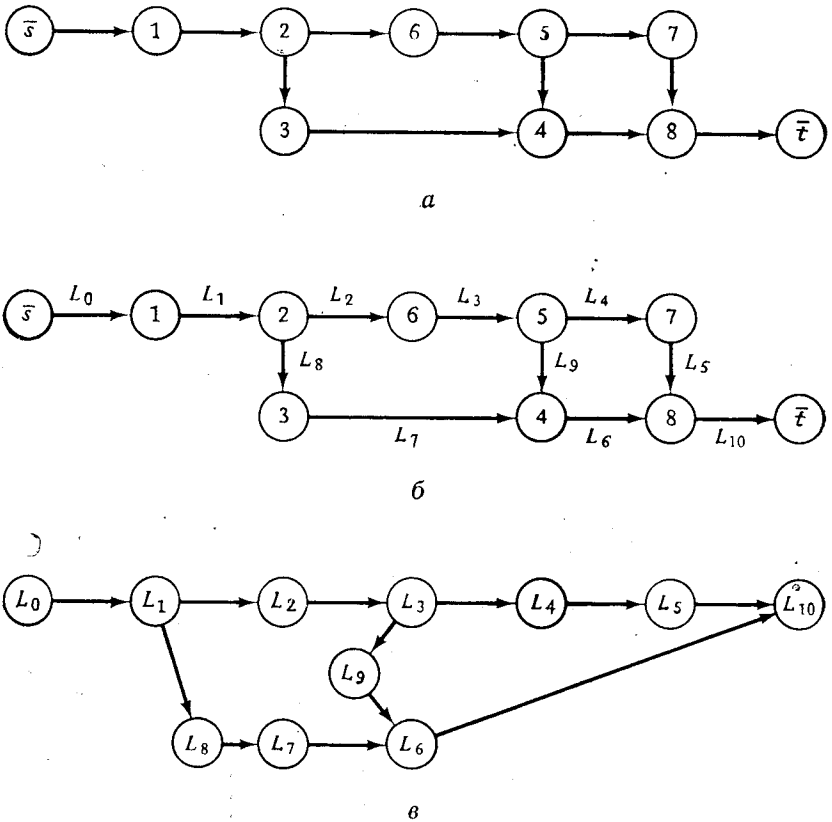


Рис. 2.21. Алгоритм поиска кратчайшего пути для сети со штрафами за выполнение поворота.
а — шаг 1; *б* — шаг 2; *в* — шаг 3.

ча может быть сведена к другой задаче, для которой указанные алгоритмы применимы [18]. Соответствующая процедура может быть описана следующим образом. Пусть задана сеть $G=(\mathbf{N}, \mathbf{A})$ с β узлами, образующими множество **N**, и α дугами, образующими множество **A**.

Шаг 1. Ввести фиктивный узел $\bar{s}$ и соединить его с источником s ориентированной дугой $(\bar{s}, s)$. Ввести фиктивный узел $\bar{t}$ и соединить его со стоком t ориентированной дугой $(t, \bar{t})$. Полу-

ченная при этом сеть для рассматриваемого примера изображена на рис. 2.21, *а*.

Шаг 2. Каждой дуге сети приписать фиктивную метку L_k. Пусть $L_0, L_1, \ldots, L_\alpha, L_{\alpha+1}$ — фиктивные метки $\alpha+2$ дуг расширенной сети. Полученная при этом сеть для рассматриваемого примера изображена на рис. 2.21, *б*.

Шаг 3. Построить фиктивную сеть, состоящую из $\alpha+2$ узлов $L_0, L_1, \ldots, L_{\alpha+1}$, причем узлы L_i и L_j соединены ориентированной дугой (L_i, L_j) в том случае, когда в помеченной сети, построенной на шаге 2, дуга с меткой L_i непосредственно предшествует дуге с меткой L_j. Значения параметра ветви, соединяющей узлы L_i и L_j, определяются равными $c(L_i)+p(L_i, L_j)$, где $c(L_i)$ — исходная стоимость дуги L_i, а $p(L_i, L_j)$ — штраф за выполнение поворота, соответствующего дуге (L_i, L_j). Стоимости дуг $(\bar{s}, s)$ и $(t, \bar{t})$ полагаются равными нулю. Предполагается также, что из узлов $\bar{s}$ и $\bar{t}$ повороты никогда не выполняются. Фиктивная сеть для рассматриваемого примера изображена на рис. 2.21, *в*. Стоимости дуг выписаны в табл. 2.8.

Таблица 2.8. Стоимости дуг фиктивной сети, построенной на шаге 3

Из узла	В узел	Стоимость
L_0	L_1	0
L_1	L_2	1
L_1	L_8	4
L_8	L_7	5
L_2	L_3	3
L_3	L_9	4
L_9	L_6	5
L_3	L_4	1
L_4	L_5	9
L_5	L_{10}	3
L_6	L_{10}	4
L_7	L_6	2

Используя один из алгоритмов поиска кратчайшего пути[1)], можно определить наиболее экономный путь из узла L_0 в узел L_{10}. Читателю предлагается проверить, что такой путь соответствует последовательности дуг (L_0, L_1), (L_1, L_8), (L_8, L_7), (L_7, L_6) и (L_6, L_{10}). Как следует из табл. 2.9 *а*, стоимость этого пути равна 15.

Записав данное решение в терминах исходной сетевой задачи, мы получим результаты, приведенные в табл. 2.9 *б*. Та-

[1)] Если не принимать во внимание ориентацию дуг, то алгоритмы, описанные в разд. 2.3 и 2.4, могут быть использованы для поиска кратчайших путей. — *Прим. перев.*

Таблица 2.9а. Оптимальный путь в фиктивной сети

Из узла	В узел	Стоимость
L_0	L_1	0
L_1	L_8	4
L_8	L_7	5
L_7	L_6	2
L_6	L_{10}	4

Таблица 2.9б. Оптимальное решение

Дуга	Узел исходной сети	Стоимость
(L_0, L_1)	1	—
(L_1, L_8)	2	1
(L_8, L_7)	3	5
(L_7, L_6)	4	5
(L_6, L_{10})	8	4

ким образом, путь минимальной стоимости в исходной сети соответствует последовательности узлов 1, 2, 3, 4, 8, а его общая стоимость равна 15.

2.6. ЗАДАЧА О *K* КРАТЧАЙШИХ ПУТЯХ

При решении некоторых практических задач, связанных с коммуникационными или транспортными сетями, требуется найти не один, а несколько кратчайших путей, расположенных в порядке возрастания их длин. Например, знание нескольких кратчайших путей в сети улиц позволит планировщикам транспортных сетей более точно моделировать потоки движения автомобилей по улицам городов. При составлении маршрута передачи информации по коммуникационной сети, когда некоторые пути временно заняты, могут быть использованы сведения о наилучших из возможных альтернативных вариантов.

Таким образом, в тех случаях, когда нет возможности воспользоваться наилучшим решением, можно найти ему замену из числа дополнительных решений. Кроме того, знание решений, ближайших к наилучшему, позволит определить устойчивость решений по отношению к внешним факторам, влияние которых не учитывается в сетевой модели.

В общей постановке задачи о K кратчайших путях не требуется, чтобы пути не содержали циклов, и учитываются только длины путей. Ослабление ограничений в данной задаче позволило разработать несколько процедур поиска решения, в ко-

торых учитывается специфика конкретной сетевой задачи. Традиционный подход к решению данной задачи был разработан рядом авторов и изложен в обзорных статьях, таких, например, как работа Дрейфуса [12]. В некоторых новых методах используется аналогия между решением общей задачи о K кратчайших путях и решением системы обыкновенных линейных уравнений. Один из таких методов, известный под названием «метод двойного поиска» [47], позволяет одновременно вычислять длины K кратчайших путей из начального узла ко всем остальным узлам сети. Перейдем к рассмотрению этого метода.

2.6.1. МЕТОД ДВОЙНОГО ПОИСКА

Рассмотрим ориентированную сеть, узлы которой занумерованы от 1 до n. Предположим, что для каждого узла имеется вектор оценок длин K кратчайших путей, соединяющих этот узел с заданным начальным узлом. Метод двойного поиска основан на последовательном уменьшении оценок, в результате чего за конечное число итераций строится оптимальный вектор оценок. Предполагается, что начальные оценки не меньше длин соответствующих путей.

На каждой итерации выполняются две процедуры. При выполнении процедуры прямого поиска узлы рассматриваются в порядке возрастания их номеров ($j=1, 2, \ldots, n$). После построения множества узлов $i<j$, соединенных дугой с узлом j, последовательно рассматриваются длины K кратчайших путей из начального узла в узел j и устанавливается возможность прохождения более коротких путей через эти узлы i. Если такие пути существуют, то их длины будут использоваться в качестве новых оценок на последующих итерациях. Аналогичная процедура выполняется при обратном поиске, однако в этом случае узлы рассматриваются в порядке убывания их номеров ($j=n, n-1, \ldots, 1$) и исследуются узлы $i>j$, соединенные дугой с узлом j.

В алгоритме двойного поиска используются две специальные алгебраические операции [47]. Они называются обобщенными операциями, поскольку выполняются не над числами, а над векторами. Данные векторы должны иметь одинаковую размерность и могут содержать как конечные, так и бесконечные компоненты. Однако требуется, чтобы конечные компоненты предшествовали бесконечным и были расположены в *строго* возрастающем порядке. Таким образом, векторы не должны содержать равных конечных компонент. Следующие векторы являются допустимыми: $\mathbf{A}=[-4, 0, 7, \infty]$, $\mathbf{B}=[3, 4, \infty, \infty]$. С другой стороны, векторы $\mathbf{C}=[-4, 3, 3, 9]$, $\mathbf{D}=[-9, 0, \infty, 9]$ не являются допустимыми.

Перейдем к рассмотрению этих двух операций. Первая из них называется *обобщенной операцией минимизации*, вторая— *обобщенной операцией сложения*. Эти операции являются двухместными, т. е. они могут выполняться только над двумя векторами.

При выполнении обобщенной операции минимизации вначале определяется множество всех компонент двух заданных векторов размерности k, а затем строится третий вектор такой же размерности, составленный из k различных наименьших по величине элементов этого множества, расположенных в строго возрастающем порядке. Если в новом векторе число конечных компонент меньше его размерности, то остальные компоненты принимаются равными ∞. В качестве примера рассмотрим векторы $\mathbf{A}=[-4, 0, 1, \infty]$ и $\mathbf{B}=[1, 7, 8, 9]$. Множеством, образованным из компонент векторов **A** и **B**, является множество $\{-4, 0, 1, \infty, 1, 7, 8, 9\}$. Различные по величине элементы этого множества могут быть упорядочены следующим образом: $-4, 0, 1, 7, 8, 9, \infty$. Поскольку размерность векторов **A** и **B** равна 4, результатом выполнения обобщенной операции минимизации над ними является вектор $\mathbf{C}=[-4, 0, 1, 7]$, составленный из четырех наименьших по величине элементов построенного множества.

При выполнении обобщенной операции сложения вначале определяется множество всевозможных попарных сумм компонент двух заданных векторов одинаковой размерности, а затем строится третий вектор такой же размерности, первой компонентой которого является минимальный элемент построенного множества. Остальные компоненты вектора выбираются точно так же, как было показано при описании обобщенной операции минимизации. Для иллюстрации выполнения обобщенной операции сложения рассмотрим векторы **A** и **B**. Множество всевозможных попарных сумм компонент этих двух векторов задается следующей таблицей:

		Элементы множества A			
		−4	0	1	∞
Элементы множества B	1	−3	1	2	∞
	7	3	7	8	∞
	8	4	8	9	∞
	9	5	9	10	∞

Поэтому множество всевозможных попарных сумм может быть определено как $\{-3, 1, 2, \infty, 3, 7, 8, \infty, 4, 8, 9, \infty, 5, 9, 10, \infty\}$. Элементы этого множества могут быть расположены в строго возрастающем порядке следующим образом: $-3, 1, 2,$

3, 4, 5, 7, 8, 9, 10, ∞. Поэтому обобщенная сумма векторов **A** и **B** равна вектору $\mathbf{C}=[-3, 1, 2, 3]$.

Для описания алгоритма двойного поиска нам потребуется формальное определение этих двух операций. Введем следующие обозначения: **R** — множество вещественных чисел; $\mathbf{R}_\infty$ — множество, содержащее все элементы из **R** и элемент ∞; K — число искомых длин путей; $\mathbf{S}(K)$ — множество векторов размерности K, компоненты которых принадлежат $\mathbf{R}_\infty$ и расположены в строго возрастающем порядке; $\min_j$ — операция нахождения j-го минимального элемента заданного множества. Поскольку по определению компоненты векторов из $\mathbf{S}(K)$ расположены в строго возрастающем порядке, то следует принять соглашение, что $\infty<\infty$ в том случае, когда вектор содержит несколько бесконечных компонент.

Обобщенная операция минимизации. Пусть **A**, **B** и **C** — векторы из множества $\mathbf{S}(K)$, т. е.

$$\mathbf{A}=[a_1, a_2,..., a_K], \quad a_1<a_2<...<a_K, \quad a_i\in\mathbf{R}_\infty,$$
$$\mathbf{B}=[b_1, b_2,..., b_K], \quad b_1<b_2<...<b_K, \quad b_i\in\mathbf{R}_\infty,$$
$$\mathbf{C}=[c_1, c_2,..., c_K], \quad c_1<c_2<...<c_K, \quad c_i\in\mathbf{R}_\infty.$$

Пусть $\mathbf{T}_+$ — множество всех различных компонент векторов **A** и **B**. Обобщенная операция минимизации $\oplus$ определяется соотношением $\mathbf{A}\oplus\mathbf{B}=\mathbf{C}$, где $c_j=\min\ [\mathbf{T}_+]$, $j=1, ..., K$.

Обобщенная операция сложения. Пусть **A**, **B** и **C** — векторы из множества $\mathbf{S}(K)$. Пусть $\mathbf{T}_\times$ — множество всевозможных попарных сумм компонент векторов **A** и **B**. Обобщенная операция сложения $\otimes$ определяется соотношением $\mathbf{A}\otimes\mathbf{B}=\mathbf{C}$, где $c_j=\min_j[\mathbf{T}_\times]$, $j=1,..., K$.

Описание алгоритма. Выберем из множества $\mathbf{S}(K)$ векторы $\mathbf{F}=[0, \infty, \infty, ..., \infty]$ и $\mathbf{V}=[\infty, \infty, ..., \infty]$. Тогда для любого вектора $\mathbf{A}\in\mathbf{S}(K)$ справедливы следующие соотношения:

$$\mathbf{A}\oplus\mathbf{V}=\mathbf{A}, \quad \mathbf{A}\otimes\mathbf{V}=\mathbf{V}, \quad \mathbf{A}\otimes\mathbf{F}=\mathbf{A}. \qquad (2.55), (2.56), (2.57)$$

Алгоритм двойного поиска, разработанный в [47], может быть описан следующим образом. Пусть узлы сети занумерованы от 1 до n, а длина каждой дуги (i, j) равна d_{ij}. Введем обозначения

$$\mathbf{D}_{ij}=[d_{ij}, \infty, \infty,..., \infty]\in\mathbf{S}(K), \quad \mathbf{D}=[\mathbf{D}_{ij}],$$
$$\mathbf{L}=[\mathbf{L}_{ij}], \quad \text{где } \mathbf{L}_{ij}=\mathbf{D}_{ij} \text{ при } i>j; \quad \mathbf{L}_{ij}=\mathbf{V} \text{ при } i\leqslant j,$$
$$\mathbf{U}=[\mathbf{U}_{ij}], \quad \text{где } \mathbf{U}_{ij}=\mathbf{D}_{ij} \text{ при } i<j, \quad \mathbf{U}_{ij}=\mathbf{V} \text{ при } i\geqslant j.$$

Здесь $\mathbf{D}_{ij}$, $\mathbf{L}_{ij}$ и $\mathbf{U}_{ij}$ — векторы из $\mathbf{S}(K)$, а **D**, **L** и **U** — матрицы, элементами которых являются векторы из $\mathbf{S}(K)$. Если

рассматриваемая сеть содержит более одной дуги, соединяющей узлы i и j, то определим $\mathbf{D}_{ij}$ как

$$\mathbf{D}_{ij}=[d^1_{ij}, d^2_{ij},..., d^t_{ij}, \infty, \infty,..., \infty] \in \mathbf{S}(K),$$

где d^1_{ij}, d^2_{ij}, ..., d^t_{ij} — длины дуг, соединяющих узлы i и j. Отметим, что если $t>K$, то вектор $\mathbf{D}_{ij}$ не содержит бесконечных компонент.

Пусть $\mathbf{E}_{0m}\in\mathbf{S}(K)$ — вектор, компонентами которого являются начальные оценки длин K кратчайших путей из источника в узел m. Предполагается, что если m — источник, то первая компонента вектора $\mathbf{E}_{0m}$ равна нулю. Определим массив $\mathbf{E}(0)$ векторов $\mathbf{E}_{0m}$ ($m=1, 2, ..., n$): $\mathbf{E}(0)=[\mathbf{E}_{01}, \mathbf{E}_{02}, ..., \mathbf{E}_{0n}]$. Отметим, что $\mathbf{E}(0)$ также является вектором, компоненты которого суть элементы множества $\mathbf{S}(K)$. На w-м шаге алгоритма двойного поиска строится вектор $\mathbf{E}(w)=[\mathbf{E}_{w1}, \mathbf{E}_{w2},..., \mathbf{E}_{wn}]$, где $\mathbf{E}_{wm}$ — элемент множества $\mathbf{S}(K)$, содержащий текущие оценки длин K кратчайших путей из источника в узел m. Построение векторов оценок выполняется с помощью следующих рекуррентных соотношений:

$$\mathbf{E}(2r+1)=\mathbf{E}(2r)\oplus\mathbf{E}(2r+1)\otimes\mathbf{L}, \tag{2.58}$$

$$\mathbf{E}(2r+2)=\mathbf{E}(2r+1)\oplus\mathbf{E}(2r+2)\otimes\mathbf{U}. \tag{2.59}$$

Данные операции выполняются поочередно для каждого $r=0, 1, 2,...$. Если на некотором шаге в результате последовательного выполнения операций (2.58) и (2.59) будут получены одинаковые векторы оценок, то найденное решение будет оптимальным. Операции, определяемые соотношениями (2.58), (2.59), называются операциями *обратного* и *прямого поиска* соответственно. В каждом из этих соотношений в первую очередь выполняется обобщенная операция сложения.

После того как будут найдены длины путей, выполняется процедура трассировки, позволяющая определить соответствующие пути. Остановимся на описании процедуры трассировки. Предположим, что требуется найти m-й кратчайший путь из источника в узел i. Пусть H_{mi} — длина этого пути и пусть узел j соединен дугой с узлом i. Тогда

$$H_{mi}=H_{tj}+d_{ji}, \tag{2.60}$$

где, как и раньше, d_{ji} — длина дуги (j, i), а H_{tj} $(t\leqslant m)$ — длина t-го кратчайшего пути из источника в узел j. Для каждого узла i процедура трассировки заключается в поиске узла j, для которого выполняется соотношение (2.60). Как только узел j будет найден, мы вновь обращаемся к данной процедуре и выполняем ее до тех пор, пока не достигнем источника.

Если требуется найти кратчайшие пути, не содержащие циклов, то описанную выше процедуру следует модифициро-

вать. По существу модифицирование заключается в следующем: если некоторый узел является «кандидатом» в узлы, принадлежащие рассматриваемому пути, то следует проверить, не был ли данный узел ранее получен с помощью соотношения (2.60).

2.6.2. ПРИМЕНЕНИЕ АЛГОРИТМА ДВОЙНОГО ПОИСКА К РЕШЕНИЮ МОДЕЛЬНОЙ ЗАДАЧИ

Рассмотрим сеть, изображенную на рис. 2.22. Для данной сети требуется найти длины трех кратчайших путей из узла 1 во все остальные узлы.

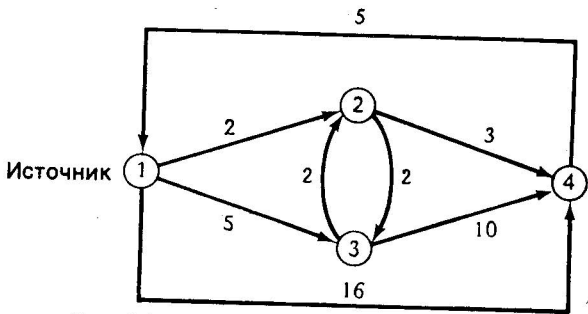


Рис. 2.22. Сеть для модельного примера.

В данном примере $K=3$, $n=4$. В качестве произвольных допустимых векторов начальных оценок выберем векторы $\mathbf{E}_{01}=[0, 21, 22]$, $\mathbf{E}_{02}=\mathbf{E}_{03}=\mathbf{E}_{04}=[20, 21, 22]$. Если выбор конечных оценок затруднен или невозможен, то все компоненты векторов можно положить равными ∞. Исключение составляет лишь оценка длины кратчайшего пути в источник, которая всегда полагается равной нулю.

Матрица расстояний $\mathbf{D}$, а также матрицы $\mathbf{L}$ и $\mathbf{U}$ содержат по 16 элементов, каждый из которых является вектором из множества $\mathbf{S}(3)$. Данные матрицы имеют следующий вид:

$$\mathbf{D}=[\mathbf{D}_{ij}]=\begin{bmatrix}\mathbf{D}_{11} & \mathbf{D}_{12} & \mathbf{D}_{13} & \mathbf{D}_{14}\\ \mathbf{D}_{21} & \mathbf{D}_{22} & \mathbf{D}_{23} & \mathbf{D}_{24}\\ \mathbf{D}_{31} & \mathbf{D}_{32} & \mathbf{D}_{33} & \mathbf{D}_{34}\\ \mathbf{D}_{41} & \mathbf{D}_{42} & \mathbf{D}_{43} & \mathbf{D}_{44}\end{bmatrix},$$

$$\mathbf{L}=[\mathbf{L}_{ij}]=\begin{bmatrix}\mathbf{V} & \mathbf{V} & \mathbf{V} & \mathbf{V}\\ \mathbf{D}_{21} & \mathbf{V} & \mathbf{V} & \mathbf{V}\\ \mathbf{D}_{31} & \mathbf{D}_{32} & \mathbf{V} & \mathbf{V}\\ \mathbf{D}_{41} & \mathbf{D}_{42} & \mathbf{D}_{43} & \mathbf{V}\end{bmatrix},$$

$$\mathbf{U} = [\mathbf{U}_{ij}] = \begin{bmatrix} \mathbf{V} & \mathbf{D}_{12} & \mathbf{D}_{13} & \mathbf{D}_{14} \\ \mathbf{V} & \mathbf{V} & \mathbf{D}_{23} & \mathbf{D}_{24} \\ \mathbf{V} & \mathbf{V} & \mathbf{V} & \mathbf{D}_{34} \\ \mathbf{V} & \mathbf{V} & \mathbf{V} & \mathbf{V} \end{bmatrix}$$

Для каждого вектора $\mathbf{D}_{ij}$ одна из его компонент равна d_{ij}, а $K-1=2$ компоненты равны ∞. Массив $[d_{ij}]$ вычисляется непосредственно из сети:

$$[d_{ij}] = \begin{bmatrix} 0 & 2 & 5 & 16 \\ \infty & 0 & 2 & 3 \\ \infty & 2 & 0 & 10 \\ 5 & \infty & \infty & 0 \end{bmatrix}.$$

Матрицы **D**, **L** и **U** задаются следующим образом:

D

0	2	5	16
∞	∞	∞	∞
∞	∞	∞	∞
∞	0	2	3
∞	∞	∞	∞
∞	∞	∞	∞
∞	2	0	10
∞	∞	∞	∞
∞	∞	∞	∞
5	∞	∞	0
∞	∞	∞	∞
∞	∞	∞	∞

L

∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	2	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
5	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞

U

∞	2	5	16
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	2	3
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	10
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞

В качестве примера рассмотрим случай $r=0$. В результате выполнения операции обратного поиска образуется вектор оценок

$$\mathbf{E}(1) = \mathbf{E}(0) \oplus \mathbf{E}(1) \otimes \mathbf{L}^*.$$

Обобщенная сумма $\mathbf{E}(1) \otimes \mathbf{L}$ равна

$$\begin{bmatrix} 25 & 22 & \infty & \infty \\ 26 & 23 & \infty & \infty \\ 27 & 24 & \infty & \infty \end{bmatrix}.$$

После выполнения операции обратного поиска получаем, что $\mathbf{E}(1) = \mathbf{E}(0)$. Остановимся более подробно на вычислении век-

тора $\mathbf{E}(1)$. Выполнение обобщенной операции сложения аналогично умножению справа вектора на матрицу с той лишь разницей, что элементы произведения вычисляются в обратном направлении. Однако в данном случае элементами массивов яв-

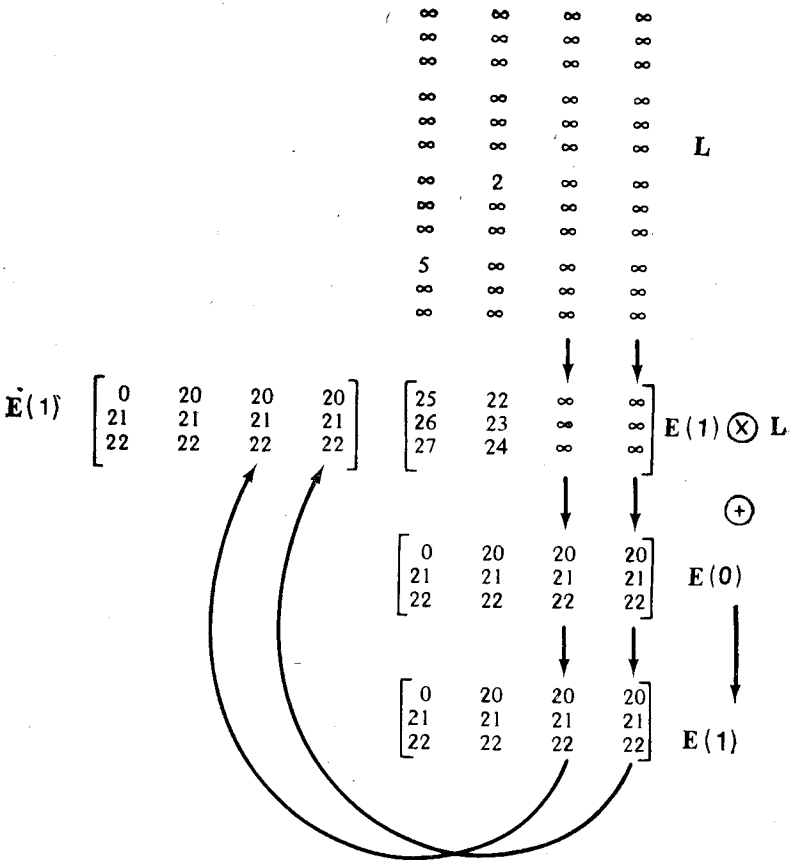


Рис. 2.23. Выполнение процедуры обратного поиска в модельном примере.

ляются не элементы множества $\mathbf{R}_\infty$, а K-мерные векторы из $\mathbf{S}(K)$. Кроме того, вместо обычных операций $+$ и $\times$ выполняются операции $\oplus$ и $\otimes$ соответственно. После того как вычислен некоторый столбец матрицы $\mathbf{E}(1) \otimes \mathbf{L}$, при помощи обобщенной операции минимизации производится его сравнение с соответствующим элементом вектора $\mathbf{E}(0)$, в результате чего определяется соответствующий элемент вектора $\mathbf{E}(1)$. На рис. 2.23 выписаны массивы $\mathbf{E}(0)$, $\mathbf{L}$, $\mathbf{E}(1) \otimes \mathbf{L}$ и показано, как с

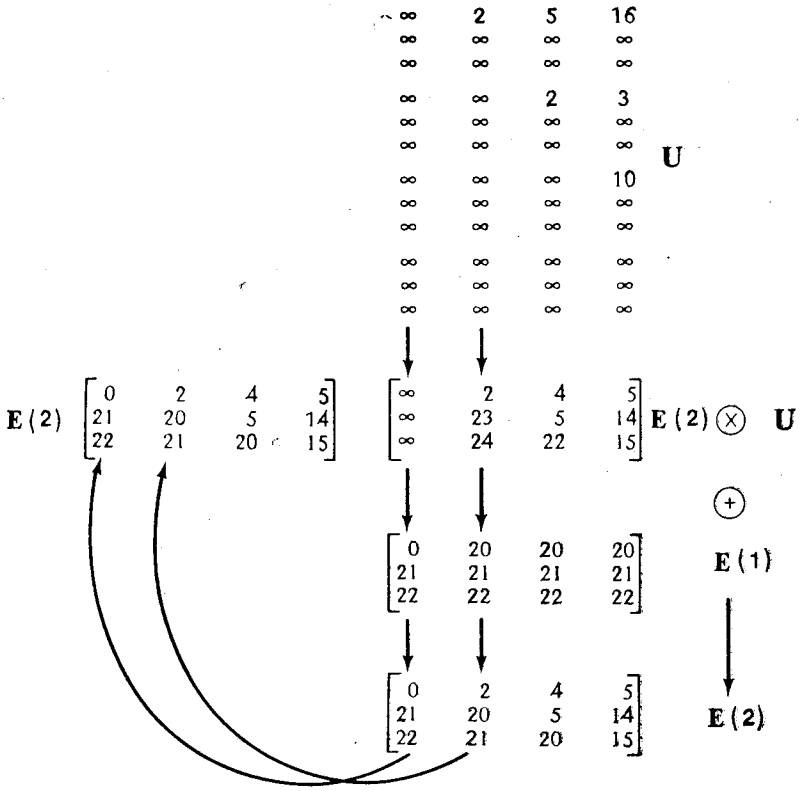


Рис. 2.24. Выполнение процедуры прямого поиска в модельном примере.

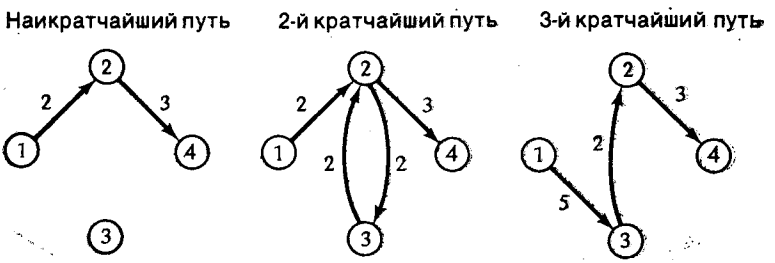


Рис. 2.25. Пути в модельном примере.

помощью процедуры обратного поиска вычисляются элементы вектора $\mathbf{E}(1)$[1].

Затем выполняется первый шаг процедуры прямого поиска. Данная операция записывается в виде

$$\mathbf{E}(2) = \mathbf{E}(1) \oplus \mathbf{E}(2) \otimes \mathbf{U}.$$

Обобщенная сумма $\mathbf{E}(2) \otimes \mathbf{U}$ является следующим вектором, компоненты которого принадлежат множеству $\mathbf{S}(3)$:

$$\begin{bmatrix} \infty & 2 & 4 & 5 \\ \infty & 23 & 5 & 14 \\ \infty & 24 & 22 & 15 \end{bmatrix}.$$

Новый вектор оценок, $\mathbf{E}(2)$, задается теперь матрицей

$$\begin{bmatrix} 0 & 2 & 4 & 5 \\ 21 & 20 & 5 & 14 \\ 22 & 21 & 20 & 15 \end{bmatrix}.$$

Далее, используя вектор $\mathbf{E}(2) \otimes \mathbf{U}$, выполняем последовательность процедур, аналогичных тем, которые были описаны при вычислении обобщенной суммы $\mathbf{E}(1) \otimes \mathbf{L}$. При этом вычисления элементов производятся в прямом направлении, как показано на рис. 2.24.

Продолжая аналогичные вычисления, мы получим результаты, приведенные в табл. 2.10. При $r=2$ в результате выполнения процедур обратного и прямого поиска мы получаем одно и то же решение. Как отмечалось выше, это означает, что данное решение является оптимальным.

С помощью процедуры трассировки, описанной рекуррентным соотношением (2.60), находятся наикратчайший и второй и третий кратчайшие пути из узла 1 в узел 4 (рис. 2.25). Отметим, что второй кратчайший путь содержит цикл.

2.6.3. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ ДВОЙНОГО ПОИСКА

Назначение: определение длины кратчайшего пути из начального узла во все остальные узлы сети.

Локализация: подпрограмма KSHORT в пакете сетевой оптимизации.

[1] При вычислении $\mathbf{E}(1)$ по формуле (*) вначале определяется вектор $\mathbf{E}_{1n}$. Это возможно в силу того, что n-й столбец матрицы $\mathbf{L}$ состоит из векторов, все компоненты которого равны ∞. Далее по соотношению (*) определяется вектор $\mathbf{E}_{1(n-1)}$. Это возможно в силу того, что $(n-1)$-й столбец матрицы $\mathbf{L}$ состоит из векторов, среди которых только у последнего могут быть компоненты, не равные ∞, и поэтому для вычисления $\mathbf{E}_{1(n-1)}$ необходим только вектор $\mathbf{E}_{1n}$. Далее по соотношению (*) определяются векторы $\mathbf{E}_{1(n-2)}$, $\mathbf{E}_{1(n-3)}$, ..., $\mathbf{E}_{11}$. — *Прим. перев.*

Таблица 2.10. Результаты работы алгоритма двойного поиска для модельной задачи

r	Тип поиска	Вектор оценок
0	Обратный	$\begin{bmatrix} 0 & 20 & 20 & 20 \\ 21 & 21 & 21 & 21 \\ 22 & 22 & 22 & 22 \end{bmatrix}$
0	Прямой	$\begin{bmatrix} 0 & 2 & 4 & 5 \\ 21 & 20 & 5 & 14 \\ 22 & 21 & 20 & 15 \end{bmatrix}$
1	Обратный	$\begin{bmatrix} 0 & 2 & 4 & 5 \\ 10 & 6 & 5 & 14 \\ 19 & 7 & 20 & 15 \end{bmatrix}$
1	Прямой	$\begin{bmatrix} 0 & 2 & 4 & 5 \\ 10 & 6 & 5 & 9 \\ 19 & 7 & 8 & 10 \end{bmatrix}$
2	Обратный	$\begin{bmatrix} 0 & 2 & 4 & 5 \\ 10 & 6 & 5 & 9 \\ 14 & 7 & 8 & 10 \end{bmatrix}$
2	Прямой	$\begin{bmatrix} 0 & 2 & 4 & 5 \\ 10 & 6 & 5 & 9 \\ 14 & 7 & 8 & 10 \end{bmatrix}$

Ограничения: программа обрабатывает сети, содержащие до 50 узлов и 50 дуг. Размеры сети можно увеличить, изменив границы массивов в операторах размерности, записанных в подпрограмме KSHORT и в основной программе. Одновременно могут решаться одна или несколько задач.

Входные данные:

Набор 1. Одна карта с именем алгоритма KSRT в формате (A4).

Набор 2. Одна карта с числом узлов и числом дуг в сети в формате (2I10).

Набор 3. Данный набор состоит из следующих трех разделов.

Раздел 1:

1) номер начального узла дуги; 2) номер конечного узла дуги; 3) длина дуги.

Формат (3I10).

Для каждой дуги соответствующие величины должны быть пробиты на одной карте. Карты располагаются в порядке возрастания конечных узлов дуг.

Раздел 2:

1) K, NS, IMAX.

Формат (4X, 3I5).

Раздел 3:
1) NF, PMAX.
Формат (4X, 2I5).

Замечание: если данную задачу требуется последовательно решать для различных начальных и (или) конечных узлов сети, то входные данные следует изменять только в разделах 2 и 3.

Набор 4. Данный набор состоит из одной карты, содержащей слово 'PEND' в формате (A4), которая указывает конец задачи.

Набор 5. Данный набор состоит из одной карты, содержащей слово 'EXIT' в формате (A4), которая указывает конец входных данных.

Составные части программы. Программа состоит из следующих подпрограмм: KSHORT (ввод данных и вычисления); DSWP (печать длин K кратчайших путей из начального узла сети во все остальные узлы); TRACE (печать узлов K кратчайших путей из начального узла сети в заданный конечный узел).

Используемые переменные и массивы:

N — число узлов в сети,

MU — число дуг (I, J), для которых I<J,

ML — число дуг (I, J), для которых J<I,

LLEN — массив, в котором значение J-го элемента равно числу дуг (I, J) при J<I,

LINC — массив номеров I узлов, соединенных дугой с узлом J, где I>J. Номера I располагаются последовательно для J=1, 2, ..., N,

LVAL — массив длин дуг, соответствующих узлам в массиве LINC,

ULEN — массив, в котором значение J-го элемента равно числу дуг (I, J) при I<J.

UINC — массив номеров I узлов, соединенных дугой с узлом J, где I<J. Номера I располагаются последовательно для J=1, 2, ..., N,

UVAL — массив длин дуг, соответствующих узлам в массиве UINC,

INF — верхняя граница длин всех путей сети,

START — массив, в котором значение J-го элемента равно номеру позиции, начиная с которой в массиве INC располагаются номера I узлов, соединенных дугой с узлом J,

INC — массив номеров I узлов, соединенных дугой с узлом J. Номера I располагаются последовательно для J=1, 2, ..., N,

VAL — массив длин дуг, соответствующих узлам в массиве INC.

Дополнительные переменные, которые должны быть описаны пользователем:

K — число искомых кратчайших путей,

IMAX — максимально допустимое число итераций в алгоритме двойного поиска,

NS, NF — начальный и конечный узлы K искомых кратчайших путей,

PMAX — максимальное число путей из узла NS в узел NF, которое должно быть сгенерировано.

Используемый метод: данный алгоритм основан на методе, описанном в разд. 2.6.1.

Литература: [48].

Если по условию задачи требуется рассматривать только пути, не содержащие циклов, то подпрограмма TRACE может быть модифицирована таким образом, что поиск пути завершится, как только встретится цикл. При этом в подпрограмме вместо операторов, расположенных в строках 52—54, следует записать следующие операторы:

```
      DO 60 J = 1, K
      IF(X(NI, J) — LT) 60, 180, 70
180   DO 181 IA = 1, KK
      IF(NI — P(IA)) 181, 70, 181
181   CONTINUE
      GO TO 80
 60   CONTINUE
```

2.6.4. РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ

Эффективность метода двойного поиска наглядно показана в работе [48]. Кроме того, в этой работе показано, что данный метод как теоретически, так и практически является более эффективным, чем другие методы решения задачи о K кратчайших путях.

В настоящем разделе мы будем рассматривать прямоугольные сети, какими являются, например, сети проселочных дорог во многих сельскохозяйственных районах Соединенных Штатов. Интересной особенностью данных сетей является наличие в них нескольких путей заданной длины. Если требуется найти кратчайший путь, удовлетворяющий одному или нескольким внешним условиям, таким, например, как непревышение допустимой нагрузки на мосты, через которые проходит этот путь, то приходится искать все пути заданной длины. В настоящем разделе мы остановимся на некоторых результатах вычислений, связанных как с определением длин путей, так и с выполнением процедуры трассировки.

Будем рассматривать сеть, состоящую из 56 узлов и 182 дуг (рис. 2.26). Через NS и NF будем обозначать соответственно

номера источников и стоков. Ниже приводятся данные о: 1) числе построенных путей; 2) порядковом номере (по длине) максимального из построенных путей; 3) общем времени решения задачи; 4) времени выполнения процедуры трассировки.

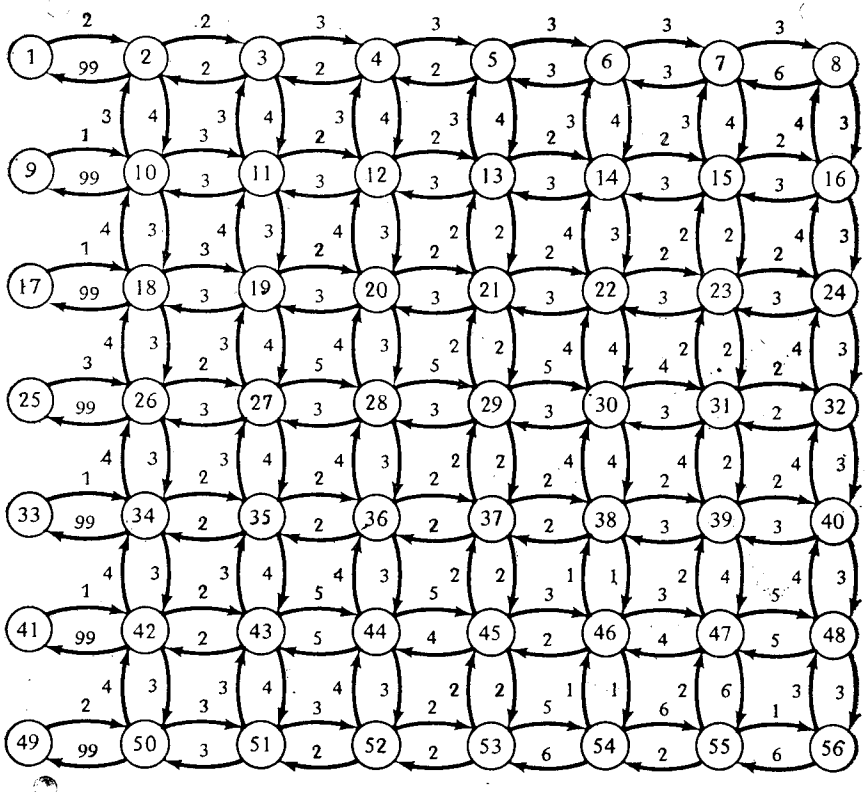


Рис. 2.26. Пример поиска K кратчайших путей.

Рассматриваются два случая. В случае 1) NS=1, NF=56, K=10; в случае 2) NS=56, NF принимает 7 различных значений (каждый пользователь, для которого находятся пути между NS и NF, задает свои значения NF), K=5, а максимальное число построенных путей между NS и NF равно 30. Результаты вычислений приводятся в табл. 2.11а и табл. 2.11б.

В табл. 2.11а приводятся следующие результаты. В первом столбце записаны значения 10, 20, 40, 80, 160, 250 и 285, которые принимал параметр PMAX, соответствующий максимальному числу построенных путей. В каждом из этих случаев были получены оценки длин K=10 путей. Порядковый номер (по

Таблица 2.11а. Результаты вычислений для случая 1 (IBM 360)

Число построенных путей	Порядковый номер максимального из построенных путей	Общее время выполнения (с)	Время выполнения процедуры трассировки (с)
10	3	4,62	0,80
20	3	4,83	1,01
40	4	4,88	1,06
80	5	5,80	1,98
160	7	7,29	3,47
250	7	9,22	5,40
285	8	10,00	6,15

Таблица 2.11б. Результаты вычислений для случая 2

Варианты	Номер пользователя	Узел NF	Число построенных путей	Порядк. номер макс. из построенных путей, не содержащих циклов	IBM 360	CYBER 175	
					Время выполнения	Общее время выполнения (с)	Время выполнения процедуры трассировки (с)
I	1	4	20	5			
	2	8	17	4			
	3	19	30	5			
	4	29	10	5	3,77	0,271	0,108
	5	33	9	5			
	6	39	8	4			
	7	52	6	4			
II	1	5	10	5			
	2	23	12	5			
	3	11	21	5			
	4	34	9	5	3,89	0,281	0,118
	5	40	5	5			
	6	9	30	5			
	7	50	7	4			
III	1	1	27	5			
	2	9	30	5			
	3	17	30	5			
	4	25	28	5	4,62	0,367	0,204
	5	33	9	5			
	6	41	30	5			
	7	49	7	5			

длине) PMAX-го пути записан во втором столбце. В третьем и четвертом столбцах указывается общее время решения задачи и время работы подпрограммы TRACE соответственно. Разность между этими двумя величинами равна времени работы программы, реализующей алгоритм двойного поиска.

В табл. 2.11б приводятся результаты трех просчетов задачи на машинах IBM 360 и CYBER 175. В первом столбце ука-

заны номера вариантов. Во втором столбце — номера пользователей системы. Третий столбец содержит номера NF конечных узлов сети, задаваемые каждым пользователем. В четвертом столбце указано число построенных путей из узла NS=56 в узел NF. В рассматриваемом случае это число не должно превосходить величины PMAX=30. В пятом столбце записывается порядковый номер (по длине) максимального из построенных путей, не содержащих циклов. Этот номер не должен превосходить $K=5$. В шестом и седьмом столбцах указано общее время решения задачи на IBM 360 и CYBER 175 соответственно, а в восьмом столбце — время выполнения подпрограммы TRACE для CYBER 175.

2.6.5. РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ ДЛЯ ЗАДАЧИ НАХОЖДЕНИЯ ЧЕТЫРЕХ КРАТЧАЙШИХ ПУТЕЙ

Требуется найти четыре кратчайших пути из узла 1 в каждый из узлов 10, 11 и 12 в сети, изображенной на рис. 2.27. При работе программы начальные оценки полагаются равными ∞. Входные данные должны располагаться в последова-

Таблица 2.12. Входные данные в модельном примере

Набор 1: KSRT
Набор 2: 12 31
Набор 3:

Номер карты				Номер карты				Номер карты			
1	2	1	4	2	3	1	4	3	1	2	4
4	3	2	5	5	4	2	2	6	5	2	3
7	1	3	4	8	2	3	5	9	5	3	2
10	6	3	2	11	2	4	2	12	5	4	10
13	7	4	5	14	2	5	3	15	3	5	2
16	4	5	10	17	6	5	5	18	8	5	4
19	3	6	2	20	5	6	5	21	9	6	3
22	4	7	5	23	8	7	8	24	5	8	4
25	7	8	8	26	9	8	12	27	6	9	3
28	8	9	12	29	7	10	2	30	8	11	2
31	9	12	2	32	4	1	30	33	11	10	
34	10	10		35	12	10					

Набор 4: PEND
Набор 5: EXIT

тельности, указанной в табл. 2.12. Отметим, что номера конечных узлов, набитые на перфокартах 1—31, расположены в *строго возрастающем порядке*. Приводимые ниже выходные

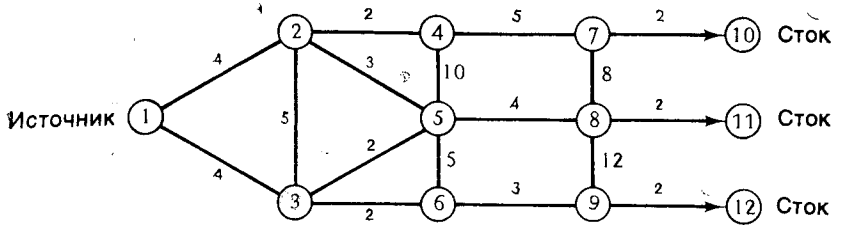


Рис. 2.27. Сеть в задаче о четырех кратчайших путях.

данные программы содержат длины четырех кратчайших путей.

ДЛИНА К = 4 КРАТЧАЙШИХ ПУТЕЙ ИЗ УЗЛА 1
В УЗЕЛ 11

1	0	8	12	13
2	4	8	9	10
3	4	8	9	12
4	6	10	11	12
5	6	7	10	11
6	6	10	11	12
7	11	15	16	17
8	10	11	14	15
9	9	13	14	15
10	13	17	18	19
11	12	13	16	17
12	11	15	16	17

ЧИСЛО ИТЕРАЦИЙ, В РЕЗУЛЬТАТЕ ВЫПОЛНЕНИЯ КОТОРЫХ ПРОЦЕСС СХОДИТСЯ, РАВНО 5

К КРАТЧАЙШИХ ПУТЕЙ ИЗ УЗЛА 1 В УЗЕЛ 11

ПУТЬ	ДЛИНА	ПОСЛЕДОВАТЕЛЬНОСТЬ УЗЛОВ						
1	12	11	8	5	3	1		
2	13	11	8	5	2	1		
3	16	11	8	5	3	5	3	1
4	16	11	8	5	3	6	3	1
5	17	11	8	5	2	4	2	1
6	17	11	8	5	3	2	1	
7	17	11	8	5	3	5	2	1
8	17	11	8	5	6	3	1	
9	17	11	8	7	4	2	1	

К КРАТЧАЙШИХ ПУТЕЙ ИЗ УЗЛА 1 В УЗЕЛ 10

ПУТЬ	ДЛИНА	ПОСЛЕДОВАТЕЛЬНОСТЬ УЗЛОВ						
1	13	10	7	4	2	1		
2	17	10	7	4	2	4	2	1
3	18	10	7	4	2	3	1	
4	18	10	7	4	2	5	3	1
5	19	10	7	4	2	5	2	1

К КРАТЧАЙШИХ ПУТЕЙ ИЗ УЗЛА 1 В УЗЕЛ 12

ПУТЬ	ДЛИНА	ПОСЛЕДОВАТЕЛЬНОСТЬ УЗЛОВ						
1	11	12	9	6	3	1		
2	15	12	9	6	3	5	3	1
3	15	12	9	6	3	6	3	1
4	16	12	9	6	3	2	1	
5	16	12	9	6	3	5	2	1
6	16	12	9	6	5	3	1	
7	17	12	9	6	5	2	1	
8	17	12	9	6	9	6	3	1

2.7. АНАЛИЗ АЛГОРИТМОВ ПОИСКА КРАТЧАЙШИХ ПУТЕЙ И ОЦЕНКА ИХ СЛОЖНОСТИ

Проведение анализа вычислительных алгоритмов имеет как практическое, так и теоретическое значение. Для практических целей необходимо иметь информацию об оценках, или верхних границах, объема машинной памяти, требуемой для работы программы, реализующей алгоритм, и времени ее прохождения. Основное теоретическое значение анализа алгоритмов заключается, по-видимому, в получении количественных критериев, позволяющих проводить сравнения нескольких алгоритмов решения одной и той же задачи.

В настоящем разделе будут получены верхние границы объема вычислений, необходимых для реализации алгоритмов Дейкстры и Флойда и метода двойного поиска. В алгоритмах Дейкстры и Флойда выполняется только два типа элементарных операций — сложение и сравнение. Обычно предполагается, что время выполнения каждой из этих двух операций приблизительно одинаковое. Будем также предполагать, что в наихудшем случае при работе алгоритма выполняется максимально возможное число элементарных операций. Это число, которое принимается за верхнюю границу объема вычислений и называется вычислительной сложностью алгоритма, является функцией размера сети и количества искомых решений (путей). Вычислительная сложность метода двойного поиска будет выражена как функция числа обобщенных операций, определенных в разд. 2.6.1.

2.7.1. ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ МЕТОДА ДЕЙКСТРЫ

Рассмотрим сеть $G=(\mathbf{N}, \mathbf{A})$, содержащую n узлов, образующих множество $\mathbf{N}$. В наихудшем случае конечный узел сети будет n-м по счету узлом, которому приписывается постоянная пометка. Предположим, что в некоторый заданный момент работы алгоритма m узлам приписаны постоянные пометки, а $n-m$ узлам — временные пометки. Для определения $(m+1)$-го узла, которому должна быть приписана постоянная пометка, необходимо вычислить новые величины $n-m$ временных пометок, выполняя при этом для каждой из них операцию сложения и операцию сравнения. После вычисления новых значений временных пометок необходимо также найти минимальное среди них, чтобы определить пометку, которая должна стать постоянной. Данная процедура минимизации состоит из $n-m-1$ операций сравнения. Таким образом, если имеется m узлов с постоянными пометками, то число элементарных операций, которое необходимо выполнить для того, чтобы еще одному узлу приписать постоянную пометку, равно $3(n-m)-1 \approx 3(n-m)$.

Общее число элементарных операций, выполнение которых необходимо для завершения работы алгоритма, в наихудшем случае равно

$$\sum_{m=1}^{n} 3(n-m) = 3\sum_{m=1}^{n}(n-m) = 3\sum_{m=1}^{n-1}(n-m) =$$

$$= 3[(n-1)+(n-2)+\ldots+1] = 3n(n-1)/2.$$

2.7.2. ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ АЛГОРИТМА ФЛОЙДА

Рассмотрим сеть $G=(\mathbf{N}, \mathbf{A})$, где $\mathbf{N}=\{1, 2, \ldots, n\}$. На каждой итерации алгоритма суммарное число элементов, значение которых должно быть оценено с помощью трехместной операции (2.53), можно вычислить, исходя из следующих соображений:

1. Общее число элементов матрицы равно n^2.
2. Суммарное число элементов в базовой строке и базовом столбце равно $2n-1$.
3. Число нулевых элементов на главной диагонали равно n и для каждого из них не надо получать новую оценку.
4. Один элемент базовой строки и один элемент базового столбца расположены на главной диагонали.
5. Анализ каждого элемента требует выполнения одной операции сложения и одной операции сравнения, соответствующих трехместной операции.
6. Таким образом, максимальное число операций, выполняемых на одной итерации алгоритма, приблизительно равно $2n(n-3)$.

Поскольку число итераций равно числу узлов, т. е. n, то общее число элементарных операций, выполнение которых необходимо для завершения работы алгоритма, в наихудшем случае равно $2n^2(n-3)$.

2.7.3. ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ МЕТОДА ДВОЙНОГО ПОИСКА

Рассмотрим сеть $G=(\mathbf{N}, \mathbf{A})$, где $\mathbf{N}=\{1, 2, ..., n\}$. Процедуру определения числа элементарных операций в алгоритме рассмотрим на примере обратного поиска. Как отмечалось в разд. 2.6.1, при обратном поиске узлы рассматриваются в последовательности n, $n-1$, ..., 1. Число обобщенных операций сложения $\otimes$ и минимизации $\oplus$, необходимых для выполнения процедуры обратного поиска, приводится в табл. 2.13.

Таблица 2.13. Число обобщенных операций, неоходимых для выполнения процедуры обратного поиска

Узел	Число обобщенных операций сложения $\otimes$	Число обобщенных операций минимизации $\oplus$
n	0	0
$n-1$	1	1
$n-2$	2	2
$n-3$	3	3
.	.	.
.	.	.
.	.	.
1	$n-1$	$n-1$
Общее число операций	$\frac{n}{2}(n-1)$	$\frac{n}{2}(n-1)$

Таким образом, в соответствии с результатами, содержащимися в табл. 2.13, число обобщенных операций, выполняемых при обратном поиске, равно $n(n-1)$. Аналогичным образом можно показать, что число обобщенных операций, выполняемых при прямом поиске, также равно $n(n-1)$. Следовательно, общее число обобщенных операций в алгоритме двойного поиска равно $2n(n-1)$.

Если требуется найти K кратчайших путей, то общее число оценок, которое необходимо получить при выполнении одной процедуры поиска, равно nK. Поскольку при каждом поиске улучшается по крайней мере одна оценка, то число обращений к процедуре двойного поиска в наихудшем случае равно $nK/2$. Таким образом, общее число обобщенных операций, выполнение которых необходимо для завершения работы алгоритма двойного поиска, равно $(nK/2)2n(n-1)=Kn^2(n-1)\approx Kn^3$.

2.8. ЗАДАЧА О КРАТЧАЙШЕМ ОСТОВНОМ ДЕРЕВЕ

Перед тем как перейти к изучению этой новой потоковой задачи, остановимся на определениях, данных в гл. 1, которые имеют непосредственное отношение к нахождению кратчайшего остова. В настоящем разделе рассматриваются неориентированные сети.

Назовем *деревом* связное множество неориентированных ребер (дуг), не содержащее циклов. Таким образом, если задано множество m узлов, соединенных неориентированными ребрами, то для построения дерева необходимо выделить подмножество, состоящее из m—1 дуги. Иными словами, каждый узел соединен с другим узлом одним-единственным путем.

Рассмотрим сеть, содержащую n узлов, совокупность которых образует множество **S**. *Остовным деревом* называется связное множество, состоящее из (n—1) дуг (ребер) и n узлов. Из любого собственного подмножества множества **S** может быть образовано дерево, которое, однако, не является может не быть остовным деревом исходной сети. Как и раньше, будем предполагать, что каждой дуге, соединяющей узлы i и j из множества **S**, приписано число c_{ij}, называемое расстоянием, или весом, дуги. Теперь мы можем ввести понятие кратчайшего (или максимального) остова. *Кратчайшим* остовом называется такой остов сети, у которого сумма весов c_{ij} всех его дуг минимальная.

Задача о кратчайшем остове имеет широкое практическое применение. Предположим, например, что вы являетесь поставщиком природного газа и хотите поставлять его с места разработки K заказчикам. Тогда кратчайший остов сети подачи газа определит такую распределительную систему, которая свяжет всех заказчиков и при этом затраты (или расстояние) будут минимальными. Аналогичная задача возникает также при проектировании транспортной сети, когда требуется найти решения, минимизирующие затраты.

Построение кратчайшего остова часто встречается в задачах субоптимизации, или декомпозиции, сложных сетевых алгоритмов. Помимо того что данная задача имеет большое практическое значение, метод ее решения является уникальным в исследовании операций.

2.8.1. АЛГОРИТМ ПОСТРОЕНИЯ КРАТЧАЙШЕГО ОСТОВНОГО ДЕРЕВА

В рассмотренных выше потоковых алгоритмах были использованы рекуррентные схемы вычислений, позволяющие строить последовательность решений, сходящуюся к оптимальному решению. Оказывается, задача о кратчайшем остове является одной из немногих задач исследования операций, которые могут

быть решены с помощью «поглощающих» алгоритмов, являющихся весьма экономичными. Используя схему «поглощения», мы приведем следующий алгоритм.

Как отмечалось выше, задача о кратчайшем остове заключается в выборе таких дуг заданной сети, что их суммарная стоимость минимальна и для любой пары узлов найдется путь (или маршрут), соединяющий их. Этого можно достигнуть, выбирая дуги таким образом, что образованное ими дерево (определение которого было дано выше) покроет, или соединит, все узлы заданной сети. Иначе говоря, нас интересует, как построить остов минимальной стоимости.

Задача о кратчайшем остове решается достаточно просто. Алгоритм начинает работу с выбора произвольного узла сети и кратчайшей дуги из множества дуг, соединяющих этот узел с другими узлами. Соединим два узла выбранной дугой. Выберем ближайший к этим узлам третий узел. Добавляем этот узел и соответствующую дугу к сети. Продолжаем данный процесс до тех пор, пока все узлы не будут соединены между собой. Алгоритм, основанный на «поглощении» кратчайших дуг, может быть описан следующим образом.

Алгоритм построения кратчайшего остова

Шаг 1. Используя узлы исходной сети, определить следующие два множества: **S** — множество соединенных узлов; $\overline{\mathbf{S}}$ — множество несоединенных узлов. Вначале *все узлы* будут принадлежать множеству $\overline{\mathbf{S}}$.

Шаг 2. Выбрать *произвольный* узел из $\overline{\mathbf{S}}$ и соединить его с ближайшим соседним узлом. (После выполнения данного шага множество **S** будет содержать два узла.)

Шаг 3. Среди всех дуг, соединяющих узлы из множества **S** с узлами из множества $\overline{\mathbf{S}}$, выбрать кратчайшую дугу. Концевой узел этой дуги, лежащий в $\overline{\mathbf{S}}$, обозначить через δ. Удалить узел δ из множества $\overline{\mathbf{S}}$ и поместить его в множество S.

Шаг 4. Выполнять шаг 3 до тех пор, пока *все* узлы не будут принадлежать множеству **S**.

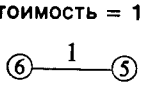
2.8.2. ПРИМЕР ПОИСКА РЕШЕНИЯ С ПОМОЩЬЮ «ПОЕДАЮЩЕГО» АЛГОРИТМА (РИС. 2.28)

Шаг 1. $\overline{\mathbf{S}}$ = (1, 2, 3, 4, 5, 6, 7), **S** = ø.

Шаг 2. Выбрать узел 6.

$\mathbf{S}$ = (6, 5)

$\overline{\mathbf{S}}$ = (1,2,3,4,7).



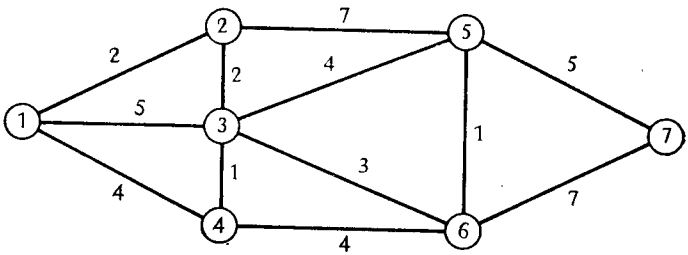
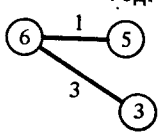


Рис. 2.28. Пример сети в задаче о кратчайшем остове.

Шаг 3. а. Выбрать узел 3. $S=(6, 5, 3)$, $\bar{S}=(1, 2, 4, 7)$.

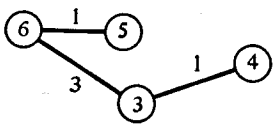
б. Выбрать узел 4. $S=(6, 5, 3, 4)$, $\bar{S}=(1, 2, 7)$.

Стоимость = 4 ед.



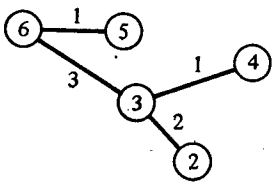
Стоимость = 4 ед.
Шаг 3 а

Стоимость = 5 ед.



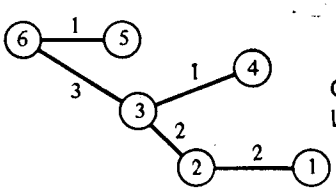
Стоимость = 5 ед.
Шаг 3 б

Стоимость = 7 ед.



Стоимость = 7 ед.
Шаг 3 в

Стоимость = 9 ед.

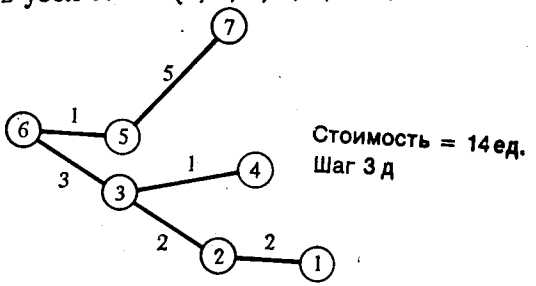


Стоимость = 9 ед.
Шаг 3 г

в. Выбрать узел 2. $S=(6, 5, 3, 4, 2)$, $\bar{S}=(1, 7)$.

г. Выбрать узел 1. $S=(6, 5, 3, 4, 1, 2)$, $\bar{S}=(7)$.

д. Выбрать узел 7. $S = (6, 5, 3, 4, 1, 2, 7)$, $\overline{S} = \varnothing$.



Алгоритм заканчивает работу, поскольку $\overline{S} = \varnothing$. Стоимость кратчайшего остова равна 14.

2.8.3. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ ПОСТРОЕНИЯ КРАТЧАЙШЕГО ОСТОВА

Назначение: построение кратчайшего остова неориентированной сети.

Локализация: подпрограмма MINSPA в пакете сетевой оптимизации.

Ограничения: программа позволяет обрабатывать сети, содержащие до 50 узлов и 50 дуг. Размеры сети можно увеличить, изменив границы массивов в операторах размерности, записанных в подпрограмме MINSPA и в основной программе.

Входные данные:

Набор 1. Одна карта с именем алгоритма MSTR в формате (A4).

Набор 2. Одна карта с числом узлов и числом дуг в сети в формате (2I10).

Набор 3. Общее число карт в данном наборе равно числу дуг в сети. С каждой карты считываются следующие величины: 1) номер начального узла дуги; 2) номер конечного узла дуги; 3) длина дуги.

Формат (4X, I6, I10, F10.2).

Набор 4. Данный набор состоит из одной карты, содержащей слово 'PEND' в формате (A4), которая указывает конец задачи.

Набор 5. Данный набор состоит из одной карты, содержащей слово 'EXIT' в формате (A4), которая указывает конец входных данных.

Используемые переменные: I — начальный узел дуги, J — конечный узел дуги, A1 — длина дуги, D — рабочий массив, содержащий длины дуг.

Используемый метод: выбрать узел 1 и соединить его с ближайшим к нему узлом. Затем найти узел, ближайший к уже

выбранным узлам, и провести соответствующую дугу. Выполнять данную процедуру до тех пор, пока все узлы не будут соединены.

Литература: Hillier S. F., Lieberman J. G., Operations Research, 2nd ed., Holden-Day, Inc., California, 1974.

2.8.4. РАСПРЕДЕЛЕНИЕ СРЕДСТВ НА РЕМОНТ АВТОСТРАДЫ

Отдел автомобильных дорог одного из округов располагает определенным фондом, предназначенным для покрытия асфальтом нескольких дорог и проведения на них ремонтных работ. Эти дороги соединяют 10 небольших сельских общин. Состояние поврежденных дорог таково, что в период выпадения осад-

ЗАДАЧА О КРАТЧАЙШЕМ ОСТОВЕ

КРАТЧАЙШИЙ ОСТОВ СОСТОИТ ИЗ СЛЕДУЮЩИХ ДУГ

НАЧАЛЬНЫЙ УЗЕЛ	КОНЕЧНЫЙ УЗЕЛ	ДЛИНА ДУГИ
1	2	7.00
2	3	4.00
3	7	2.00
3	8	6.00
4	5	6.00
5	6	4.00
6	7	5.00
6	10	4.00
6	9	3.00

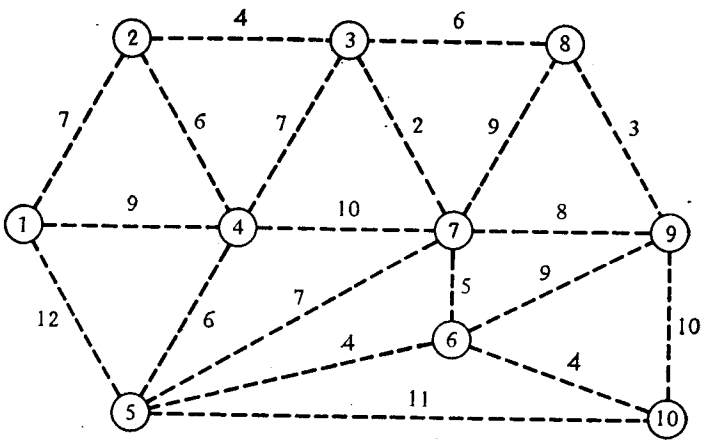


Рис. 2.29. Задача о кратчайшем остове.

ков они становятся непроходимыми. Кроме того, из-за возникновения на дорогах выбоин, их приходится периодически ремонтировать.

Для минимизации расходов отдел автомобильных дорог должен разработать план усовершенствованной системы взаимосвязи между всеми десятью общинами, реализация которого потребует минимума строительных материалов и рабочего вре-

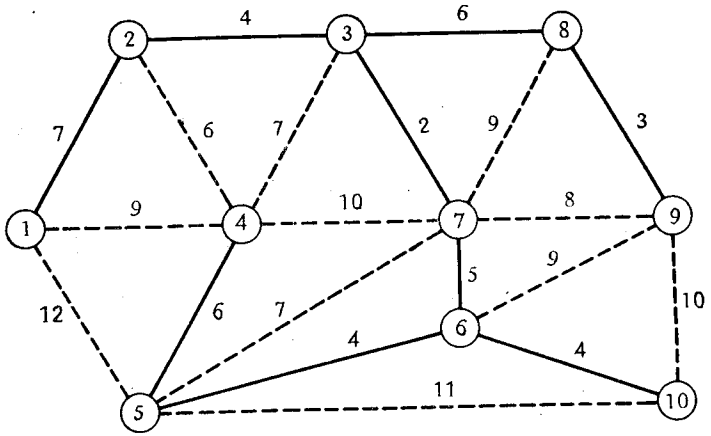


Рис. 2.30. Решение задачи о кратчайшем остове.

мени. На рис. 2.29 показано расположение 10 общин и задана сеть поврежденных дорог. Длина каждой дуги соответствует затратам, связанным с ремонтом данного участка дороги. Эта задача решается с помощью программы, реализующей алгоритм построения кратчайшего остова. На рис. 2.30 изображен кратчайший остов, являющийся частью дорог, которые соединяют все 10 городов и которые должны быть отремонтированы. На с. 107 приводятся результаты работы программы, с помощью которой было получено решение задачи.

2.8.5. ПРИМЕНЕНИЯ ЗАДАЧИ О КРАТЧАЙШЕМ ОСТОВЕ

Задача о кратчайшем остове (ЗКО) находит широкое практическое применение. Многие задачи, на первый взгляд не похожие на ЗКО, после некоторых преобразований сводятся к ней. С помощью построения кратчайшего остова, например, Д. Росси, Хейзер и Кинг [10] предложили схему прокладки телевизионных кабелей, соединяющих все станции в единую сеть. Но наиболее интересное и важное применение ЗКО, по-видимому, находит в кластерном анализе, где ряд задач, решаю-

щихся другими методами этого анализа, наиболее эффективно решаются с помощью построения КО [53]. Другое применение ЗКО находит при оценке надежности сетей: вес КО соответствует минимальной вероятности того, что дерево будет повреждено в одной или нескольких дугах. Гомори и Ху [25] использовали алгоритм построения КО при решении задач о многополюсных потоках. Хелд и Карп [26, 27] нашли аналогичное применение ЗКО для решения задачи коммивояжера.

2.9. ЗАДАЧА КОММИВОЯЖЕРА

Задача коммивояжера может быть сформулирована следующим образом. Коммивояжер должен выехать из заданного города, побывать в каждом из остальных n—1 городов ровно один раз и вернуться в исходный город. Задача заключается в определении последовательности объезда городов, при которой коммивояжеру требуется проехать наименьшее суммарное расстояние. При этом предполагается, что расстояние для каждой пары городов известно. Вместо длины пути можно рассматривать любые другие критерии эффективности, такие, как стоимость, время и т. д.

Нетрудно заметить, что всего существует $(n-1)!$ возможных маршрутов, среди которых один или несколько — оптимальные. Однако если некоторые города для коммивояжера недоступны, то минимальное значение целевой функции должно быть бесконечным. В большинстве случаев можно считать, что расстояние между городами i и j является симметричным, т. е. расстояние от города i до города j равно расстоянию от города j до города i. Однако для алгоритма, который приводится ниже, данное предположение можно опустить.

Приводимый ниже алгоритм называется алгоритмом *ветвей и границ*. Впервые он был разработан Литтлом и др. [41]. Алгоритм работает следующим образом. Вначале определяется некоторое допустимое решение, после чего множество всех оставшихся маршрутов разбивается на все более мелкие подмножества. На каждом шаге разбиения легко вычисляется нижняя граница длины текущего наилучшего маршрута. С помощью найденных границ производится дальнейшее разбиение подмножеств допустимых маршрутов и в конечном итоге определяется оптимальный маршрут. Если находится маршрут, длина которого не превосходит наименьшей нижней границы всех других маршрутов[1]), то данное промежуточное решение становится «наилучшим» допустимым решением. Основными

[1]) Здесь имеются в виду множества маршрутов, для которых к данному шагу алгоритма вычислены оценки. — *Прим. ред.*

процедурами предлагаемого алгоритма являются вычисление нижних границ длин маршрутов, выделение субоптимальных решений и ветвление с целью получения новых (более коротких) маршрутов.

Подмножества множества всех маршрутов можно рассматривать как узлы дерева, а процесс разбиения — как ветвление дерева. Поэтому данный метод называется методом поиска по дереву решений, или методом ветвей и границ.

Работу алгоритма мы опишем для одной модельной задачи. Параметры задачи коммивояжера, соответствующие расстояниям между городами, будут записаны в виде матрицы. Элемент этой матрицы, расположенный в i-й строке и j-м столбце, соответствует расстоянию от города i до города j, которое мы будем рассматривать как длину *звена* (i, j). Обозначим через **D** *матрицу расстояний,* каждый элемент которой соответствует расстоянию от города i до города j. Если в задаче n городов (остановок), то **D** является матрицей размером $n \times n$ с неотрицательными элементами d_{ij}. Вначале в качестве **D** выбирается исходная матрица расстояний, а в процессе работы алгоритма будут выполняться различные преобразования матрицы **D**. Маршрут **T** можно представить как множество n упорядоченных пар городов: $\mathbf{T}=\{(i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n), (i_n, i_1)\}$. Каждый допустимый маршрут представляет собой цикл, проходя по которому коммивояжер посещает каждый город ровно один раз и возвращается в исходный город. Каждая упорядоченная пара (i, j) является дугой, или звеном, маршрута. Длина $Z(\mathbf{T})$ маршрута **T** равна сумме соответствующих элементов матрицы расстояний:

$$Z(\mathbf{T}) = \sum_{(i,\ j)\ \in \mathbf{T}} d_{ij}.$$

Очевидно, что для любого допустимого маршрута каждая строка и каждый столбец матрицы **D** содержат ровно по одному элементу, соответствующему этому маршруту. Отметим, что величина $Z(\mathbf{T})$ определена для *любого* допустимого маршрута и что длина *оптимального* маршрута не может превосходить текущее значение $Z(\mathbf{T})$. Следовательно, текущее значение $Z(\mathbf{T})$ является *верхней границей* длины оптимального маршрута. Эту верхнюю границу будем обозначать через $Z_U(\mathbf{T})$.

2.9.1. ВЫЧИСЛЕНИЕ НИЖНИХ ГРАНИЦ

При определении нижних границ мы воспользуемся понятием редукции. Если из каждого элемента некоторой строки матрицы расстояний вычесть постоянную величину C, то длина любого маршрута, определяемая новой матрицей, меньше дли-

ны того же маршрута, определяемой старой матрицей, на величину C. Данное утверждение справедливо потому, что каждому маршруту соответствует ровно один элемент данной строки. Однако относительные длины всех маршрутов останутся неизменными. Следовательно, при переходе от исходной матрицы расстояний к новой останутся неизменными и все оптимальные маршруты. Очевидно, что аналогичное утверждение справедливо и для *элементов столбцов* матрицы расстояний.

Процедуры вычитания из каждого элемента строки наименьшего элемента этой же строки и из каждого элемента столбца наименьшего элемента этого же столбца называются *редукцией строк* и *редукцией столбцов* соответственно. Матрица с неотрицательными элементами, в *каждой* строке и *каждом* столбце которой содержится по крайней мере один нулевой элемент, называется *редуцированной матрицей.* Она может быть получена в результате последовательной редукции ее строк и столбцов. Если $Z(\mathbf{T})$ — длина маршрута $\mathbf{T}$, определяемая матрицей расстояний до выполнения редукции, $Z_1(\mathbf{T})$ — длина того же маршрута, определяемая редуцированной матрицей, а H — сумма всех констант, используемых при вычислении редуцированной матрицы, то $Z(\mathbf{T})=H+Z_1(\mathbf{T})$. Поскольку редуцированная матрица содержит только неотрицательные элементы, то H является нижней границей длины маршрута $\mathbf{T}$ для нередуцированной матрицы расстояний.

Рассмотрим задачу, в которой число городов равно 6 [41], а матрица расстояний задается табл. 2.14. Будем предполагать, что $d_{ij}=\infty$ для всех $i=j$.

Таблица 2.14. Исходная матрица расстояний

	1	2	3	4	5	6
1	∞	27	43	16	30	26
2	7	∞	16	1	30	30
3	20	13	∞	35	5	0
4	21	16	25	∞	18	18
5	12	46	27	48	∞	5
6	23	5	5	9	5	∞

Выберем вначале произвольный допустимый маршрут, например, маршрут, состоящий из звеньев (1, 4), (4, 5), (5, 3), (3, 6), (6, 2) и (2, 1). Длина данного маршрута равна $Z_U(\mathbf{T})=$ $=16+18+27+0+5+7=73$. Величина $Z_U(\mathbf{T})$ принимается в качестве верхней границы длины оптимального маршрута, т. е. для оптимального маршрута значение $Z(\mathbf{T})$ не может превосходить значения $Z_U(\mathbf{T})$. В начале работы алгоритма нахожде-

ние допустимого решения, позволяющего оценить оптимальное решение, не является обязательным. Однако часто определение верхней границы позволяет сократить проводимые вычисления.

Далее мы переходим к выполнению редукции, вначале строк, а затем столбцов матрицы расстояний. Для этого в *каждой* строке определяется минимальный элемент и найденное значение вычитается из элементов соответствующей строки. Результаты выполнения данной процедуры приведены в табл. 2.15, где столбец C_i содержит вычитаемые константы.

Таблица 2.15. Редукция строк

	1	2	3	4	5	6	C_i
1	∞	11	27	0	14	10	16
2	6	∞	15	0	29	29	1
3	20	13	∞	35	5	0	0
4	5	0	9	∞	2	2	16
5	7	41	22	43	∞	0	5
6	18	0	0	4	0	∞	5

Затем производится редукция столбцов. Из табл. 2.15 видно, что каждый столбец, кроме первого, содержит нулевой элемент. Поэтому может быть выполнена только редукция первого столбца. В табл. 2.16 приводятся результаты, полученные после вычитания из каждого элемента первого столбца числа 5. Строка Q_j содержит вычитаемые константы для каждого столбца. Значение H элемента, расположенного на пересечении строки Q_j и столбца C_i, равно сумме всех вычитаемых констант. Как видно из табл. 2.16, $H=\sum_{i,j}(C_i+Q_j)=48$. Данное значение является нижней границей $Z_L^0(\mathbf{T})$ длин *всех* маршрутов в рассматриваемой задаче.

Таблица 2.16. Редукция столбцов

	1	2	3	4	5	6	C_i
1	∞	11	27	0	14	10	16
2	1	∞	15	0	29	29	1
3	15	13	∞	35	5	0	0
4	0	0	9	∞	2	2	16
5	2	41	22	43	∞	0	5
6	13	0	0	4	0	∞	5
Q_j	5	0	0	0	0	0	48

Теперь задача заключается в нахождении оптимального (минимального) маршрута. Отметим, что в идеальном случае

поиск решения заключался бы в выборе ровно одного нулевого элемента в каждой строке и каждом столбце. Другими словами, если бы такой маршрут нулевой длины мог бы быть найден, то длина оптимального маршрута равнялась бы 48. Вместо того чтобы одновременно определять все звенья оптимального маршрута с помощью текущей матрицы расстояний, мы воспользуемся алгоритмом, на каждом шаге которого по матрице расстояний строится одно звено оптимального маршрута. В результате работы алгоритма будет построен один из маршрутов минимальной длины. Для построения решения вначале, по-видимому, целесообразно выбрать звено нулевой длины, а затем последовательно добавлять звенья нулевой или минимальной длины. Данная процедура основана на двух основных утверждениях. Во-первых, если *выбирается* звено (i, j), то решение не должно содержать других звеньев, соответствующих элементам i-й строки или j-го столбца. Во-вторых, если звено (i, j) можно *исключить* из окончательного решения, то его можно не рассматривать при выполнении последующих вычислений. Следовательно, для каждого звена достаточно рассмотреть следующие два случая. В первом из них звено включается в текущее (частичное) и все последующие (частичные) решения, пока определяется маршрут. Во втором случае звено исключается из дальнейшего рассмотрения.

2.9.2. ВЕТВЛЕНИЕ

Процесс разбиения множества всех маршрутов на непересекающиеся подмножества может быть представлен в виде ветвления дерева, как показано на рис. 2.31. Узел с пометкой «все маршруты» соответствует множеству всех маршрутов. Узел с пометкой (i, j) соответствует множеству «всех маршрутов», *содержащих* звено (i, j). Узел с пометкой $(\overline{i, j})$ соответствует подмножеству «всех маршрутов», *не содержащих* звено (i, j). На рис. 2.31 дальнейшее ветвление из узла $(\overline{i, j})$ не указывается. Узел с пометкой (k, l) соответствует всем маршрутам, содержащим *оба* звена (i, j) и (k, l). Как и раньше, узел с пометкой $(\overline{k, l})$ соответствует всем маршрутам, содержащим (i, j), но *не содержащим* (k, l). Цель ветвления можно легко объяснить следующим образом: если осуществляется разбиение множества всех маршрутов на все более мелкие подмножества, то в конце концов будет получено подмножество, содержащее один-единственный маршрут. Звенья, или пары городов, образующие этот маршрут, могут быть восстановлены, если двигаться по дереву в обратном направлении, т. е. к начальному узлу (с пометкой «все маршруты»). Процесс ветвления выполняется

на основе сравнения нижних границ. Если на некоторой итерации алгоритма нижняя граница длин маршрутов из одного подмножества превосходит нижнюю границу, соответствующую другому подмножеству, то первое подмножество можно исключить из рассмотрения на следующей итерации. Иными словами, ветвление из соответствующего узла не производится.

Предположим, что из некоторого узла X дерева решений исходят две дуги, и рассмотрим концевые узлы этих дуг. Тогда

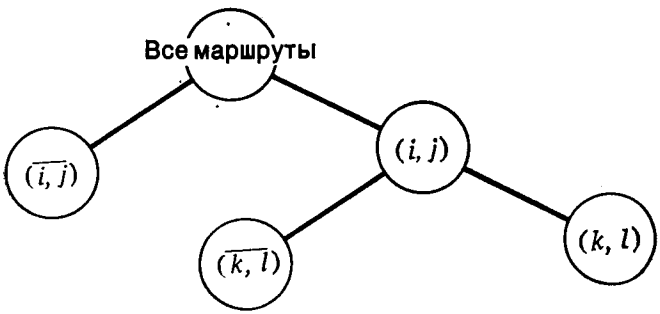


Рис. 2.31. Процесс построения подмножеств.

тот узел, который соответствует подмножеству маршрутов, *содержащих* новое звено, будем обозначать через Y, а узел, который соответствует подмножеству маршрутов, *не содержащих* нового звена, будем обозначать через $\overline{Y}$.

2.9.3. ПРОЦЕДУРА ВЫЧИСЛЕНИЙ

Вернемся к рассмотрению примера. На рис. 2.32 изображен начальный узел дерева, соответствующий множеству всех маршрутов, и указаны нижняя и верхняя границы длин всех маршрутов.

Следующим шагом процедуры является выбор звена, на котором будет базироваться ветвление. Цель ветвления заключается в разбиении множества маршрутов X на два таких подмножества Y и $\overline{Y}$, что наилучший маршрут из X *наиболее вероятно* содержится в Y и *наименее вероятно* — в $\overline{Y}$. Как отмечалось выше, в первую очередь во множество Y должны включаться маршруты, содержащие звенья (i,j), для которых $d_{ij}=0$. Однако в каждой строке и каждом столбце содержится по меньшей мере один нулевой элемент. Вопрос заключается в том, какое из этих звеньев следует включить в Y. Для того чтобы ответить на данный вопрос, рассмотрим маршруты, которые могут быть включены в $\overline{Y}$ (маршруты, не содержащие звено (i,j)). Поскольку город i должен быть связан с некоторым дру-

гим городом, что каждый маршрут из $\overline{Y}$ должен содержать звено, длина которого не меньше минимального элемента i-й строки, *не считая* $\mathbf{D}_{ij}$. Соответствующее расстояние обозначим через A_i. Поскольку необходимо, чтобы в город j можно было бы попасть из некоторого другого города, то каждый маршрут из $\overline{Y}$ содержит звено, длина которого не меньше минимального

$$Z_L^0(T) = 48 \leqslant \underset{\text{маршруты}}{\overset{\text{Все}}{(\quad)}} \leqslant 73 = Z_U^0(T)$$

Рис. 2.32. Границы длин маршрутов исходного множества.

элемента j-го столбца, *не считая* d_{ij}. Обозначим это расстояние через B_j, а сумму величин A_i и B_j — через ϕ_{ij}. Величины ϕ_{ij} будем называть *вторичным штрафом*. Если звено (i, j) *не* включается в возможный маршрут, то звено, соответствующее некоторому другому элементу i-й строки, и звено, соответствующее некоторому другому элементу j-го столбца, *должны* входить в окончательный маршрут. Величина ϕ_{ij} равна (минимальному) штрафу, которому мы подвергаемся, если *не включаем звено* (i, j) в оптимальный маршрут. Если штраф за *неиспользование* звена (i, j) вычислить для всех звеньев, для которых $d_{ij}=0$, то можно сравнить соответствующие значения ϕ_{ij} и включить в текущий маршрут звено (i, j), за неиспользование которого мы заплатили бы максимальный штраф. Очевидно, что выбор звена (i, j), соответствующего максимальному штрафу, позволяет определить новое подмножество маршрутов, не содержащих звена (i, j). Нижняя граница для соответствующей ветви должна быть выбрана таким образом, чтобы она не превосходила длины ни одного из маршрутов, не содержащих звена (i, j). Данное требование будет выполнено, если новую нижнюю границу положить равной сумме текущей нижней границы и максимального штрафа за *неиспользование* звена (i, j). Это означает, что для определения максимального значения ϕ_{ij} мы будем исследовать все элементы $d_{ij}=0$. Следует отметить, что $\phi_{ij}=0$ для всех (i, j), при которых $d_{ij}\neq 0$. Данное утверждение справедливо в силу того, что если положить $d_{ij}=\infty$ и затем произвести редукцию i-й строки и j-го столбца, то сумма вычитаемых констант будет равна ϕ_{ij}.

Для нашего примера значения A_i и B_j записываются в отдельном столбце и отдельной строке соответственно. Матрица $\mathbf{D}$ вместе с этими значениями выписана в табл. 2.17. Значения ϕ_{ij} для различных звеньев, или пар городов, приведены в табл. 2.18. Величина ϕ_{ij} выражает дополнительное расстояние,

которое коммивояжер проезжает в случае, когда в маршрут не включено звено (i, j), соответствующее элементу $d_{ij}=0$ редуцированной матрицы.

Таблица 2.17. Первая матрица решений

	1	2	3	4	5	6	C_i	A_i
1	∞	11	27	0	14	10	16	10
2	1	∞	15	0	29	29	1	1
3	15	13	∞	35	5	0	0	5
4	0	0	9	∞	2	2	16	0
5	2	41	22	43	∞	0	5	2
6	13	0	0	4	0	∞	5	0
Q_j	5	0	0	0	0	0	48	
B_j	1	0	9	0	2	0		

Как видно из табл. 2.18, максимальное значение ϕ_{ij} равно 10. Выбирая звено (1, 4), мы можем получить выигрыш в расстоянии, равный 10, т. е. больший, чем при выборе любого

Таблица 2.18. Вторичный штраф

Звено (i, j)	Расстояние $\varphi_{ij} = A_i + B_j$
(1, 4)	10 + 0 = 10
(2, 4)	1 + 0 = 1
(3, 6)	5 + 0 = 5
(4, 1)	0 + 1 = 1
(4, 2)	0 + 0 = 0
(5, 6)	2 + 0 = 2
(6, 2)	0 + 0 = 0
(6, 3)	0 + 9 = 9
(6, 5)	0 + 2 = 2

другого звена. Следовательно, в качестве базового звена ветвления мы выбираем звено (1, 4). На данном этапе $Y=(1, 4)$, $\overline{Y}=(\overline{1, 4})$. Нижней границей длин маршрутов из $\overline{Y}$ является величина $Z_L^0(\mathbf{T})+\phi_{14}$. В нашем примере она равна $48+10=58$. Перед тем как определить новую нижнюю границу для множества Y маршрутов, содержащих звено (1, 4), необходимо выполнить некоторое преобразование матрицы **D**. Отметим, что если мы включаем в маршруты некоторое звено (k, l), то в дальнейшем можно не рассматривать k-ю строку и l-й столбец. Следовательно, в данном примере первую строку и четвертый столбец можно исключить из рассмотрения. Далее, если звено (k, l) принадлежит некоторому маршруту, т. е. (k, l) является

звеном некоторого ориентированного цикла из Y, то звено (l, k) не может принадлежать маршруту из Y. Выполнения данного условия можно достичь, полагая на всех последующих итерациях $d_{lk}=\infty$. И наконец, могут существовать другие звенья, с помощью которых в дальнейшем также могут быть образованы подмаршруты. (Подмаршрут — это цикл, включающий в себя неполное множество городов.) Эти звенья называются *запрещенными*. Их можно исключить из рассмотрения, полагая элементы матрицы **D**, соответствующие длинам этих звеньев, равными ∞. После проведенного преобразования матрицы **D** процедуру редукции можно выполнять с любым столбцом, который содержал один-единственный нулевой элемент, расположенный в k-й позиции, и с любой строкой, которая содержала один-единственнуй нулевой элемент, расположенный в l-й позиции. Кроме того, не исключена возможность выполнения процедуры редукции со всеми другими строками и столбцами, содержащими элементы, которые соответствуют запрещенным звеньям. Нижняя граница для Y может быть теперь вычислена как сумма всех новых вычитаемых констант и старой нижней границы. Модифицированная матрица **D**, а также значения C_i, Q_j, A_i и B_j приводятся в табл. 2.19. Отметим, что первая строка и четвертый столбец были вычеркнуты и что запрещенных звеньев в данном случае не существует.

Таблица 2.19. Вторая матрица решений

	1	2	3	5	6	C_i	A_i
2	0	∞	14	28	28	1	14
3	15	13	∞	5	0	0	5
4	∞	0	9	2	2	0	2
5	2	41	22	∞	0	0	2
6	13	0	0	0	∞	0	0
Q_j	0	0	0	0	0	1	
B_j	2	0	9	2	0		

Новая нижняя граница для Y в данном примере равна $48+1=49$. Дерево решений теперь может быть изображено так, как это показано на рис. 2.33.

Значения ϕ_{ij} после второй редукции следующие: $\phi_{21}=16$, $\phi_{36}=5$, $\phi_{42}=2$, $\phi_{56}=2$, $\phi_{62}=0$, $\phi_{63}=9$ и $\phi_{65}=2$. Максимальным среди них является значение $\phi_{21}=16$, и новая нижняя граница для $\overline{Y}$ равна $49+16=65$. Рассмотрим теперь множество Y маршрутов, *содержащих* звено (2, 1). Вычеркнем вначале в матрице **D** вторую строку и первый столбец. Длина звена (1, 2) равна теперь ∞, однако данное значение не содержится в приво-

Таблица 2.20. Третья матрица решений

	2	3	5	6	C_i	A_i
3	13	∞	5	0	0	5
4	∞	7	0	0	2	0
5	41	22	∞	0	0	22
6	0	0	0	∞	0	0
Q_j	0	0	0	0	2	
B_j	13	7	0	0		

димой таблице. Отметим, что звено (4, 2) является теперь запрещенным, поскольку оно могло бы образовать подмаршрут. Чтобы предотвратить это, полагаем $d_{42}=\infty$. В табл. 2.20 при-

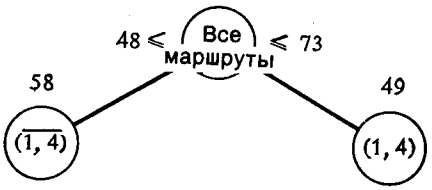


Рис. 2.33. Границы на первой итерации

ведена матрица **D** после выполнения редукции. Новая нижняя граница для Y равна $49+2=51$, а дерево решений на данном этапе выглядит так, как показано на рис. 2.34.

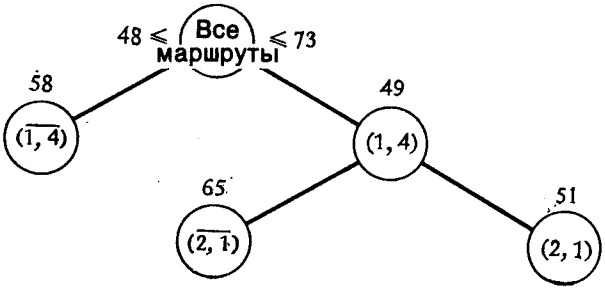


Рис. 2.34. Границы на второй итерации.

Для изображенных здесь множеств минимальная нижняя граница равна 51 и соответствует узлу (2, 1). Поэтому на следующей итерации ветвление будет осуществляться из узла (2, 1). Продолжая аналогичные вычисления, мы должны были бы осуществлять ветвление из узлов с наименьшей нижней гра-

ницей. Предположим, однако, что процесс ветвления выполняется так, как это показано на рис. 2.35, и будем рассматривать полученный в результате ветвления полный маршрут. Соответствующая матрица решения приведена в табл. 2.21.

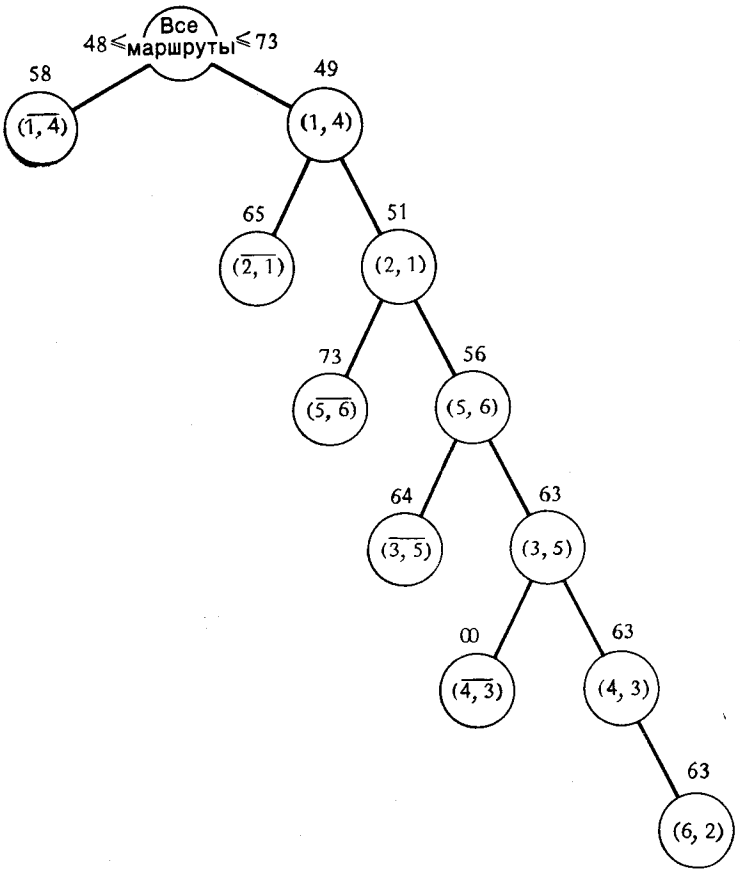


Рис. 2.35. Промежуточное решение.

Построенный полный маршрут является оптимальным, если его длина не превосходит длины любого маршрута, соответствующего другим звеньям дерева. Однако нижняя граница для узла $(\overline{1, 4})$, равная 58, меньше длины построенного маршрута, равной 63. Следовательно, необходимо исследовать подмножество маршрутов, которые *не содержат* звено (1, 4). Матрица **D** на данном этапе будет совпадать со старой матрицей **D**, кото-

Таблица 2.21. Окончательная матрица решений

	2	3	C_i	A_i
4	∞	0	7	∞
5	0	∞	0	0
Q_j	0	0	7	
B_j	∞	0		

рую мы имели до этапа ветвления из узла (1, 4). Однако, для того чтобы исключить все маршруты, содержащие звено (1, 4), значение элемента d_{14} матрицы **D** мы принимаем равным ∞. В нашем примере соответствующая матрица совпадает с мат-

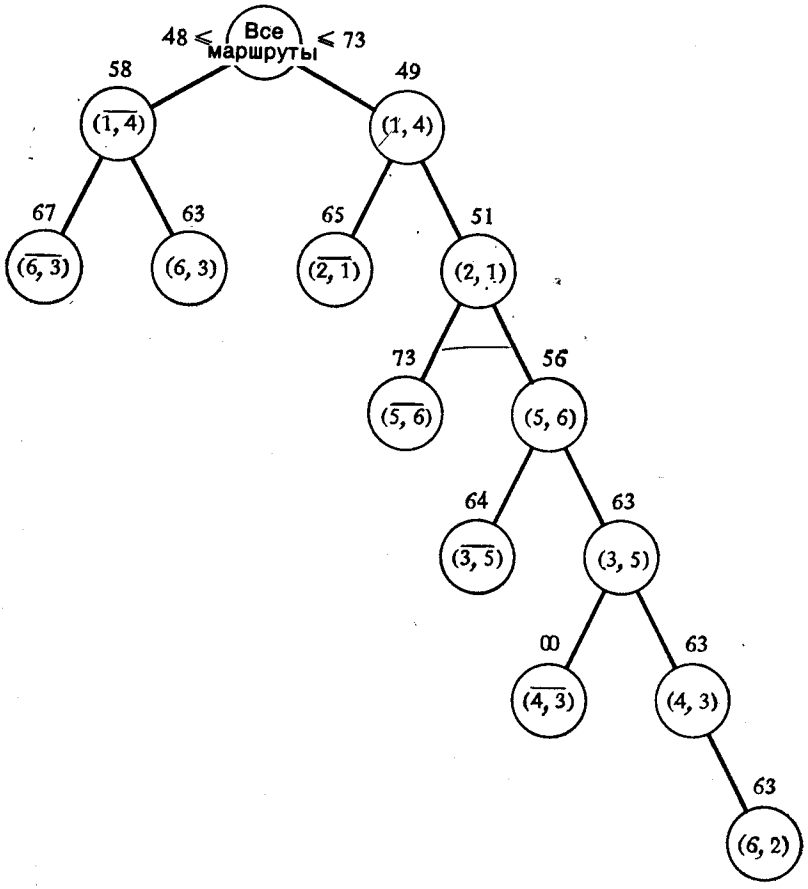


Рис. 2.36. Возврат.

рицей **D**, приведенной в табл. 2.14, если элемент $d_{14}=16$ заменить на $d_{14}=\infty$. Далее с этой новой матрицей выполняются описанные выше процедуры ветвления и построения границ. Полученное при этом дерево изображается на рис. 2.36. Дан-

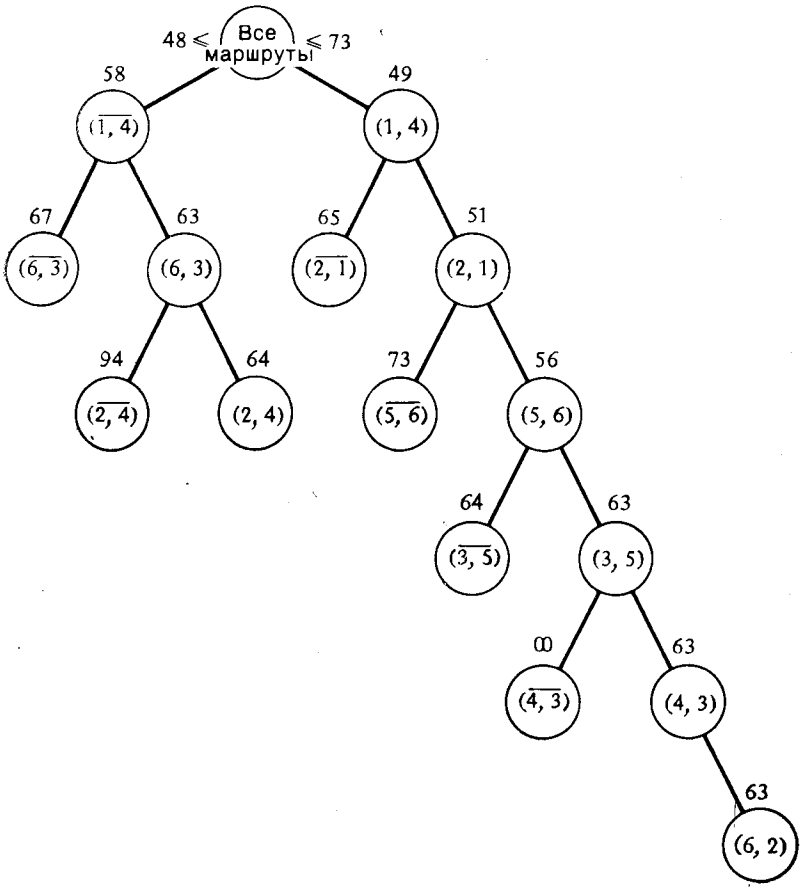


Рис. 2.37. Оптимальное решение.

ная процедура анализа предыдущих промежуточных точек ветвления, которые могли бы определить более короткий маршрут, называется *возвратом*.

Проводя аналогичные вычисления, построим разбиение подмножества маршрутов, содержащих звено (6, 3). Полученное при этом дерево изображено на рис. 2.37.

Заметим, что нижняя граница первоначального маршрута, содержащего звено (1, 4), теперь является наименьшей. Следовательно, этот маршрут должен быть оптимальным. Оптимальный маршрут состоит из следующих звеньев, или пар городов: (4, 3), (6, 2), (3, 5), (5, 6), (2, 1) и (1, 4). Он является ориентированным циклом, длина которого равна 63.

2.9.4. ЗАКЛЮЧИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

Следует отметить, что задача коммивояжера является комбинаторной задачей, поскольку число возможных маршрутов экспоненциально зависит от числа городов. Данная задача принадлежит классу так называемых *NP-полных задач*. Было показано, что многочисленный класс комбинаторных задач является классом эквивалентных задач, где эквивалентность понимается в том смысле, что либо все они могут быть решены с помощью алгоритмов, имеющих полиномиальную сложность вычислений, либо ни одна из них не может быть решена с помощью таких алгоритмов (см. разд. 2.7). Данная группа задач образует класс *NP*-полных задач. И наконец, задача коммивояжера не может быть непосредственно сформулирована и решена как задача линейного программирования. Она относится к классу потоковых задач в силу исторически сложившейся связи между методами их решения и в силу ее графического представления как задачи об оптимальном маршруте.

Ряд задач, представляющих собой разновидность задачи коммивояжера, был поставлен и решен с целью исследовать маршруты движения людей или перевозки товаров. Однако рассмотрение этих прикладных задач отодвигает на второй план одну важную особенность задачи коммивояжера. Основное различие между задачей коммивояжера и другими задачами о кратчайшем пути заключается в том, что в первой из них требуется существование ориентированного цикла, в который ровно один раз входят все узлы (города или километровые столбы) сети. Другие важные приложения задачи коммивояжера связаны с определением нагрузки учителей, составлением расписания выполнения разных работ на различных станках, а также с нахождением последовательности выполнения операций на универсальном станке. Кроме того, на практике встречаются задачи, связанные с составлением расписания работы станков на некотором предприятии, нахождением оптимальной последовательности переключения их скоростей и определением порядка их обслуживания.

2.9.5. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА

Назначение: данная программа позволяет находить оптимальное решение задачи коммивояжера.

Локализация: подпрограмма TRASAL в пакете сетевой оптимизации.

Ограничения: программа позволяет обрабатывать сети, содержащие до 50 узлов и 50 дуг. Размеры сети можно увеличить, изменив границы массивов в операторах размерности, записанных в подпрограмме TRASAL и основной программе.

Входные данные:

Набор 1. Одна карта с именем алгоритма TSPR в формате (A4).

Набор 2. Одна карта с числом узлов и числом дуг в сети в формате (2I10).

Набор 3. Общее число карт в данном наборе равно числу дуг в сети. С каждой карты считываются следующие значения: 1) номер начального узла дуги; 2) номер конечного узла дуги; 3) длина дуги.

Формат (4X, I6, I10, F10.2).

Набор 4. Данный набор состоит из одной карты, содержащей слово 'PEND' в формате (A4), которая указывает конец задачи.

Набор 5. Данный набор состоит из одной карты, содержащей слово 'EXIT' в формате (A4), которая указывает конец входных данных.

Составные части программы. Программа состоит из следующих подпрограмм и функций: подпрограмма TRASAL (ввод-вывод, построение дерева, возврат, отсечение путей); функция MINCOL (редукция столбца); функция NINROW (редукция строки); подпрограмма PRTRXS (печать начального решения).

Используемые переменные: I — начальный узел дуги, J — конечный узел дуги, VAL — длина дуги.

Используемый метод: данный алгоритм основан на методе «ветвей и границ», описанном Литтлом и др.

Литература: [41].

2.9.6. СОСТАВЛЕНИЕ МАРШРУТА ЗАРУБЕЖНОГО ПУТЕШЕСТВИЯ

Выиграв третий приз в ирландской лотерее, один дипломник Техасского университета решил, что после пяти лет непрерывной учебы на инженерно-технологическом факультете настало время вырваться из дома и посмотреть на жизнь в других стра-

Таблица 2.22. Ирландская лотерея

Из	В							
	Колледж-Стейшен	Акапулько	Гонолулу	Токио	Гонконг	Лондон	Сидней	Рим
Колледж-Стейшен	—	190	210	680	690	460	780	750
Акапулько	190	—	380	760	790	610	670	450
Гонолулу	210	380	—	890	900	340	410	600
Токио	680	760	890	—	480	760	510	250
Гонконг	690	790	900	480	—	890	490	560
Лондон	460	610	340	760	890	—	720	600
Сидней	780	670	410	510	490	720	—	500
Рим	750	450	600	250	560	600	500	—

нах. Он выбрал семь мест, наиболее интересных для посещения в рождественские каникулы. В местном бюро путешествий он получил информацию о стоимости проезда самолетом в каж-

РЕШЕНИЕ ЗАДАЧИ КОММИВОЯЖЕРА МЕТОДОМ ВЕТВЕЙ И ГРАНИЦ

ИСХОДНАЯ МАТРИЦА

	1	2	3	4	5
1	99999999	19	21	68	69
2	19	99999999	38	76	79
3	21	38	99999999	89	90
4	68	76	89	99999999	48
5	69	79	90	48	99999999
6	46	61	34	76	89
7	78	67	41	51	49
8	75	45	60	25	56

	6	7	8
1	46	78	75
2	61	67	45
3	34	41	60
4	76	51	25
5	89	49	56
6	99999999	72	60
7	72	99999999	50
8	60	50	99999999

```
**ДОПУСТИМОЕ РЕШЕНИЕ** 307

1  2  8  4  5  7  3  6

ОПТИМАЛЬНОЕ РЕШЕНИЕ  307

ИЗ УЗЛА 1  В УЗЕЛ 2  РАССТОЯНИЕ 19
ИЗ УЗЛА 2  В УЗЕЛ 8  РАССТОЯНИЕ 45
ИЗ УЗЛА 8  В УЗЕЛ 4  РАССТОЯНИЕ 25
ИЗ УЗЛА 4  В УЗЕЛ 5  РАССТОЯНИЕ 48
ИЗ УЗЛА 5  В УЗЕЛ 7  РАССТОЯНИЕ 49
ИЗ УЗЛА 7  В УЗЕЛ 3  РАССТОЯНИЕ 41
ИЗ УЗЛА 3  В УЗЕЛ 6  РАССТОЯНИЕ 34
ИЗ УЗЛА 6  В УЗЕЛ 1  РАССТОЯНИЕ 46
```

дый из городов, которые он собирался посетить, и стоимость проезда из одного города в другой. Соответствующие цены (в долл.) приведены в табл. 2.22. Поскольку плата за проезд из одного города в другой равна плате за проезд в обратном направлении, то матрица стоимостей является симметричной.

Перед тем как заказать билеты на самолет, он решил определить наиболее дешевый маршрут объезда всех выбранных им городов. Это не представляло для него трудностей, поскольку он знал алгоритм решения задачи коммивояжера.

Соответствующая задача коммивояжера была решена с помощью программы, описанной в разд. 2.9.5. Самым дешевым маршрутом для посещения выбранных им городов оказался маршрут Колледж-Стейшен — Лондон — Гонолулу — Сидней — Гонконг — Токио — Рим — Акапулько и снова Колледж-Стейшен. Общая стоимость за проезд при этом равна 3070 долл.

2.10. ТРАНСПОРТНАЯ ЗАДАЧА

В стандартной постановке данной задачи заданы m пунктов снабжения, из которых продукт может транспортироваться в каждый из n пунктов потребления. Производительность i-го источника снабжения равна a_i, а потребление j-го стока равно b_j. Параметры a_i и b_j являются фиксированными в заданном периоде планирования. Предполагается, что стоимость транспортировки единицы продукта из пункта i в пункт j не зависит от объема перевозимого груза и равна c_{ij}. Требуется для заданного периода планирования найти такую схему транспортировки продукта, при которой общие транспортные затраты ми-

нимальны. Здесь предполагается, что учет ведется по одному виду продукции. Однако в гл. 5 мы докажем, что многопродуктовые транспортные задачи сводятся к однопродуктовым задачам.

Основными понятиями, необходимыми для формулирования транспортной задачи, являются понятия стоимости транспортировки единицы продукта, производительности пунктов снабжения и спроса пунктов потребления. Рис. 2.38 дает графическую интерпретацию данной задачи.

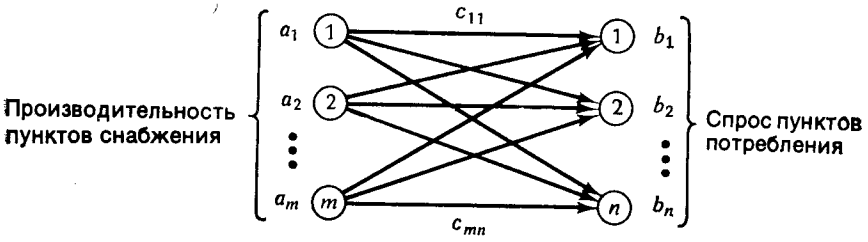


Рис. 2.38. Графическая интерпретация транспортной задачи.

Входная информация в транспортной задаче обычно представляется двумерным массивом, содержащим стоимости перевозок продукта по всевозможным маршрутам, а также производительности пунктов снабжения и спросы пунктов потребления. Пример такого массива для задачи, описанной на рис. 2.38, приведен в табл. 2.23.

Таблица 2.23. Входная информация в транспортной задаче

	$j=1$	$j=2$	$j=3$	$\cdots$	$j=n$	
$i=1$	c_{11}	c_{12}	c_{13}	$\cdots$	c_{1n}	a_1
$i=2$	c_{21}	c_{22}	c_{23}	$\cdots$	c_{2n}	a_2
$i=3$	c_{31}	c_{32}	c_{33}	$\cdots$	c_{3n}	a_3
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$i=m$	c_{m1}	c_{m2}	c_{m3}	$\cdots$	c_{mn}	a_m
	b_1	b_2	b_3	$\cdots$	b_n	

Рассмотрим следующий пример, в котором заданы два источника снабжения и три пункта потребления и, кроме того, предполагается, что если стоимость производства единицы продукции не постоянна для каждого источника снабжения, то ве-

личина c_{ij} может включать в себя стоимость производства единицы продукции i-м источником:

	1	2	3	a_i
1	3	4	7	20
2	6	3	2	10
b_j	10	12	8	

Если в силу причин физического или экономического характера некоторый маршрут не может быть использован, то стоимость транспортировки единицы продукта по данному маршруту принимается равной ∞.

2.10.1. МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА

Пусть x_{ij} — количество продукта, транспортируемого из i-го источника в j-й сток. Целевая функция в данной модели соответствует общим транспортным затратам. Накладываемые ограничения нужны для того, чтобы вся произведенная продукция использовалась и спрос каждого пункта потребления удовлетворялся. Задача формулируется следующим образом:

$$\text{минимизировать } Z=\sum_i \sum_j c_{ij}x_{ij}$$

при условии, что

$$\sum_j x_{ij}=a_i, \quad i=1,\ldots,m,$$

$$\sum_i x_{ij}=b_j, \quad j=1,\ldots,n,$$

$$x_{ij}\geqslant 0, \quad i=1,\ldots,m, \quad j=1,\ldots,n.$$

Для этой задачи выполнены некоторые необходимые и достаточные условия существования целочисленного оптимального решения для задач ЛП с целочисленными параметрами. Для существования решения в сформулированной задаче мы требуем, чтобы предложение и спрос удовлетворяли соотношению $\sum_i a_i=\sum_j b_j$, которое означает, что по крайней мере одно ограничение является избыточным. Если в качестве целевой функции выбрать $-Z$, то можно дать эквивалентную формулировку данной задачи:

$$\text{максимизировать } \sum_i \sum_j (-c_{ij})\, x_{ij}$$

при условии, что

$$\sum_j x_{ij} = a_i, \quad i = 1,\ldots,m,$$

$$\sum_i x_{ij} = b_j, \quad j = 1,\ldots,n,$$

$$x_{ij} \geqslant 0, \quad i = 1,\ldots,m, \quad j = 1,\ldots,n.$$

Ниже будет дано описание симплексного алгоритма для транспортной задачи. Для этого сформулируем двойственную задачу по отношению к предыдущей задаче линейного программирования:

$$\text{минимизировать} \left\{\sum_i a_i R_i + \sum_j b_j K_j\right\}$$

при условии, что

$$R_i + K_j + c_{ij} \geqslant 0 \text{ для всех } i \text{ и всех } j.$$

В данной задаче R_i — это двойственная переменная, соответствующая i-му ограничению на предложение, а K_j — двойственная переменная, соответствующая j-му ограничению на спрос. Обе переменные могут принимать как положительные, так и отрицательные значения, поскольку они соответствуют ограничениям типа равенства в прямой задаче.

Для рассмотренного выше примера прямая и двойственная задачи линейного программирования формулируются следующим образом.

Прямая задача.

$$\text{Максимизировать } \{-3x_{11} - 4x_{12} - 7x_{13} - 6x_{21} - 3x_{22} - 2x_{23}\}$$

при условии, что

$$\begin{aligned}
x_{11} + x_{12} + x_{13} &= 20,\\
x_{21} + x_{22} + x_{23} &= 10,\\
x_{11} + x_{21} &= 10,\\
x_{12} + x_{22} &= 12,\\
x_{13} + x_{23} &= 8,\\
x_{ij} &\geqslant 0.
\end{aligned}$$

Двойственная задача.

$$\text{Минимизировать } \{20R_1 + 10R_2 + 10K_1 + 12K_2 + 8K_3\}$$

при условии, что

$$\begin{aligned}
R_1 + K_1 &\geqslant -3,\\
R_1 + K_2 &\geqslant -4,\\
R_1 + K_3 &\geqslant -7,\\
R_2 + K_1 &\geqslant -6,\\
R_2 + K_2 &\geqslant -3,\\
R_2 + K_3 &\geqslant -2.
\end{aligned}$$

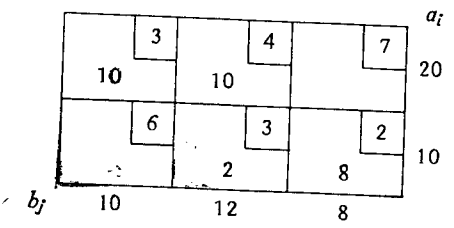
R_i и K_j могут принимать как положительные, так и отрицательные значения.

Допустимое решение транспортной задачи может быть записано в виде следующего двумерного массива, состоящего из m строк и n столбцов. Значение элемента x_{ij} этого массива равно количеству продукта, транспортируемого из i-го источника в j-й сток:

$$\begin{bmatrix}
x_{11} & x_{12} & x_{13} & \cdots & x_{1n}\\
x_{21} & x_{22} & x_{23} & \cdots & x_{2n}\\
\cdot & \cdot & \cdot & & \cdot\\
\cdot & \cdot & \cdot & & \cdot\\
\cdot & \cdot & \cdot & & \cdot\\
x_{m1} & x_{m2} & x_{m3} & \cdots & x_{mn}
\end{bmatrix}$$

Допустимое решение, соответствующие ему транспортные затраты и величины предложения и спроса могут быть записаны в виде одного массива, как это показано на рис. 2.39.

Для рассматриваемого числового примера допустимое решение может быть записано в следующем виде:



Общие транспортные затраты в данном случае равны $10 \cdot 3 + 10 \cdot 4 + 2 \cdot 3 + 8 \cdot 2 = 92$.

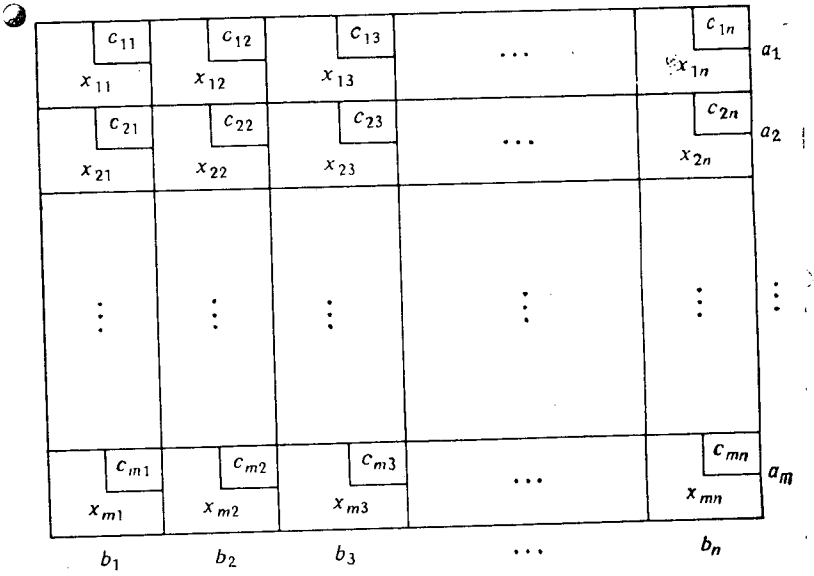


Рис. 2.39. Матрица условий транспортной задачи.

2.10.2. СИМПЛЕКСНЫЙ АЛГОРИТМ ДЛЯ ТРАНСПОРТНОЙ ЗАДАЧИ

Перед тем как перейти к описанию алгоритма, введем несколько важных понятий. Базисным допустимым решением транспортной задачи с m источниками и n стоками называется допустимое решение, содержащее не более $m+n-1$ положительных переменных. Как отмечалось в разд. 2.10.1, в транспортной задаче, решаемой методом линейного программирования, не более $m+n-1$ независимых ограничений. Совокупность mn переменных базисного решения может быть разбита на две группы, в одну из которых входят $m+n-1$ переменных, а во вторую $mn-(m+n-1)$ переменных. Базисное допустимое решение можно найти, полагая значения переменных второй группы равными нулю и определяя затем значения переменных первой группы как решение системы уравнений, соответствующих независимым ограничениям задачи. Переменные первой группы называются *базисными,* а переменные второй группы — *внебазисными.* Маршрут будем называть базисным, если он соответствует базисным переменным (т. е. поток по нему не равен нулю), и внебазисным, если он соответствует внебазисным переменным. Говорят, что базисное решение является вырожденным, если по крайней мере одна из его базисных переменных равна нулю.

Симплексный алгоритм для транспортной задачи начинает работу при некотором невырожденном базисном допустимом решении. Затем, выполняя проверку, аналогичную проверке условия оптимальности для задачи линейного программирования,

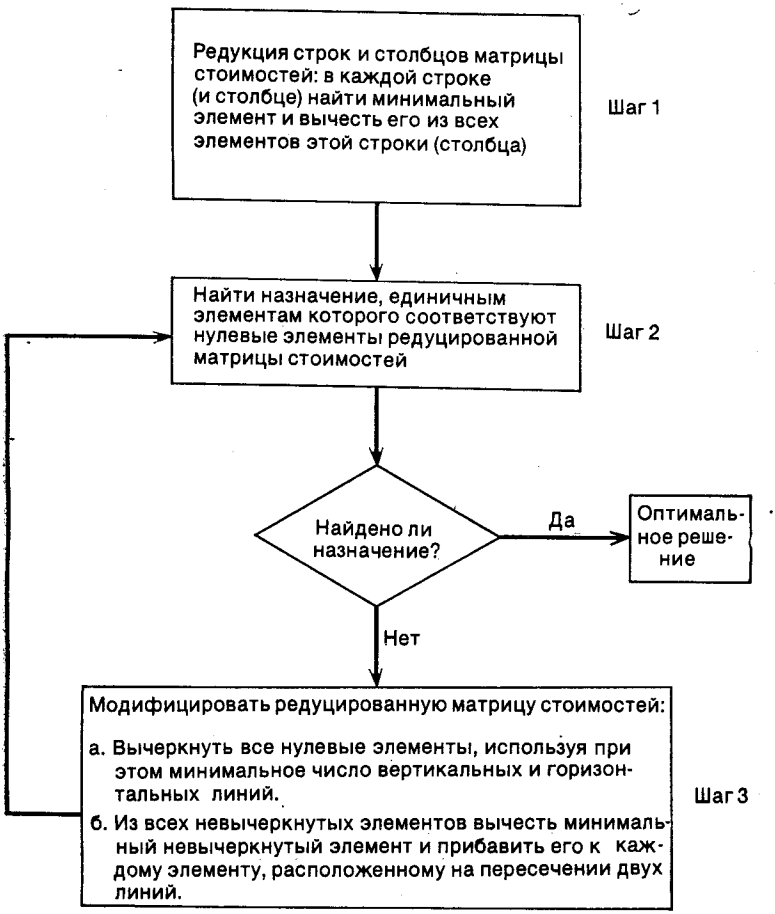


Рис. 2.40. Блок-схема симплексного алгоритма для транспортной задачи.

проверяется, не окажется ли полезной замена одного из текущих базисных маршрутов другим (внебазисным), еще не использованным маршрутом. Если в результате такой проверки устанавливается, что замена некоторого маршрута приводит к улучшению решения, то все потоки должны быть выбраны таким образом, чтобы базисное решение оставалось допустимым.

Алгоритм состоит из четырех шагов. Назначение каждого шага и его связь с другими шагами показаны на блок-схеме, изображенной на рис. 2.40. В дальнейшем для простоты изложения множество базисных маршрутов будем называть *базисом.*

Обозначим через **Y** вектор двойственных переменных, т. е. $\mathbf{Y}=[R_1, R_2, ..., R_m, K_1, K_2, ..., K_n]$, и пусть $\mathbf{A}_{ij}$ — вектор коэффициентов, с которыми переменные x_{ij} входят в ограничения в прямой задаче. Как было установлено при рассмотрении примера из разд. 2.10.1, вектор $\mathbf{A}_{ij}$ может быть записан в виде

$$\mathbf{A}_{ij}=\begin{bmatrix}0\\ \cdot\\ \cdot\\ \cdot\\ 1\\ 0\\ \cdot\\ \cdot\\ \cdot\\ 1\\ 0\\ \cdot\\ \cdot\\ \cdot\\ 0\end{bmatrix}\begin{matrix}\\ \\ \\ \\ \leftarrow \text{i-я позиция}\\ \\ \\ \\ \\ \leftarrow \text{(m + j)-я позиция}\\ \\ \\ \\ \\ \end{matrix}$$

Условие оптимальности для прямой задачи может быть записано в виде неравенства $\mathbf{YA}_{ij}+c_{ij}\geqslant 0$. Поскольку $\mathbf{YA}_{ij}=R_i+K_j$, то данное условие можно переписать в виде $R_i+K_j+c_{ij}\geqslant 0$, что совпадает с условием допустимости для *двойственной задачи.* Это означает, что **Y** не является допустимым решением двойственной задачи, а решение прямой задачи не является оптимальным, если для некоторого внебазисного маршрута, соединяющего i-й источник с j-м стоком, величина $R_i+K_j+c_{ij}<0$. Кроме того, наименьшая отрицательная величина $R_i+K_j+c_{ij}$ соответствует маршруту с наибольшей возможностью улучшить решение. Перейдем к описанию алгоритма.

Шаг 1. Выбор начального допустимого решения. Существует несколько методов построения начального допустимого решения. Наиболее распространенными из них являются, вероятно, *метод «северо-западного угла»* и *приближенный метод Фогеля* (ПМФ). Остановимся на кратком описании этих методов.

Метод «северо-западного угла». Определяем для маршрута, соединяющего первый источник с первым стоком, максимально допустимый поток. Пусть x_{11} — число единиц этого потока. Если $a_1=b_1$, то удаляем источник 1 и сток 1 и выбираем маршрут,

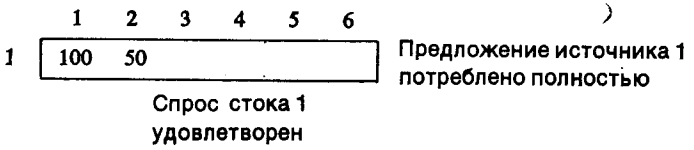
соответствующий переменной x_{22}. В противном случае поступаем следующим образом. Если предложение источника реализовано полностью, то удаляем источник и выбираем маршрут, соответствующий переменной x_{21}. Если же удовлетворяется спрос стока, то удаляем сток и выбираем маршрут, соответствующий переменной x_{12}. Определяем для выбранного маршрута максимально допустимый поток и удаляем либо источник, если его предложение реализовано полностью, либо сток, если его спрос удовлетворяется. Если одновременно выполняются условия на предложение и спрос, то удаляется и источник, и сток. Продолжаем выполнять аналогичную процедуру. Пусть x_{ij} — последняя из выбранных базисных переменных. Тогда на следующем шаге выбирается переменная $x_{i,\,j+1}$, если предложение i-го источника реализовано не полностью, переменная $x_{i+1,\,j}$, если спрос j-го стока удовлетворен не полностью, и $x_{i+1,\,j+1}$ — в противном случае.

Таблица 2.24. Начальное решение, найденное с помощью метода «северо-западного угла»

	1	2	3	4	5	6	a_i
1	19	49	43	55	47	53	150
2	20	37	28	42	30	38	200
3	19	33	32	40	70	35	300
4	20	22	20	30	25	M	350
5	20	32	29	48	37	41	250
b_j	100	150	250	250	300	200	

В качестве примера рассмотрим транспортную задачу, описанную в табл. 2.24, и найдем для нее начальное решение с помощью метода «северо-западного угла». Через M здесь обозначено произвольное достаточно большое число.

а. Задание 1-й строки



б. Задание 2-й строки

	1	2	3	4	5	6
1	100	50				
2		100	100			

Предложение источника 2 потреблено полностью

Спрос стока 2 удовлетворен

в. Задание 3-й строки

	1	2	3	4	5	6
1	100	50				
2		100	100			
3			150	150		

Предложение источника 3 потреблено полностью

Спрос стока 3 удовлетворен

г. Задание 4-й строки

	1	2	3	4	5	6
1	100	50				
2		100	100			
3			150	150		
4				100	250	

Предложение источника 4 потреблено полностью

Спрос стока 4 удовлетворен

д. Начальное базисное допустимое решение, полученное с помощью метода «северо-западного угла»

	1	2	3	4	5	6	a_i	
1	100	50					150	
2		100	100				200	
3			150	150			300	
4				100	250		350	
5					50	200	250	$Z = 40\,950$
b_j	100	150	250	250	300	200		

Приближенный метод Фогеля. Данный эвристический метод является более совершенным по сравнению с методом «северо-западного угла», поскольку при выборе маршрута с помощью ПМФ используется информация о транспортных затратах. Для каждого источника (и каждого стока) вычисляется минимальный штраф за отказ воспользоваться наиболее экономным маршрутом, исходящим из этого источника (заходящим в этот

сток). Затем среди вычисленных значений минимального штрафа определяется наибольшее и выбирается соответствующий этому значению источник или сток. Наиболее экономному маршруту, исходящему из выбранного источника (или заходящему в выбранный сток), назначается максимально допустимый поток. Смысл такого назначения заключается в том, что оно позволяет избежать наибольшего из минимальных значений штрафа. После «насыщения» выбранного маршрута необходимо выполнить проверку условий, касающихся предложения и спроса соответственно в начальном и конечном узлах этого маршрута. Источник или сток, для которого соответствующее условие выполнено, исключается, и вычисляются новые значения штрафа. Данная процедура повторяется до тех пор, пока не будут исключены все источники и стоки. Штраф для каждого источника или стока легко вычисляется. Он равен разности между двумя минимальными стоимостями. Если эти стоимости равны между собой, то штраф равен нулю. Кроме того, если на множестве источников и стоков существует несколько максимальных значений минимального штрафа, то можно выбрать произвольный узел, соответствующий одному из этих значений.

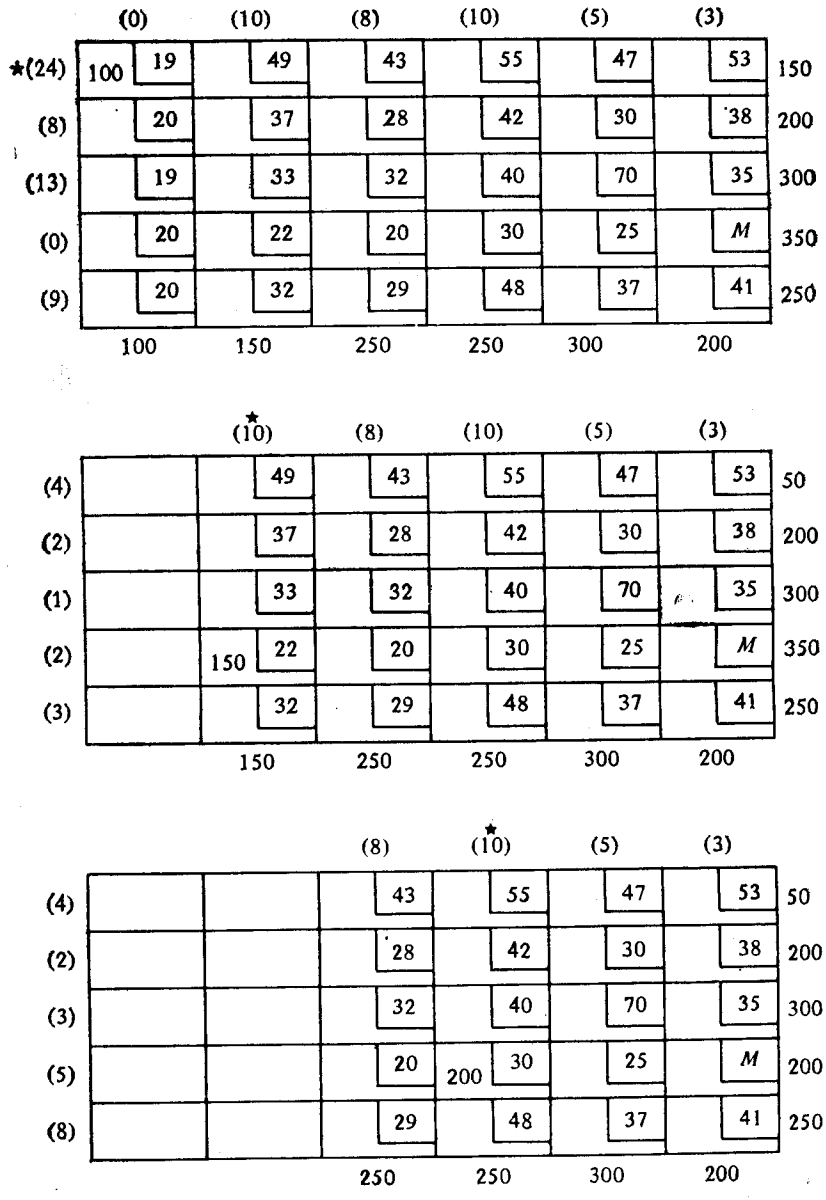
Обратимся к рассмотренному выше примеру. Результаты вычислений, выполняемых при определении начального базисного допустимого решения с помощью ПМФ-алгоритма, приведены в табл. 2.25. В скобках записаны значения штрафов для источников и стоков. Максимальное из этих значений помечается звездочкой. В таблице указаны также величины предложения и спроса, вычисляемые на каждой итерации.

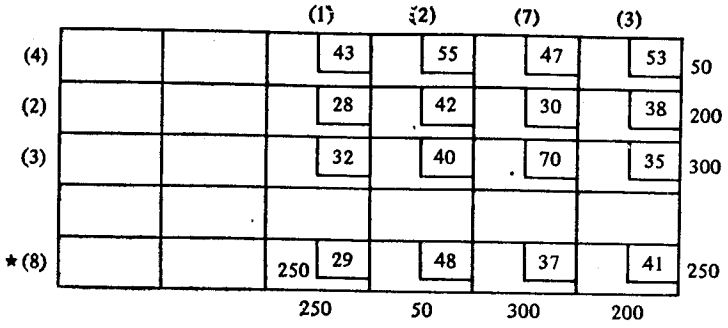
Таким образом, выполняя вычисления по ПМФ-алгоритму (табл. 2.25), мы получили следующее начальное базисное допустимое решение:

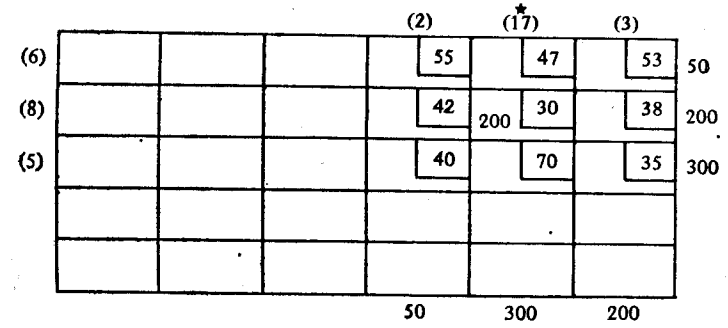
	1	2	3	4	5	6	a_i	
1	100				50		150	
2					200		200	
3				50	50	200	300	
4		150		200			350	
5			250				250	$Z = 39\,300$
b_j	100	150	250	250	300	200		

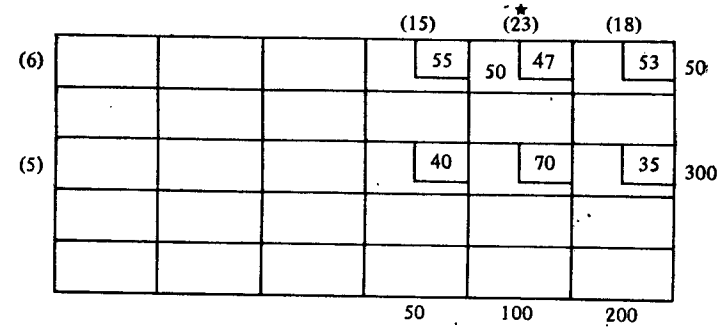
Общие транспортные затраты для схемы перевозки, полученной с помощью метода «северо-западного угла», составляют 40 950, а для схемы, полученной с помощью ПМФ, они составляют 39 300. Это говорит о превосходстве ПМФ над методом «северо-западного угла».

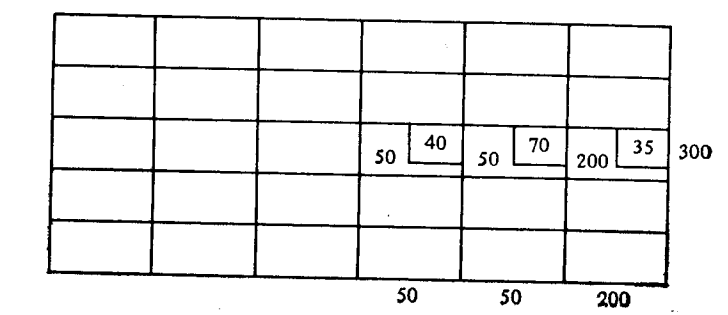
Таблица 2.25. Начальное решение, найденное с помощью ПМФ-алгоритма




(1)
(2)
(7)
(3)
(4)
43
55
47
53
50
(2)
28
42
30
38
200
(3)
32
40
70
35
300
★(8)
250
29
48
37
41
250
250
50
300
200


(2)
(17)
(3)
(6)
55
47
53
50
(8)
42
200
30
38
200
(5)
40
70
35
300
50
300
200


(15)
(23)
(18)
(6)
55
50
47
53
50
(5)
40
70
35
300
50
100
200


50
40
50
70
200
35
300
50
50
200

В рассмотренном примере $m+n-1=10$. Поэтому решение, полученное с помощью ПМФ, является вырожденным, а решение, полученное с помощью метода «северо-западного угла», — невырожденным.

Шаг 2. Определение внебазисного маршрута, который должен быть включен в базис. Поскольку в двойственной задаче число переменных равно $m+n$ и мы должны решить $m+n-1$ уравнений вида $R_i+K_j+c_{ij}=0$, то можно значение R_1 положить равным нулю и искать значения остальных переменных. После того как переменной R_1 присвоено значение 0, могут быть найдены значения переменных K_j, соответствующих столбцам матрицы условий с базисными переменными в первой позиции. Затем можно определить строки с базисными переменными, расположенными в столбцах, соответствующих только что найденным значениям K_j. Для каждой такой строки определяем значение соответствующей ей переменной R_i. Будем повторять описанную процедуру до тех пор, пока не определятся значения всех двойственных переменных. После того как для каждого внебазисного маршрута будет вычислена сумма $R_i+K_j+c_{ij}$, для каждого из них проверяется условие оптимальности. Напомним, что решение является оптимальным, когда все значения $R_i+K_j+c_{ij}\geqslant 0$. Если не все эти значения неотрицательные, то наименьшее отрицательное значение соответствует маршруту с наибольшей возможностью улучшить решение и данный маршрут должен быть включен в базис.

Отметим, что, когда число базисных маршрутов меньше $m+n-1$, значения R_i и K_j не могут быть найдены и поэтому для внебазисных маршрутов нельзя оценить их возможность улучшить решение. Наиболее простой путь определения значений двойственных переменных в данном случае заключается в следующем. Нужно приписать бесконечно малые потоки стольким маршрутам, сколько необходимо для того, чтобы построить невырожденное решение. Эти бесконечно малые потоки должны быть приписаны независимым маршрутам. Независимым маршрутом называется такой маршрут, поток по которому не может быть представлен линейной комбинацией базисных потоков. Каждому маршруту соответствует некоторая клетка таблицы размером $m\times n$, используемой для записи входных данных и решений. Независимому маршруту соответствует клетка, для которой не существует контура, в одном из углов которого расположена данная клетка, а в остальных углах — клетки с положительными потоками.

Для иллюстрации шага 2 рассмотрим начальное базисное допустимое решение, полученное с помощью ПМФ-алгоритма. Как отмечалось выше, данное решение является вырожденным, поскольку число базисных маршрутов меньше $m+n-1=10$.

Таблица 2.26. Правильное расположение ε-клетки

100				50		150
				200		200
			50	50	200	300
	150		200			350
	ϵ	250				250
100	150	250	250	300	200	

В данном случае необходимо ввести один бесконечно малый поток, величину которого мы обозначим через ε (табл. 2.26). Читателю предлагается проверить, что для данной клетки нельзя построить контур, о котором говорилось выше.

Отметим, что расположение ε-клетки в табл. 2.27 неправильное, поскольку маршруты, соответствующие клеткам, расположенным в углах обозначенного контура, не являются независимыми.

Таблица 2.27. Неправильное расположение ε-клетки

100				50		150
				200		200
	ϵ		50	50	200	300
	150		200			350
		250				
100	150	250	250	300	200	

Проверим, является ли оптимальным решение, полученное с помощью ПМФ-алгоритма. В табл. 2.28 символом • обозначены базисные потоки (включая ε-поток). В нижнем левом углу каждой клетки, соответствующей внебазисному маршруту, указано значение $R_i+K_j+c_{ij}$. Значения R_i указаны слева от каждой строки, а значения K_j — над каждым столбцом. Поскольку среди отрицательных величин $R_i+K_j+c_{ij}$ наименьшей является величина $R_4+K_5+c_{45}=-35$ (т. е. маршрут (4, 5) имеет максимальную возможность улучшить решение), то в ре-

зультате выполнения шага 2 внебазисный маршрут, соответствующий клетке (4,5), будет включен в базис.

Шаг 3. Нахождение маршрута, который должен быть исключен из базиса. Определив маршрут, который следует включить в базис (т. е. маршрут, поток по которому возрос с нулевого до некоторого положительного значения), можно построить схему перестройки базисных маршрутов. Данная схема перестройки называется *ступенчатым путем.*

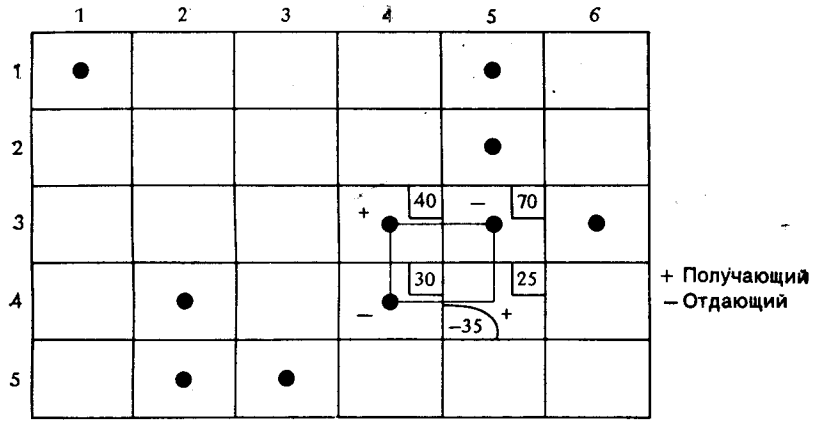
Таблица 2.28. Возможности маршрутов улучшить решение

	$K_1=-19$	$K_2=-9$	$K_3=-6$	$K_4=-17$	$K_5=-47$	$K_6=-12$
$R_1=0$	• 19	49 40	43 37	55 38	• 47	53 41
$R_2=17$	20 18	37 45	28 39	42 42	• 30	38 43
$R_3=-23$	19 -23	33 1	32 3	• 40	• 70	• 35
$R_4=-13$	20 -12	• 22	20 1	• 30	25 -35	M M -25
$R_5=-23$	20 -22	• 32	• 29	48 8	37 -33	41 6

В рассматриваемом примере было показано, что маршрут, соответствующий элементу (4, 5), следует включить в базис. При возрастании потока по данному маршруту необходимо увеличить или уменьшить потоки по некоторым другим маршрутам, с тем чтобы не нарушались исходные ограничения на спрос и предложение и новое решение было бы допустимым. Маршрут, поток по которому возрос, будем называть *получающим,* а маршрут, поток по которому уменьшился, — *отдающим.* Очевидно, что маршрут, который исключается из базиса, должен быть отдающим маршрутом, а поток по нему должен быть наименьшим среди всех текущих базисных потоков. В результате выполнения данной процедуры одна или более базисных переменных станут равными нулю. В последнем случае решение становится вырожденным.

Для нахождения получающих и отдающих маршрутов мы строим ступенчатый путь, который имеет форму контура, состоящего из вертикальных и горизонтальных отрезков, соответствующих выполняемым заменам. В одном из углов данного контура расположена клетка, соответствующая маршруту, включенному в базис, а в остальных углах — клетки, соответствующие базисным маршрутам. Поскольку включенный в базис маршрут является получающим, то соседние с ним маршруты, относящиеся к построенному контуру, могут быть поме-

Таблица 2.29. Ступенчатый путь



чены как отдающие. Остальные маршруты, относящиеся к контуру, попеременно помечаются как получающие (+) и отдающие (—).

В табл. 2.29 показан ступенчатый путь для рассматриваемого примера и указано, как следует изменить потоки, чтобы при включении маршрута, соответствующего клетке (4, 5), в базис решение оставалось допустимым. Отдающим маршрутам соответствуют клетки (3, 5) и (4, 4), а потоки по этим маршрутам равны 50 и 200 соответственно. Поэтому маршрут, соответствующий клетке (3, 5), следует исключить из базиса.

Шаг 4. Корректировка потоков по базисным маршрутам. Величина, на которую мы можем уменьшить или увеличить эти потоки, определяется потоком по маршруту, исключенному из базиса. Для определения новых потоков нужно прибавить эту

Таблица 2.30. Новое базисное допустимое решение (оптимальное)

	1	2	3	4	5	6
1	100				50	
2					200	
3				100		200
4		150		150	50	
5		ϵ	250			

величину к потокам по получающим маршрутам и вычесть эту же величину из потоков по отдающим маршрутам.

При выполнении данной процедуры строится новое базисное решение, для которого не нарушаются ограничения, задаваемые предложением и спросом. Двигаясь по ступенчатому пути, мы просто переносим единицы потока из одних клеток в другие. После выполнения шага 4 в рассматриваемом нами примере получено новое решение, записанное в табл. 2.30.

Теперь следует возвратиться на шаг 2, с тем чтобы проверить, удовлетворяет ли данное решение условию оптимальности. Можно проверить, что решение является оптимальным, и на этом алгоритм завершает работу.

2.10.3. СЕТЕВАЯ ИНТЕРПРЕТАЦИЯ СИМПЛЕКСНОГО АЛГОРИТМА РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ

В настоящем разделе мы покажем, как можно описать симплексный алгоритм для транспортной задачи, пользуясь только сетевыми понятиями. Некоторым читателям, по-видимому, может показаться, что при изложении данного материала мы

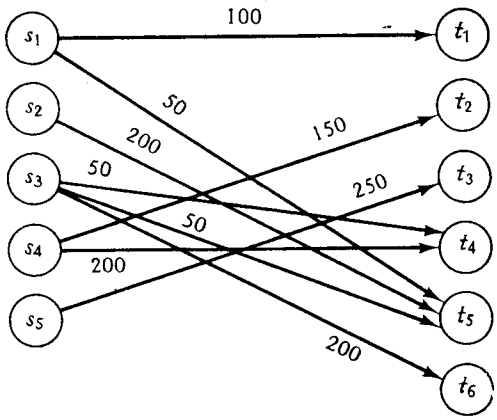


Рис. 2.41. Сетевое представление начального решения, полученного с помощью ПМФ.

больше апеллируем к здравому смыслу, так как симплексный метод будет описан не с помощью стандартных таблиц (разд. 2.10.2), а с помощью сети. При описании алгоритма мы воспользуемся примером из предыдущего раздела. Сеть, изображенная на рис. 2.41, представляет собой начальное решение, полученное с помощью ПМФ. Числа, приписанные дугам сети, равны соответствующим потокам.

Можно показать, что число дуг с положительным потоком равно 9 и что у данной сети две не связанные между собой компоненты. Первая компонента соответствует подсети $G_1 = (\mathbf{N}_1, \mathbf{A}_1)$, где $\mathbf{N}_1 = \{s_1, s_2, s_3, s_4, t_1, t_2, t_4, t_5, t_6\}$ и $\mathbf{A}_1 = \{(s_1, t_1), (s_1, t_5), (s_2, t_5), (s_3, t_4), (s_3, t_5), (s_3, t_6), (s_4, t_2), (s_4, t_4)\}$. Вторая компонента соответствует подсети $G_2 = (\mathbf{N}_2, \mathbf{A}_2)$, где $\mathbf{N}_2 = \{s_5, t_3\}$ и $\mathbf{A}_2 = \{(s_5, t_3)\}$. Для того чтобы сеть, изображенная на

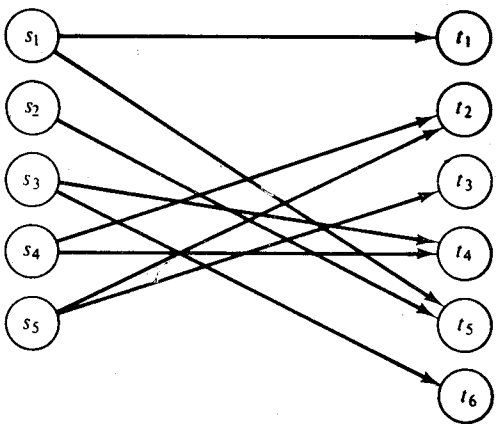


Рис. 2.42. Связная сеть, соответствующая начальному решению, полученному с помощью ПМФ.

рис. 2.41, состояла ровно из одной связной компоненты, необходимо ввести еще одну дугу, соединяющую источник s_5 с любым стоком t_j, $j \neq 3$, или любой источник s_i, $i \neq 5$ со стоком t_3. Как и в разд. 2.10.2, введем дугу (s_5, t_2) с бесконечно малым потоком ε так, что ограничения сетевой модели, задаваемые предложением и спросом, не будут нарушены. Образованная при этом связная сеть изображена на рис. 2.42. Вообще, для того чтобы сеть была связной (состояла из одной компоненты), необходимо $m+n-1$ дуг. Вырожденность эквивалентна тому, что сеть состоит из двух или более не связанных между собой компонент. $m+n-1$ дуг соответствует «независимым клеткам».

При анализе полезности внебазисных маршрутов, проводимом на шаге 2 (разд. 2.10.2), было установлено, что следует использовать маршрут (s_4, t_5). Схема замены, представленная в табл. 2.29 в виде контура, соответствует циклу в рассматриваемой сети (рис. 2.43). Если, передвигаясь по данному циклу, мы будем обходить узлы в последовательности s_4, t_5, s_3, t_4, s_4, то вдоль дуг (s_4, t_5) и (s_3, t_4) мы будем двигаться по направлению, совпадающему с их ориентацией, а вдоль дуг (s_3, t_5) и (s_4, t_4) — по направлению, противоположному их ориентации.

Поэтому потоки по дугам (s_4, t_5) и (s_3, t_4) увеличиваются, а потоки по дугам (s_3, t_5) и (s_4, t_4) уменьшаются. Легко проверить, что максимальная величина потока по дуге (s_4, t_5) рав-

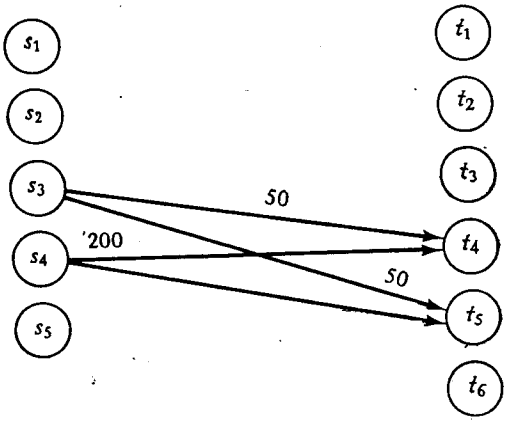


Рис. 2.43. Сетевое представление схемы замены маршрутов в транспортной сети.

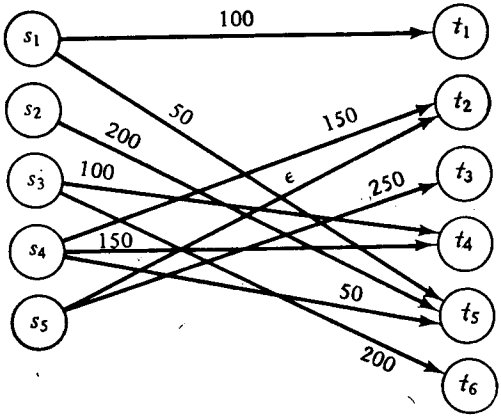


Рис. 2.44. Сетевое представление оптимального решения рассматриваемой транспортной задачи.

на 50. Все это является иллюстрацией основного результата, согласно которому *отдающие клетки* соответствуют дугам в цикле, потоки по которым *уменьшаются*, а *получающие клетки* — дугам, потоки по которым *возрастают*. На рис. 2.44 изображено новое решение, которое, как было показано в разд. 2.10.2, является оптимальным.

2.10.4. МОДЕЛЬ ПРОИЗВОДСТВО — РАСПРЕДЕЛЕНИЕ

Фирма владеет четырьмя заводами, расположенными в различных местах. Каждый завод производит одну и ту же продукцию, причем ее себестоимость, а также цены на сырье для различных заводов не одинаковые. В данном районе имеется пять пунктов сбыта продукции. В каждом из них установлены свои цены на продукцию. Требуется определить оптимальный план производства и распределения продукции для случая, описанного в табл. 2.31.

Таблица 2.31. Стоимостные данные о производстве и распределении продукции[1)]

Завод:	1	2	3	4
Затраты на производство единицы продукции (не включ. стоим.сырья):	20	25	19	20
Стоимость сырья:	10	9	12	8
Транспортные затраты на единицу сырья:				
Пункт сбыта 1	4	10	12	19
Пункт сбыта 2	3	7	5	6
Пункт сбыта 3	5	9	4	7
Пункт сбыта 4	7	5	9	3
Пункт сбыта 5	6	8	7	6

1) Предполагается, что производственные мощности заводов в рассматриваемом периоде планирования составляют 200, 250, 325 и 150 ед. соответственно. Максимальный товарооборот пунктов сбыта составляет 130, 160, 200, 150 и 210 ед. соответственно. Цены на единицу продукции, установленные в каждом из пунктов сбыта, составляют 40, 42, 41 42 и 41 долл. соответственно.

Решение. Следует отметить некоторые особенности данной задачи. Во-первых, в ней требуется максимизировать прибыль, а не минимизировать затраты. Этого можно достичь, если рассматривать прибыль как отрицательные затраты. Во-вторых, суммарная производственная мощность заводов превосходит общий спрос и поэтому необходимо ввести фиктивный пункт сбыта, куда поступала бы избыточная продукция, причем соответствующие транспортные затраты и прибыль равны нулю. В-третьих, если в пункты сбыта перевозить максимально допустимый объем продукции, то некоторые перевозки окажутся убыточными. Поэтому можно ввести фиктивный завод с производственной мощностью x, нулевой прибылью за сбыт продукции и нулевыми транспортными затратами и приписать этому заводу все перевозки, не приносящие прибыли. Если x достаточно большое число, то введение фиктивного завода не повлияет на перевозку продукции от действительных заводов к пунктам сбыта. Величину x можно выбрать равной минимуму между суммарной производственной мощностью заводов и сум-

Таблица 2.32. Матрица условий транспортной задачи для случая максимизации прибыли

		Пункты сбыта						
		W_1	W_2	W_3	W_4	W_5	W_6	
Заводы	P_1	6	9	6	5	5	0	200
	P_2	−4	1	−2	3	−1	0	250
	P_3	−3	6	6	2	3	0	325
	P_4	−7	8	6	11	7	0	150
	P_5	0	0	0	0	0	0	850
		130	160	200	150	210	925	

мой максимальных значений товарооборотов пунктов сбыта. В рассматриваемом примере $x=850$.

В табл. 2.32 приводятся величины прибыли, получаемой за сбыт единицы продукции, и величины предложения и спроса для данной задачи. Фиктивный завод и фиктивный пункт сбыта обозначены через P_5 и W_6 соответственно. Для того чтобы свести данную задачу максимизации к эквивалентной ей задаче минимизации, умножим каждый элемент матрицы условий на —1 и прибавим к нему 11 (максимальный элемент матрицы). Матрица условий соответствующей задачи минимизации приводится в табл. 2.33, где пункты сбыта названы источниками, а заводы — стоками.

Таблица 2.33. Матрица условий транспортной задачи для случая минимизации затрат

		Стоки					
		P_1	P_2	P_3	P_4	P_5	a_i
Источники	W_1	5	15	14	18	11	130
	W_2	2	10	5	3	11	160
	W_3	5	13	5	5	11	200
	W_4	6	8	9	0	11	150
	W_5	6	12	8	4	11	210
	W_6	11	11	11	11	11	925
	b_j	200	250	325	150	850	

С помощью симплексного алгоритма решения транспортной задачи находится оптимальное решение для данной задачи, которое представлено матрицей

$$X = [x_{ij}^*] = \begin{bmatrix} 130 & 0 & 0 & 0 & 0 \\ 70 & 0 & 90 & 0 & 0 \\ 0 & 0 & 200 & 0 & 0 \\ 0 & 0 & 0 & 150 & 0 \\ 0 & 0 & 35 & 0 & 175 \\ 0 & 250 & 0 & 0 & 675 \end{bmatrix}.$$

Прибыль, получаемая при данной схеме перевозки, равна 4905 долл. Из полученного решения следует, что ни из одного действительного источника продукция не поступает в сток P_2, т. е. со второго завода продукция никуда не перевозится.

2.11. ЗАДАЧА О ПЕРЕВОЗКАХ

В транспортной задаче предполагается, что ни в одном маршруте, соединяющем источник с некоторым стоком, другие источники и стоки не могут быть использованы в качестве промежуточных пунктов. Если считать допустимой перевозку груза из источника в сток через другие источники и стоки, то возникает новая задача, которая может быть сведена к обычной транспортной задаче. Безусловно, в данном случае возрастет объем вычислений, поскольку в сеть будут включены дополнительные маршруты, соединяющие каждый источник со всеми другими источниками и каждый сток со всеми другими стоками.

Будем предполагать, что транспортные затраты, соответствующие дополнительным маршрутам, известны. Тогда задача о перевозках может быть сведена к модифицированной транспортной задаче, в которой величины предложения и спроса, соответствующие дополнительным маршрутам, заданы таким образом, что они не влияют на выбор маршрутов, осуществляемый в основном алгоритме. Выполнение последнего требования необходимо по той причине, что ограничения на поток по дополнительным маршрутам, задаваемые предложением и спросом, являются фиктивными и вводятся только для вспомогательных целей.

Пусть f — минимальная из величин Σa_i и Σb_j, т. е. f — это величина реального потока, протекающего по модифицированной сети. Тогда очевидно, что величину «спроса» $\bar{b}_i$ i-го исходного источника и величину «предложения» $\bar{a}_j$ j-го исходного стока можно положить равными f. Иными словами, весь поток может протекать через один источник или через один сток.

Поскольку $\bar{b}_i=\bar{a}_j=f$, то исходные величины предложения и спроса должны быть увеличены на f. Отметим, что если вместо f использовать величину $\bar{f}<f$, то некоторые планы перевозок, являющиеся допустимыми в исходной задаче, станут недопустимыми в новой задаче. В то же время выбор величины $\bar{f}\geqslant f$ не приведет к дополнительным ограничениям на перевозки, не отраженным в исходной задаче. Поэтому наиболее разумным является выбор величины f.

Рассмотрим следующую задачу о перевозках. Предположим, что в транспортной задаче, сформулированной в разд. 2.10.1, каждый источник и каждый сток может использоваться в ка-

честве промежуточного пункта. Исходные данные рассматриваемой транспортной задачи таковы, что $\sum_i a_i = \sum_j b_j = 30$. Следовательно, $f = 30$.

Задача о перевозках может быть сведена к транспортной задаче со следующими величинами предложения и спроса:

1. $\bar{a}_i = a_i + 30$ для исходных источников $i = 1, 2$;
2. $\bar{b}_i = 30$ для исходных источников $i = 1, 2$;
3. $\bar{a}_j = 30$ для исходных стоков $j = 1, 2, 3$;
4. $\bar{b}_j = b_j + 30$ для исходных стоков $j = 1, 2, 3$.

Транспортные затраты, соответствующие дополнительным маршрутам, предполагаются известными. Матрица условий данной транспортной задачи приводится в табл. 2.34.

Таблица 2.34. Матрица условий в задаче о перевозках

		Исходные стоки			Исходные источники		
		1	2	3	1	2	
Исходные источники	1	3	4	7	0	5	50
	2	6	3	2	3	0	40
Исходные стоки	1	0	5	4	2	5	30
	2	9	0	1	3	2	30
	3	2	4	0	2	6	30
		40	42	38	30	30	

2.12. ЗАДАЧА О НАЗНАЧЕНИЯХ

Данная задача заключается в выборе такого распределения ресурсов по некоторым действующим объектам, при котором минимизируются стоимости назначений. Предполагается, что каждый ресурс назначается ровно один раз и каждому объекту приписывается ровно один ресурс. Примеры ресурсов и объектов, а также соответствующие им критерии эффективности, или «стоимости», приводятся в табл. 2.35.

Матрица стоимостей **C** определяется следующим образом: $\mathbf{C} = [c_{ij}]$, где c_{ij} — затраты, связанные с назначением i-го ресурса на j-й объект. Индексы i и j принимают значения 1, 2, ..., m, где m — число объектов или ресурсов.

Определим x_{ij} следующим образом:

$$x_{ij} = \begin{cases} 1, & \text{если } i\text{-й ресурс назначается на } j\text{-й объект,} \\ 0 & \text{в противном случае.} \end{cases}$$

Таблица 2.35. Применения задачи о назначениях

Пример	Ресурсы	Объекты	Критерий эффективности
1	Рабочие	Рабочие места	Время
2	Грузовые автомобили	Маршруты	Затраты
3	Станки	Участки	Объем переработанной продукции
4	Экипажи	Рейсы	Время простоя
5	Коммивояжер	Города	Товарооборот

Таким образом, решение задачи может быть записано в виде двумерного массива $\mathbf{X}=[x_{ij}]$. Допустимое решение называется *назначением*. Для заданного значения m существует $m!$ допустимых решений. Допустимое решение строится путем выбора ровно одного элемента в каждой строке матрицы $\mathbf{X}=[x_{ij}]$ и ровно одного элемента в каждом столбце этой матрицы.

Элементы c_{ij} матрицы **C**, соответствующие элементам $x_{ij}=1$ матрицы **X**, будем помечать кружками. Эти величины c_{ij} выражают затраты, соответствующие допустимому решению **X**. Например, при $m=3$ допустимое решение и соответствующие ему затраты могут быть записаны следующим образом:

$$\mathbf{C}=[c_{ij}]=\begin{bmatrix} 4 & 7 & \textcircled{0} \\ \textcircled{0} & 3 & 8 \\ 6 & \textcircled{3} & 9 \end{bmatrix}, \quad \mathbf{X}=[x_{ij}]=\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

2.12.1. МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ

В данной задаче требуется минимизировать целевую функцию, выражающую общую стоимость назначений. Ограничения можно разбить на две группы. Ограничения первой группы необходимы для того, чтобы каждый ресурс использовался ровно один раз. Ограничения второй группы гарантируют, что каждому объекту будет приписан ровно один ресурс. Математическая постановка задачи выглядит следующим образом:

$$\text{минимизировать } Z=\sum_i \sum_j c_{ij}x_{ij}$$

при условии, что

$$\sum_j x_{ij} = 1 \quad \text{для всех } i,$$

$$\sum_i x_{ij} = 1 \quad \text{для всех } j.$$

$$x_{ij} \geqslant 0 \quad \text{для всех } i \text{ и всех } j.$$

Очевидно, что задача о назначениях является частным случаем транспортной задачи, соответствующим единичным значениям параметров a_i и b_j. Поэтому для решения задачи о назначениях можно воспользоваться любым алгоритмом линейного программирования или симплексным алгоритмом для транспортной задачи. Однако метод, описанный в следующем разделе, является более эффективным, поскольку он использует специфику математической постановки данной задачи.

2.12.2. ВЕНГЕРСКИЙ АЛГОРИТМ

Рассмотрим задачу о назначениях с матрицей стоимостей $\mathbf{C} = [c_{ij}]$. Предположим, что каждый элемент i-й строки складывается с действительным числом γ_i, а каждый элемент j-го столбца — с действительным числом δ_j. В результате такого преобразования матрицы $\mathbf{C}$ будет получена новая матрица стоимостей $\mathbf{D}$, для которой $d_{ij} = c_{ij} + \gamma_i + \delta_j$ или $c_{ij} = d_{ij} - \gamma_i - \delta_j$.

Из последнего равенства следует, что $c_{ij}x_{ij} = d_{ij}x_{ij} - \gamma_i x_{ij} - \delta_j x_{ij}$. Поэтому $\sum\limits_i \sum\limits_j c_{ij}x_{ij} = \sum\limits_i \sum\limits_j d_{ij}x_{ij} - \sum\limits_i \sum\limits_j \gamma_i x_{ij} - \sum\limits_i \sum\limits_j \delta_j x_{ij} = \sum\limits_i \sum\limits_j d_{ij}x_{ij} - \sum\limits_i \gamma_i - \sum\limits_j \delta_j$.

Отсюда следует, что при ограничениях задачи о назначениях минимизация функции $\Sigma\Sigma c_{ij}x_{ij}$ эквивалентна минимизации функции $\Sigma\Sigma d_{ij}x_{ij}$. Основанный на данном результате венгерский алгоритм работает следующим образом: из элементов каждой строки и каждого столбца матрицы стоимостей вычитаются их наименьшие элементы, после чего ведется поиск допустимого решения, единичным элементам которого соответствуют нулевые элементы модифицированной матрицы стоимостей. Если такое допустимое решение существует, то оно является оптимальным назначением. В противном случае матрица стоимостей модифицируется еще раз с целью получить в ней большее число нулевых элементов.

Алгоритм состоит из следующих трех шагов: (1) редукция строк и столбцов; (2) определение назначения; (3) модификация редуцированной матрицы. Ниже дается краткое описание каждого шага.

Шаг 1. *Редукция строк и столбцов.* Цель данного шага состоит в получении максимально возможного числа нулевых элементов в матрице стоимостей. Для этого можно из всех элементов каждой строки вычесть минимальный элемент соответствующей строки, а из всех элементов каждого столбца вычесть минимальный элемент соответствующего столбца. Затем можно исходную матрицу стоимостей заменить редуцированной матрицей стоимостей и перейти к поиску назначения.

Шаг 2. *Определение назначений.* Если после выполнения процедуры редукции в каждой строке и каждом столбце матрицы стоимостей можно выбрать по одному нулевому элементу, так что соответствующее этим элементам решение будет допустимым, то данное назначение будет оптимальным. Если назначения нулевой стоимости для редуцированной матрицы не существует, то данная матрица подлежит дальнейшей модификации (см. шаг. 3).

Шаг 3. *Модификация редуцированной матрицы.* Если нулевых элементов в матрице стоимостей не достаточно для того, чтобы построить назначение нулевой стоимости, то с помощью следующей простой процедуры можно получить новые нулевые элементы. Определим для редуцированной матрицы стоимостей минимальное множество строк и столбцов, содержащих все нулевые элементы, и найдем минимальный элемент вне данного множества. Если значение этого элемента вычесть из всех остальных элементов матрицы, то на месте нулей будут стоять отрицательные величины и по крайней мере один элемент, не принадлежащий выделенному множеству строк и столбцов, станет равным нулю. Однако теперь назначение нулевой стоимости может не быть оптимальным, поскольку матрица содержит отрицательные элементы.

Для того чтобы матрица не содержала отрицательных элементов, прибавим абсолютную величину наименьшего отрицательного элемента ко всем элементам выделенных строк и столбцов. Отметим, что к элементам, расположенным на пересечении выделенных строк и столбцов, данная величина будет прибавляться дважды. Кроме того, как и раньше, все отрицательные элементы будут преобразованы в нулевые или положительные элементы.

В результате выполнения данной процедуры новая редуцированная матрица стала содержать больше нулей, расположенных вне строк и столбцов, соответствующих нулевым элементам текущего неоптимального решения.

Основную процедуру венгерского алгоритма, который был только что описан, можно свести к следующим двум наборам процедур. Процедуры первого набора могут быть использованы для поиска назначения, которому соответствуют нулевые

элементы редуцированной матрицы стоимостей. Второй набор процедур может быть использован для модификации редуцированной матрицы стоимостей.

1. Процедуры, используемые для поиска допустимого решения нулевой стоимости (шаг. 2).

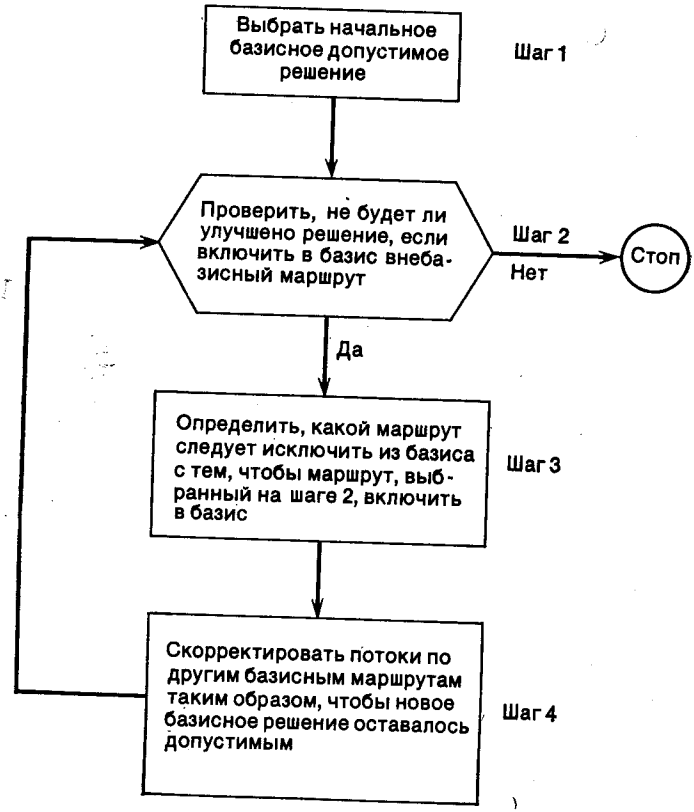


Рис. 2.45. Блок-схема венгерского алгоритма.

а. Найти строки, содержащие ровно один невычеркнутый нулевой элемент. В каждой такой строке произвести назначение, соответствующее невычеркнутому нулевому элементу. В каждом столбце, в котором было произведено назначение, вычеркнуть все невычеркнутые ранее нулевые элементы. Строки рассматриваются в порядке возрастания их номеров.

б. Найти столбцы, содержащие ровно один невычеркнутый нулевой элемент. В каждом таком столбце произвести назначение, соответствующее невычеркнутому нулевому элементу. В каждой строке, в которой было произведено назначение, вы-

черкнуть все невычеркнутые ранее нулевые элементы. Столбцы рассматриваются в порядке возрастания их номеров.

в. Выполнять шаги «а» и «б» до тех пор, пока не будет вычеркнуто максимально возможное число нулевых элементов. Если построенное назначение полное, то оно является оптимальным. Если некоторые нули остались невычеркнутыми, то можно попытаться найти полное назначение (например, методом проб и ошибок) среди нескольких оптимальных вариантов. Если нельзя найти ни одного полного назначения, то необходима дальнейшая модификация матрицы.

2. Процедуры, используемые для модификации редуцированной матрицы (шаг 3).

а. Вычислить число нулей в каждой невычеркнутой строке и каждом невычеркнутом столбце.

б. Вычеркнуть строку или столбец с максимальным числом нулей. В случае равенства числа нулей в нескольких строках и столбцах вычеркнуть любую из этих строк (или любой из этих столбцов). Строки вычеркиваются горизонтальными линиями, а столбцы— вертикальными линиями.

в. Выполнять шаги «а» и «б» до тех пор, пока не будут вычеркнуты все нули.

г. Из всех невычеркнутых элементов вычесть минимальный невычеркнутый элемент и прибавить его к каждому элементу, расположенному на пересечении двух линий.

На рис. 2.45 изображена блок-схема венгерского алгоритма. На шаге 3 используются описанные выше процедуры. Отметим, что в данных процедурах явно не используются отрицательные элементы, о которых говорилось при обосновании алгоритма.

2.12.3. РЕШЕНИЕ ПРИМЕРА С ПОМОЩЬЮ ВЕНГЕРСКОГО АЛГОРИТМА

Покажем работу венгерского алгоритма на примере задачи о назначениях со следующей матрицей стоимостей:

$$\mathbf{C} = [c_{ij}] = \begin{bmatrix} 2 & 10 & 9 & 7 \\ 15 & 4 & 14 & 8 \\ 13 & 14 & 16 & 11 \\ 4 & 15 & 13 & 19 \end{bmatrix}$$

1. Редукция строк и столбцов.

а. Значения минимальных элементов строк 1, 2, 3 и 4 равны 2, 4, 11 и 4 соответственно. Вычитая из элементов каждой

строки соответствующее минимальное значение, получим следующую матрицу:

0	8	2	5
11	0	5	4
2	3	0	0
0	11	4	15

б. Значения минимальных элементов столбцов 1, 2, 3 и 4 равны 0, 0, 5 и 0 соответственно. Вычитая из элементов каждого столбца соответствующее минимальное значение, получим следующую матрицу:

0	8	7	5
11	0	10	4
2	3	5	0
0	11	9	15

2. Поиск допустимого решения, для которого все назначения имеют нулевую стоимость.

а. Строки 1, 2 и 4 содержат по одному невычеркнутому нулю. Рассматривая эти строки в порядке возрастания их номеров, произведем вначале назначение, соответствующее элементу (1, 1), и вычеркнем нулевой элемент (4, 1). Затем произведем назначение, соответствующее элементу (2, 2). Строка 4 не может быть использована, поскольку нулевой элемент (4, 1) был вычеркнут после того, как мы произвели назначение, соответствующее элементу (1, 1).

б. Столбцы 3 и 4 содержат по одному невычеркнутому нулю. Рассматривая эти столбцы в порядке возрастания их номеров, мы можем произвести третье назначение, соответствующее элементу (3, 3). В столбце 4 ни одно назначение невозможно, так как расположенные в нем нулевые элементы были вычеркнуты после того, как мы произвели назначение, соответствующее элементу (3, 3). После выполнения данного шага матрица стоимостей имеет следующий вид:

⓪	8	2	5
11	⓪	5	4
2	3	⓪	0
0	11	4	15

Таким образом, ни одно полное назначение не может быть получено, и необходимо провести дальнейшую модификацию редуцированной матрицы стоимостей.

3. Модификация редуцированной матрицы стоимостей.

а. Число нулей в строках 1, 2, 3 и 4 матрицы, полученной на предыдущем шаге, равно 1, 1, 2 и 1 соответственно. Для столбцов соответствующие величины равны 2, 1, 1 и 1.

б. Максимальное число нулей, по два, содержат строка 3 и столбец 1. Выбираем строку 3 и вычеркиваем все ее элементы горизонтальной линией.

в. Число невычеркнутых нулей в строках 1, 2 и 4 равно 1, 1 и 1 соответственно. Для столбцов соответствующие значения равны 2, 1, 0 и 0. Поэтому мы должны выбрать столбец 1 и вычеркнуть его вертикальной линией. После этого останется только один невычеркнутый нуль — элемент (2, 2). Поэтому можно вычеркнуть либо строку 2, либо столбец 2. Вычеркивая строку 2 горизонтальной линией, получаем следующую матрицу:

	8	2	5
	11	4	15

г. Значение минимального невычеркнутого элемента равно 2. Вычитая его из всех невычеркнутых элементов и складывая его со всеми элементами, расположенными на пересечении двух линий, получаем новую редуцированную матрицу стоимостей:

0	6	0	3
13	0	5	4
4	3	0	0
0	9	2	13

Выполняя вновь процедуру построения допустимого решения нулевой стоимости, получаем следующее оптимальное решение:

0	6	⓪	3
13	⓪	5	4
4	3	0	⓪
⓪	9	2	13

Результаты вычислений, проведенных в данном разделе, отражены на рис. 2.46.

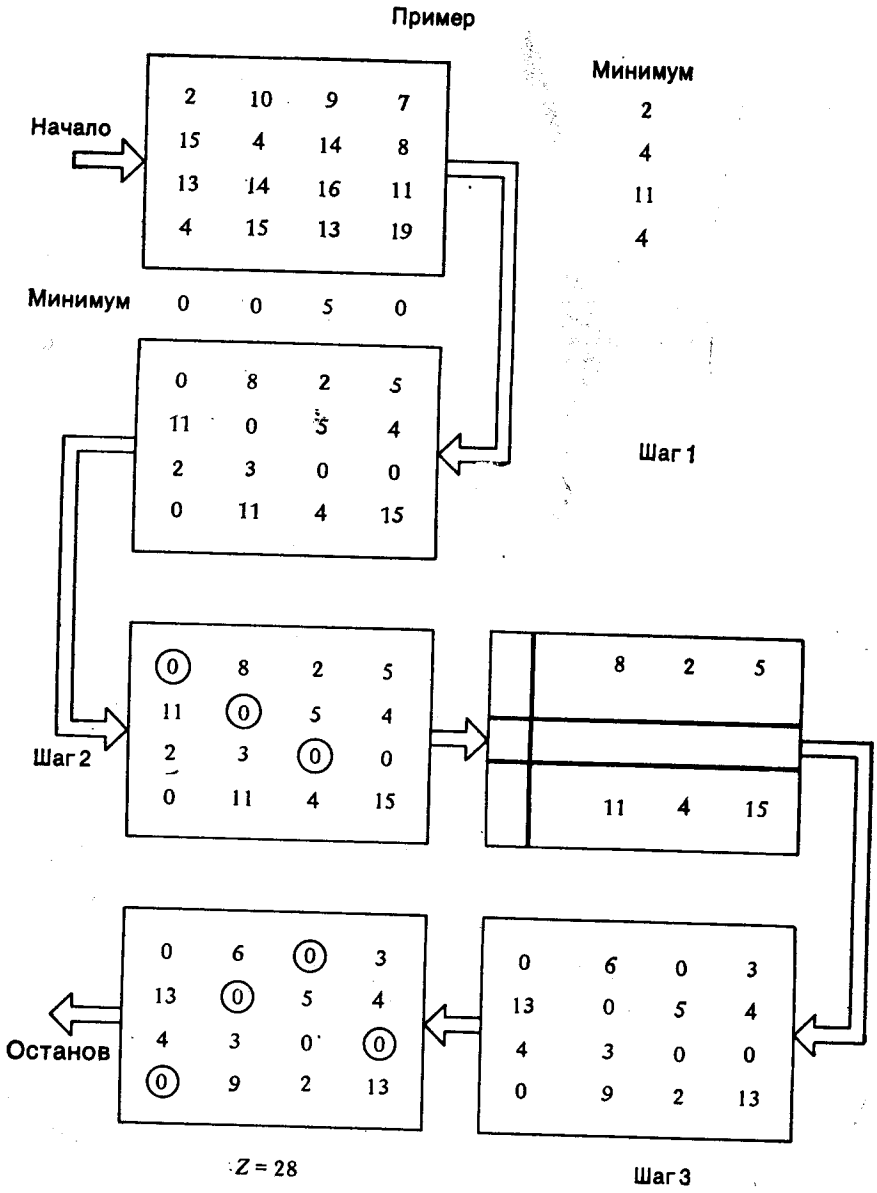


Рис. 2.46. Шаги вычислений в венгерском алгоритме.

2.12.4. ЗАМЕЧАНИЯ

1. Число допустимых назначений равно $m!$, где m — число строк или столбцов матрицы стоимостей (матрица является квадратной).

2. Для любого недопустимого назначения соответствующая ему стоимость полагается равной M (достаточно большому числу в минимизационных задачах).

3. Если исходная матрица не является квадратной, то следует ввести нужное количество строк или столбцов и их элементам присвоить значения, определяемые условиями решаемой задачи.

4. Если исходная задача является задачей максимизации, то все элементы матрицы стоимостей следует умножить на —1 и сложить их с достаточно большим числом так, чтобы матрица не содержала бы отрицательных элементов. Затем задачу следует решать как задачу минимизации.

5. Если число линий, необходимое для того чтобы вычеркнуть нулевые элементы, равно числу строк или столбцов (квадратной матрицы), то существует назначение нулевой стоимости.

2.12.5. ЗАДАЧА РАЗМЕЩЕНИЯ ПРОИЗВОДСТВА

Компания разрабатывает план выпуска трех новых видов продукции. Предположим, что компания владеет пятью предприятиями и что на трех из них должны производиться новые виды продукции — по одному виду на одно предприятие. Ниже указаны издержки производства и сбыта единицы продукции, соответствующие каждой паре вид продукции — предприятие.

1. Издержки производства единицы продукции (долл.):

		Предприятие				
		1	2	3	4	5
Вид продукции	1	20	23	38	15	35
	2	8	29	6	35	35
	3	5	8	3	4	7

2. Издержки сбыта единицы продукции (долл.):

		Предприятие				
		1	2	3	4	5
Вид продукции	1	20	50	20	10	13
	2	7	90	8	35	60
	3	5	5	4	15	6

Плановый объем годового производства, который позволил бы удовлетворить спрос, и плановая стоимость единицы продукции каждого вида следующие:

Вид продукции	Плановый объем производства	Плановая стоимость (в долл.)
1	35 000	55
2	160 000	50
3	54 000	30

Решение. Общие издержки на единицу продукции складываются из издержек производства и издержек сбыта единицы продукции. Поскольку продажная цена единицы каждого вида продукции известна, то для каждой пары вид продукции — предприятие можно вычислить прибыль на единицу продукции:

		Предприятие				
		1	2	3	4	5
Вид продукции	1	15	−18	−3	30	7
	2	35	−69	36	−20	−45
	3	20	17	23	11	17

Умножая прибыль, приходящуюся на единицу продукции, на годовой объем сбыта, можно получить общую годовую прибыль, соответствующую каждой паре вид продукции — предприятие. Данные величины (в тыс долл.) приведены в следующей таблице:

		Предприятие				
		1	2	3	4	5
Вид продукции	1	525	−630	−105	1050	245
	2	5600	−11 040	5760	−3200	−7200
	3	1080	918	1242	594	918

Если прибыль рассматривать как отрицательные затраты и ввести два вида фиктивной продукции (4 и 5), которой соответ-

ствует нулевая прибыль, то исходная задача максимизации может быть сведена к минимизационной задачи о назначениях. Для того чтобы матрица стоимостей не содержала отрицательных элементов, сложим каждый элемент матрицы с числом 5760. В результате будет получена следующая матрица:

Вид продукции	Предприятие 1	2	3	4	5
1	5235	6390	5865	4710	5515
2	160	16800	0	8960	12960
3	4680	4842	4518	5166	4842
4	0	0	0	0	0
5	0	0	0	0	0

Оптимальное решение данной задачи следующее: производство первого вида продукции назначается предприятию 4, второго вида — предприятию 1, третьего вида — предприятию 3, четвертого вида — предприятию 2 и пятого вида — предприятию 5. Очевидно, что два последних назначения являются фиктивными. Суммарная годовая прибыль, соответствующая данному решению, равна 7 892 000 долл.

Вид продукции	Предприятие 1	2	3	4	5
1	365	1520	1155	⓪	645
2	⓪	16640	0	8960	12800
3	2	164	⓪	648	164
4	0	⓪	160	160	0
5	0	0	160	160	⓪

Это решение получается так: модифицируем редуцированную матрицу, вычеркивая в ней третий и четвертый столбцы и четвертую и пятую строки.

2.13. ЗАДАЧА О НАЗНАЧЕНИЯХ И ЗАДАЧА КОММИВОЯЖЕРА

В разд. 2.9 был описан алгоритм решения задачи коммивояжера. Алгоритм решения задачи о назначениях, описанный в разд. 2.12, также может быть использован в качестве составной части специальной программы решения задачи коммивояжера.

Очевидно, что в общем случае оптимальное решение (решение минимальной стоимости) $(n \times n)$-задачи о назначениях может не быть решением соответствующей $(n \times n)$-задачи коммивояжера. Однако алгоритм решения задачи о назначениях может быть соответствующим образом модифицирован и использован для решения более сложной задачи.

Пусть **D** — матрица расстояний (стоимостей) размером $n \times n$, элементы d_{ij} которой представляют собой штраф за переезд из города, или пункта, i в другой город, или пункт, j, где $i=1, 2, ..., n$, $j=1, 2, ..., n$.

Преобразуем далее матрицу **D**, полагая $d_{ij}=0$ при $i \geqslant j$ и оставляя без изменения элементы d_{ij} (которые могут быть положительными, отрицательными или равными нулю) при $i<j$ (табл. 2.36). Данная матрица расстояний (стоимостей) называется *верхней треугольной* матрицей. Будем обозначать ее через $\overline{\mathbf{D}}$.

Таблица 2.36. Верхняя треугольная матрица

Из (i) \ В (j)	1	2	3	...	n
1	0	d_{12}	d_{13}	...	d_{1n}
2	0	0	d_{23}	...	d_{2n}
3	0		0	...	d_{3n}
⋮	⋮	⋮	⋮		⋮
n	0	0	0	...	0

$= \overline{D}$

Предлагаемый алгоритм решения задачи коммивояжера, разработанный Лоулером [40], состоит в следующем.

Шаг 1. Исключить первый столбец и n-ю строку из матрицы $\overline{\mathbf{D}}$ и решить соответствующую $(n-1) \times (n-1)$-задачу о назначениях.

Замечание. В результате выполнения шага 1 будет построен путь из узла 1 в узел n. Возможно, будут построены также и другие *непересекающиеся замкнутые подпути*. Эти подпути содержат узлы, не принадлежащие главному пути из узла 1 в узел n. Если в результате работы алгоритма решения задачи о назначениях будет построен путь или цепь, не содержащий циклов и проходящий через каждый узел ровно один раз, то решение исходной

задачи будет найдено. Поэтому мы рассмотрим случай, когда существуют непересекающиеся замкнутые подпути.

Перед тем как продолжить описание алгоритма, отметим, что $d_{ij}=0$ для всех дуг (i, j), для которых $i \geqslant j$. Такие дуги будем называть *недействительными*. Путь из узла 1 в узел n, а также все непересекающиеся замкнутые подпути могут содержать одну или несколько недействительных дуг.

Шаг 2. Последовательно рассмотреть все непересекающиеся замкнутые подпути, полученные в результате решения задачи о назначениях, и удалить из них все недействительные дуги.

Замечание. Поскольку в результате выполнения данного шага недействительные дуги будут удалены из всех замкнутых подпутей, то последние будут содержать дуги, для каждой из которых номер конечного узла больше номера начального узла. (Напомним, что матрица расстояний является верхней треугольной матрицей.) Обозначим эти подпути символами i_1—j_1, i_2—j_2, i_3—j_3 и т. д., где $i_1<i_2<\ldots<i_m$ и $i_1 \leqslant j_1$, $i_2 \leqslant j_2$, ..., $i_m \leqslant j_m$.

Шаг 3. Вводя новые недействительные дуги, соединить узел n (конечный узел пути из узла 1 в узел n) с узлом i_m, j_m — с i_{m-1}, ..., j_2 — с i_1 и узел j_1 — с узлом 1.

Замечание. В результате выполнения шага 3 строится цикл, ведущий из узла 1 в тот же самый узел и проходящий через каждый из оставшихся n—1 узлов равно один раз. Отметим, что поскольку мы добавляем только недействительные дуги, то значение целевой функции в задаче коммивояжера точно такое же, как и в задаче о назначениях. А поскольку решение задачи о назначениях является оптимальным (т. е. стоимость полученного назначения минимальная), то и решение задачи коммивояжера — оптимальное.

Для иллюстрации данного алгоритма рассмотрим следующий пример, принадлежащий Лоулеру [40].

Матрица расстояний

Из \ В	1	2	3	4	5	6	7
1	0	–1	7	–20	3	2	5
2	0	0	12	8	16	9	8
3	0	0	0	3	7	6	2
4	0	0	0	0	4	4	9
5	0	0	0	0	0	–18	–1
6	0	0	0	0	0	0	3
7	0	0	0	0	0	0	0

$=\bar{D}$

Шаг 1. Обведенные элементы данной матрицы соответствуют оптимальному решению задачи о назначениях.

Редуцированная матрица в задаче о назначениях

	2	3	4	5	6	7
1	–1	7	(–20)	3	–2	5
2	(0)	12	8	16	9	8
3	0	0	3	7	6	(2)
4	0	(0)	0	4	4	9
5	0	0	0	0	(–18)	–1
6	0	0	0	(0)	0	3

$= \bar{D}$

Шаг 2. Путь из узла 1 в узел 7 следующий:

$$①\xrightarrow{-20}④\xrightarrow{0}③\xrightarrow{2}⑦$$

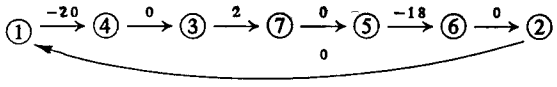
Имеются два непересекающихся замкнутых подпути:

1. $②\xrightarrow{0}②$
2. $⑤\xrightarrow{-18}⑥\xrightarrow{0}⑤$

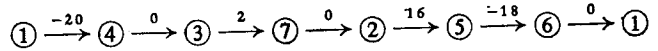
Поскольку дуги (2, 2) и (6, 5) являются недействительными, то исключим их из построенных подпутей. В результате мы получим следующие непересекающиеся множества:

$$①\xrightarrow{-20}④\xrightarrow{0}③\xrightarrow{2}⑦$$
$$②$$
$$⑤\xrightarrow{-18}⑥$$

Шаг 3. Используя недействительные дуги, соединяем не связанные между собой узлы (пути) следующим образом:



Стоимость данного пути, представляющего решение задачи коммивояжера, равна —36. В заключение отметим, что следующее решение не является оптимальным, так как в последовательность непересекающихся подпутей включена дуга (2, 5), не являющаяся недействительной:



2.14. ЗАДАЧА О МАКСИМАЛЬНОМ ПОТОКЕ

Пусть $\mathbf{G}=(\mathbf{N}, \mathbf{A})$ — ориентированная сеть с одним источником $s \in \mathbf{N}$ и одним стоком $t \in \mathbf{N}$, и пусть дуги $(i, j) \in \mathbf{A}$ имеют ограниченную пропускную способность. Задача о максимальном потоке заключается в поиске таких потоков по дугам, принадлежащим множеству **A**, что результирующий поток, протекающий из источника s в сток t, является максимальным. Предполагается, что в источник может поступать неограниченный поток и что для каждого промежуточного узла сети выполняется условие сохранения потока. Данная задача является нетривиальной, когда пропускная способность U_{ij} каждой дуги представляет собой конечную верхнюю границу потока f_{ij} по этой дуге.

В разд. 2.2 задача о максимальном потоке была сформулирована в виде задачи линейного программирования, поэтому для ее решения можно воспользоваться обычным симплексным методом. В настоящем разделе будет описана еще одна, более эффективная процедура поиска решения данной задачи. Алгоритм начинает работу с некоторого допустимого решения. Затем выполняется процедура расстановки пометок, разработанная Фордом и Фалкерсоном [15], с помощью которой определяется другой допустимый поток большей величины. В данном алгоритме узлы рассматриваются как промежуточные пункты передачи потока, а дуги — как распределительные каналы. Для формального описания алгоритма необходимо ввести два основных понятия — пометки и аугментального пути[1)] потока.

Пометка узла используется для указания как величины потока, так и источника потока, вызывающего изменение текущей величины потока по дуге, соединяющей этот источник с рассматриваемым узлом. Если q_j единиц потока посылается из узла i в узел j и вызывает увеличение потока по этой дуге, то будем говорить, что узел j помечается из узла i символом

[1)] Аугментальный путь называют также увеличивающим путем. — *Прим. перев.*

$+q_j$. В данном случае узлу j приписывается пометка $[+q_j, i]$. Аналогично если посылка q_j единиц потока вызывает уменьшение потока по дуге, то будем говорить, что узел j помечается из узла i символом $-q_j$. В данном случае узлу j приписывается пометка $[-q_j, i]$.

Текущий поток из узла i в узел j увеличивается, когда q_j единиц дополнительного потока посылается в узел j по ориентированной дуге (i, j) в направлении, совпадающем с ее ориентацией. В данном случае дуга (i, j) называется *прямой*. Соответствующий пример показан на рис. 2.47.

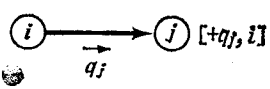


Рис. 2.47. Если узел j помечается из узла i, то поток по дуге (i, j) увеличивается, поскольку дуга (i, j) является прямой.

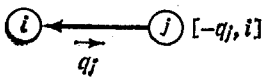


Рис. 2.48. Если узел j помечается из узла i, то поток по дуге (j, i) уменьшается, поскольку дуга (j, i) является обратной.

Текущий поток из j в i уменьшается, когда q_j единиц потока посылается в узел j по ориентированной дуге (j, i) в направлении, противоположном ее ориентации. В этом случае дуга (j, i) называется *обратной*. Соответствующий пример показан на рис. 2.48.

Если узел j помечается из узла i и дуга (i, j) прямая, то поток по данной дуге увеличивается и величина, соответствующая оставшейся неиспользованной пропускной способности дуги, должна быть нужным образом скорректирована. Данную величину обычно называют остаточной пропускной способностью дуги. Читатель может легко установить, что если некоторому узлу приписывается пометка и при этом используется прямая ветвь, то она может иметь только положительную «остаточную пропускную способность». Кроме того, узел j может быть помечен из узла i только после того, как узлу i приписана пометка.

Аугментальный путь потока из s в t определяется как связная последовательность прямых и обратных дуг, по которым из s в t можно послать несколько единиц потока. Поток по каждой прямой дуге увеличивается, не превышая при этом ее пропускной способности, а поток по каждой обратной дуге уменьшается, оставаясь при этом неотрицательным. Аугментальный путь потока используется для выбора такого способа изменения потока, при котором поток в узле t увеличивается и при этом для каждого внутреннего узла сети не будет нарушено условие сохранения потока.

2.14.1. ПРОЦЕДУРА РАССТАНОВКИ ПОМЕТОК ДЛЯ ЗАДАЧИ О МАКСИМАЛЬНОМ ПОТОКЕ

Задача о максимальном потоке часто встречается на практике, причем число узлов и дуг в сети нередко достигает нескольких тысяч. Поэтому для решения таких задач необходимо использовать эффективную процедуру вычислений. Благодаря простоте постановки задачи о максимальном потоке был разработан эффективный рекуррентный алгоритм поиска оптимального решения (максимального потока), использующий процедуру расстановки пометок. Перейдем к описанию этого алгоритма.

Пусть (i, j) — ориентированная дуга, ведущая из узла i в узел j. В каких случаях поток по данной дуге может быть *увеличен*? Как отмечалось выше, поток можно увеличить на q_j единиц, если дуга (i, j) является прямой и узлу j приписывается пометка $[+q_j, i]$. Остается выяснить, когда это имеет место. Предположим, что дуге (i, j) уже приписан поток $f_{ij} \geqslant 0$ $(f_{ij} \leqslant U_{ij})$. Очевидно, что величина q_j не может превосходить остаточной пропускной способности $U_{ij} - f_{ij}$. Достаточно ли этого, для того чтобы пометить узел j? Нет, поскольку из узла i не всегда можно получить $U_{ij} - f_{ij}$ единиц потока. Отметим, что в узел j можно послать столько единиц потока, сколько их добавляется в узел i, т. е. самое большее q_i. Следовательно, поток по прямой дуге (i, j) можно увеличить на величину q_j, где $q_j = \min[q_i, U_{ij} - f_{ij}]$.

Точно так же можно пометить узел j, если дуга (j, i) является обратной. Здесь возникает следующий вопрос: возможно ли уменьшение потока по дуге (j, i)? Очевидно, что это возможно только в том случае, когда $f_{ji} > 0$. На сколько этот поток может быть уменьшен? Самое большее — на число единиц потока, которое можно взять из узла i, т. е. на величину q_i. Следовательно, поток по обратной дуге (j, i) может быть уменьшен на величину q_j, где $q_j = \min[q_i, f_{ji}]$.

Алгоритм расстановки пометок работает следующим образом. Вначале источнику приписывается пометка $[\infty, -]$, указывающая на то, что из данного узла может вытекать поток бесконечно большой величины. Далее мы ищем аугментальный путь потока от источника к стоку, проходящий через помеченные узлы. Все узлы, отличные от источника, в начальный момент не помечены. Пытаясь достичь стока, мы проходим по прямым и обратным дугам и последовательно помечаем принадлежащие им узлы. Возможны два следующих случая:

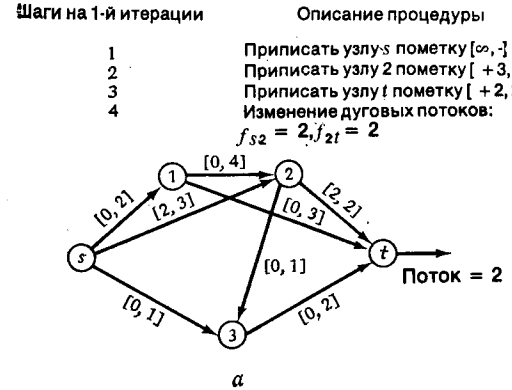
1. Стоку t приписана пометка $[+q_t, k]$. В этом случае аугментальный путь потока найден и поток по каждой дуге этого пути может быть увеличен или уменьшен на величину q_t. По-

сле изменения дуговых потоков текущие пометки стираются и вся описанная выше процедура выполняется заново.

2. Сток t не может быть помечен. Это означает, что аугментальный путь потока не может быть найден. Следовательно, построенные дуговые потоки образуют оптимальное решение (максимальный поток).

2.14.2. ПРИМЕР РАБОТЫ АЛГОРИТМА РАССТАНОВКИ ПОМЕТОК

Для иллюстрации работы алгоритма расстановки пометок для сети, изображенной на рис. 2.49, *а*, будет найден максимальный поток, который может протекать из узла s в узел t. Каждой дуге (i, j) приписывается пометка $[f_{ij}, U_{ij}]$, где f_{ij} — те-



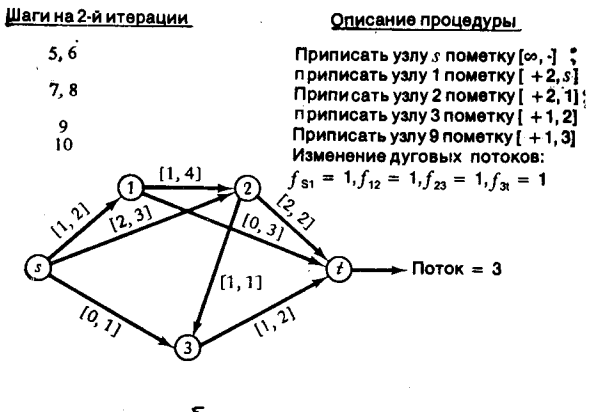


Рис. 2.49. Пример работы алгоритма расстановки пометок.
а — первая итерация; *б* — вторая итерация; *в* — третья итерация; *г* — четвертая итерация; *д* — пятая итерация.

Шаги на 3-й итерации

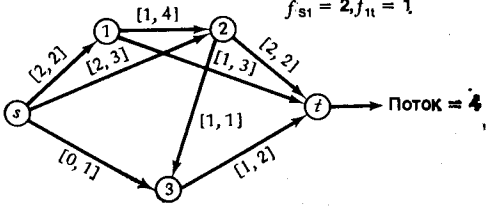
11, 12

13

14

Описание процедуры

Приписать узлу s пометку $[\infty, -]$;
приписать узлу 1 пометку $[+1, s]$
Приписать узлу t пометку $[+1, 1]$
Изменение дуговых потоков:
$f_{s1} = 2, f_{1t} = 1$



в

Шаги на 4-й итерации

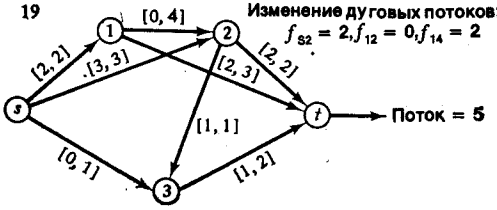
15, 16, 17

18

19

Описание процедуры

Приписать узлу s пометку $[\infty, -]$;
приписать узлу 2 пометку $[+1, s]$;
приписать узлу 1 пометку $[-1; 2]$
Приписать узлу t пометку $[+1, 1]$
Изменение дуговых потоков:
$f_{s2} = 2, f_{12} = 0, f_{14} = 2$



г

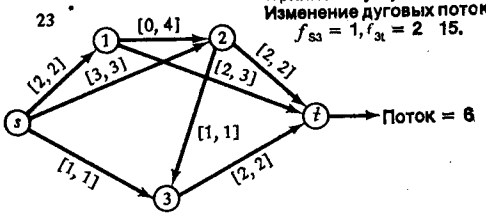
Шаги на 5-й итерации

20, 21, 22

23

Описание процедуры

Приписать узлу s пометку $[\infty, -]$;
приписать узлу 3 пометку $[+1, s]$;
приписать узлу t пометку $[+1, 3]$
Изменение дуговых потоков:
$f_{s3} = 1, f_{3t} = 2$ 15.



д

Шаги на 6-й итерации

24

25

Описание процедуры

Приписать узлу s пометку $[\infty, -]$
Ни один из узлов не может быть помечен, поэтому максимальный поток равен 6

Рис. 2.49 (продолжение).

кущее значение дугового потока, а U_{ij} — пропускная способность дуги. Вначале величины всех дуговых потоков полагаются равными нулю. При выполнении каждой итерации нужно пометить сток t. Эта задача решается в 6 итерациях, результаты каждой из которых приводятся на рис. 2.49.

2.14.3. ТЕОРЕМА О МАКСИМАЛЬНОМ ПОТОКЕ И МИНИМАЛЬНОМ РАЗРЕЗЕ

Пусть задана сеть $G=(\mathbf{N}, \mathbf{A})$. Разобьем множество узлов $\mathbf{N}$ на два непересекающихся подмножества $\mathbf{N}_c$ и $\overline{\mathbf{N}}_c$. Эти два подмножества соединены между собой дугами, образующими множество $\mathbf{A}_c$. Множество всех оставшихся дуг обозначим через $\overline{\mathbf{A}}_c$. Предположим далее, что сток t принадлежит подмножеству $\overline{\mathbf{N}}_c$, а источник s — подмножеству $\mathbf{N}_c$. Тогда величина любого потока из $\mathbf{N}_c$ в $\overline{\mathbf{N}}_c$, протекающего по дугам из множества $\mathbf{A}_c$, не может быть больше, чем сумма пропускных способностей всех дуг из $\mathbf{A}_c$, т. е.

$$\sum f_{ij} \leqslant \sum_{\substack{i \in \mathbf{N}_c \\ j \in \overline{\mathbf{N}}_c}} \mathbf{U}_{ij}.$$

Этот «барьер для потока», отделяющий множество $\mathbf{N}_c$ от $\overline{\mathbf{N}}_c$, будем называть «разрезом» и обозначать его через $(\mathbf{N}_c, \overline{\mathbf{N}}_c)$. Очевидно, что величина максимального потока, который может протекать из узла s в узел t, ограничена сверху величиной этого разреза. Величина разреза $(\mathbf{N}_c, \overline{\mathbf{N}}_c)$ равна сумме пропускных способностей всех дуг из множества $\mathbf{A}_c$, по которым поток может протекать из $\mathbf{N}_c$ в $\overline{\mathbf{N}}_c$. Согласно теореме о максимальном потоке и минимальном разрезе, величина максимального потока из узла s в узел t равна величине минимального разреза, отделяющего узел s от узла t.

Для иллюстрации теоремы о максимальном потоке и минимальном разрезе рассмотрим сеть, изображенную на рис. 2.50. Отметим, что существует несколько разрезов, отделяющих узел 6 от узла 1, и что величина максимального потока равна 8. Пропускные способности разрезов 1, 2 и 3, изображенных на рис. 2.50, равны 9, 12 и 8 соответственно. Поэтому разрез 3 является минимальным.

Значение теоремы о максимальном потоке и минимальном разрезе заключается в том, что максимальный поток в сети с ограниченной пропускной способностью можно находить, вычисляя пропускные способности всех разрезов и выбирая среди полученных значений минимальное. Конечно, при решении зада-

чи о максимальном потоке этот результат имеет небольшое практическое значение, поскольку мы не получаем никакой информации о самих потоках f_{ij} по дугам. Однако данный результат важен с теоретической точки зрения и часто используется при разработке сложных потоковых алгоритмов или при анализе решения на оптимальность. Доказывается данная теорема следующим образом. Пусть F — величина некоторого до-

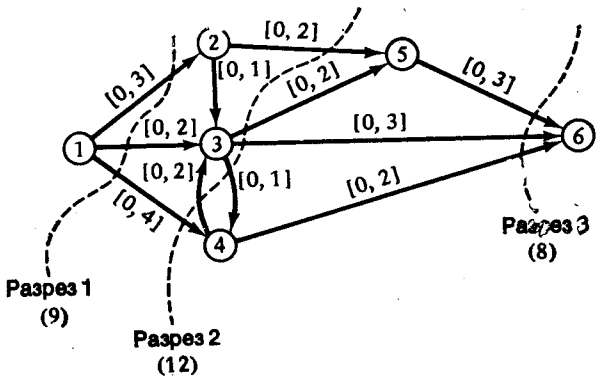


Рис. 2.50. Теорема о максимальном потоке и минимальном разрезе.

пустимого потока из узла s в узел t. Для любого разреза $(\mathbf{N}_c, \overline{\mathbf{N}}_c)$ справедливо равенство

$$F=\sum_{i\in \mathbf{N}_c}\Big[\sum_{j\in \mathbf{N}} f_{ij}-\sum_{j\in \mathbf{N}} f_{ji}\Big].$$

Поскольку $\mathbf{N}=\mathbf{N}_c \cup \overline{\mathbf{N}}_c$ и узел не может одновременно принадлежать двум подмножествам $\mathbf{N}_c$ и $\overline{\mathbf{N}}_c$, то

$$F=\sum_{\substack{i\in \mathbf{N}_c\\ j\in \mathbf{N}_c}} f_{ij}+\sum_{\substack{i\in \mathbf{N}_c\\ j\in \overline{\mathbf{N}}_c}} f_{ij}-\sum_{\substack{i\in \mathbf{N}_c\\ j\in \mathbf{N}_c}} f_{ji}-\sum_{\substack{i\in \mathbf{N}_c\\ j\in \overline{\mathbf{N}}_c}} f_{ji},$$

или

$$F=\Big[\sum_{\substack{i\in \mathbf{N}_c\\ j\in \overline{\mathbf{N}}_c}} f_{ij}-\sum_{\substack{i\in \mathbf{N}_c\\ j\in \overline{\mathbf{N}}_c}} f_{ji}\Big]+\Big[\sum_{\substack{i\in \mathbf{N}_c\\ j\in \mathbf{N}_c}} f_{ij}-\sum_{\substack{i\in \mathbf{N}_c\\ j\in \mathbf{N}_c}} f_{ji}\Big].$$

Второе слагаемое равно нулю, поэтому

$$F=\sum_{\substack{i\in \mathbf{N}_c\\ j\in \overline{\mathbf{N}}_c}} f_{ij}-\sum_{\substack{i\in \mathbf{N}_c\\ j\in \overline{\mathbf{N}}_c}} f_{ji},$$

откуда следует, что

$$F \leqslant \sum_{\substack{i \in \mathbf{N}_c \\ j \in \overline{\mathbf{N}}_c}} f_{ij}, \tag{2.61}$$

поскольку $f_{ji} \geqslant 0$. Следовательно, величина F любого потока ограничена пропускной способностью произвольного разреза $(\mathbf{N}_c, \overline{\mathbf{N}}_c)$, откуда следует, что величина максимального допустимого потока ограничена сверху пропускной способностью минимального разреза.

Из (2.61) и неравенства $f_{ij} \leqslant U_{ij}$ следует, что

$$F \leqslant \sum_{i \in \mathbf{N}_c} \sum_{j \in \overline{\mathbf{N}}_c} U_{ij}. \tag{2.62}$$

Правая часть неравенства (2.62) равна величине разреза $(\mathbf{N}_c, \overline{\mathbf{N}}_c)$, отделяющего узел s от узла t. Обозначая эту величину через V_{st}, запишем (2.62) в следующем виде:

$$F \leqslant V_{st}. \tag{2.63}$$

Пусть $(\mathbf{L}, \overline{\mathbf{L}})$ — такой разрез, что (а) $s \in \mathbf{L}$, $t \in \overline{\mathbf{L}}$ и (б) $j \in \mathbf{L}$, если существует $i \in \mathbf{L}$, при котором $f_{ij} < U_{ij}$ или $f_{ji} > 0$[1]. Как было показано выше,

$$F = \sum_{i \in \mathbf{L}} \sum_{j \in \overline{\mathbf{L}}} f_{ij} - \sum_{i \in \mathbf{L}} \sum_{j \in \overline{\mathbf{L}}} f_{ji}. \tag{2.64}$$

Из определения множества $\mathbf{L}$ следует, что если $(i, j) \in (\mathbf{L}, \overline{\mathbf{L}})$, то $f_{ij} = U_{ij}$, а если $(j, i) \in (\overline{\mathbf{L}}, \mathbf{L})$, то $f_{ji} = 0$. Следовательно,

$$\sum_{i \in \mathbf{L}} \sum_{j \in \overline{\mathbf{L}}} f_{ij} = \sum_{i \in \mathbf{L}} \sum_{j \in \overline{\mathbf{L}}} U_{ij} \tag{2.65}$$

и

$$\sum_{i \in \mathbf{L}} \sum_{j \in \overline{\mathbf{L}}} f_{ji} = 0. \tag{2.66}$$

Поэтому

$$F = \sum_{i \in \mathbf{L}} \sum_{j \in \overline{\mathbf{L}}} U_{ij} \geqslant V_{st}. \tag{2.67}$$

---

[1] Здесь предполагается, что поток $\{f_{ij}\}$ максимальный. Разрез $(\mathbf{L}, \overline{\mathbf{L}})$ может быть построен с помощью следующей процедуры (см. [15]): а) определить множество $\mathbf{L} = \{s\}$; б) если $i \in \mathbf{L}$ и, кроме того, $f_{ij} < U_{ij}$ или $f_{ji} > 0$, то включить j в $\mathbf{L}$. Повторить шаг б) до тех пор, пока множество $\mathbf{L}$ нельзя будет расширять дальше. Тогда $t \in \overline{\mathbf{L}}$, поскольку в противном случае существовал бы аугментальный путь потока из s в t и поток не был бы максимальным. — *Прим. перев.*

Из неравенства (2.63) и (2.67) следует, что

$$F = V_{st}. \tag{2.68}$$

2.14.4. ПРОЕКТИРОВАНИЕ ЦЕНТРАЛИЗОВАННОЙ ВОДООЧИСТНОЙ СТАНЦИИ

На водоочистную станцию поступают сточные воды с девяти децентрализованных насосных подстанций, расположенных в городе. Разрабатывается проект перестройки города. В результате реализации этого проекта в существующую очистную систему дополнительно будет поступать большой объем сточных

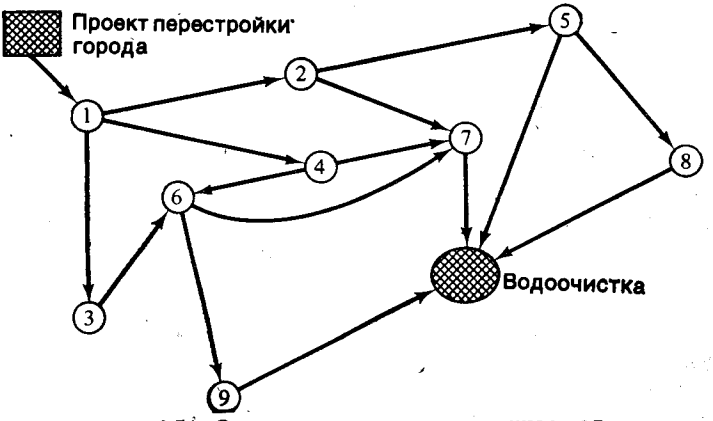


Рис. 2.51. Задача распределения сточных вод.

вод. Водоочистная станция является относительно новой, но опасаются, что пропускная способность системы сбора сточных вод может оказаться ниже требуемой. Задача состоит в определении максимального объема сточных вод, который может проходить через систему при существующем оборудовании. Основные элементы системы показаны на рис. 2.51. Узлы соответствуют насосным подстанциям, а дуги — трубопроводам.

На рис. 2.52 данная система изображена в виде обычной сети, для всех звеньев которой указаны верхние границы потока, соответствующие их максимальной пропускной способности. Для нахождения решения с помощью алгоритма поиска максимального потока мы воспользуемся начальным решением, указанным на рис. 2.52. На рис. 2.53, *а*—*е* дается последовательность шагов, в результате выполнения которых находится решение, указанное на рис. 2.53,*е*. На рис. 2.53,*е* показано, что максимальный поток равен 8 и что в оптимальном решении потоки на участках (6, 7), (5, 8) и (8, 10) трубопровода равны нулю. Это означает, что насосная подстанция 8 не играет никакой

роли в системе и поэтому она может быть отключена. Звено (6, 7) также можно исключить из системы. Если величина максимального допустимого потока не достаточна для планируемого расширения города, то очевидно, что трубопроводы бо́льших размеров для звеньев (1, 2) и (2, 5) увеличат пропускную способность системы. И наконец, отметим, что новые трубопроводы для звеньев (1, 4) и (1, 3) не увеличат пропускную способность системы без дополнительных ее изменений.

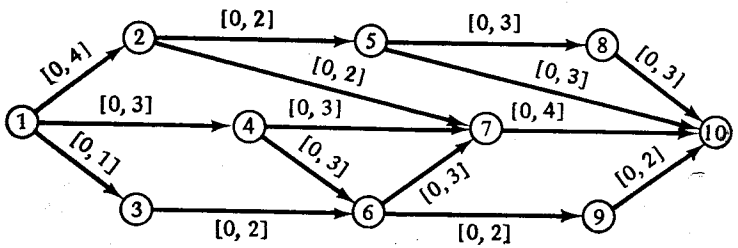


Рис. 2.52. Максимальный поток в задаче распределения сточных вод.

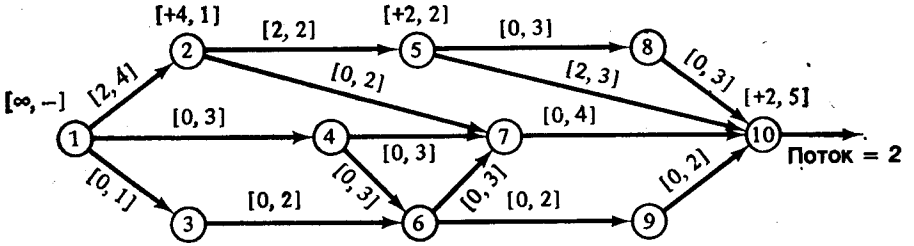
Результаты, приведенные на рис. 2.53, *а*, получаются после выполнения следующей последовательности операций: приписать узлу 1 пометку $[\infty, -]$, узлу 2 — пометку $[+4, 1]$, узлу 5 — пометку $[+2, 2]$, узлу 10 — пометку $[+2, 5]$. Изменить потоки: $f_{12}=2$, $f_{25}=2$ и $f_{5,10}=2$.

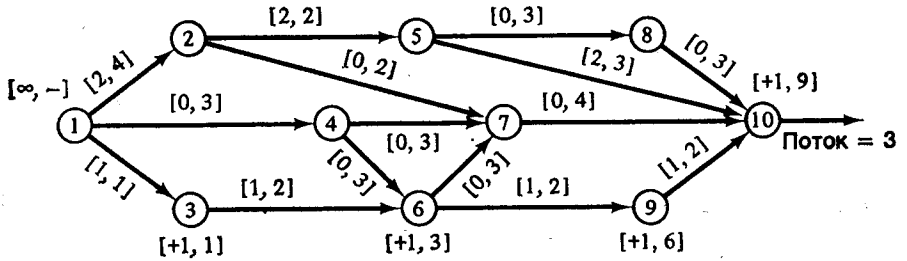
Результаты, приведенные на рис. 2.53, *б*, получаются после выполнения следующей последовательности операций: приписать узлу 1 пометку $[\infty, -]$, узлу 3 — пометку $[+1, 1]$, узлу 6 — пометку $[+1, 3]$, узлу 9 — пометку $[+1, 6]$, узлу 10 — пометку $[+1, 9]$. Изменить потоки: $f_{13}=1$, $f_{36}=1$, $f_{69}=1$ и $f_{9,10}=1$.

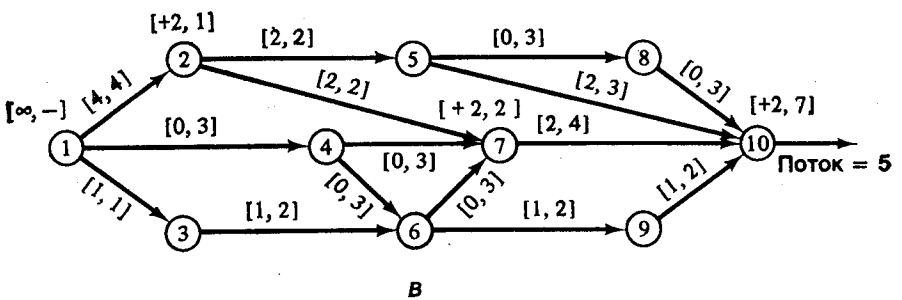
Результаты, приведенные на рис. 2.53, *в*, получаются после выполнения следующей последовательности операций: приписать узлу 1 пометку $[\infty, -]$, узлу 2 — пометку $[+2, 1]$, узлу 7 — пометку $[+2, 2]$, узлу 10 — пометку $[+2, 7]$. Изменить потоки: $f_{14}=2$, $f_{47}=2$ и $f_{7,10}=4$.

Результаты приведенные на рис. 2.53, *г*, получаются после выполнения следующей последовательности операций: приписать узлу 1 пометку $[\infty, -]$, узлу 4 — пометку $[+3, 1]$, узлу 7 — пометку $[+3, 4]$, узлу 10 — пометку $[+2, 7]$. Изменить потоки: $f_{14}=2$, $f_{47}=2$ и $f_{7,10}=4$.

Результаты, приведенные на рис. 2.53, *д*, получаются после выполнения следующей последовательности операций: приписать узлу 1 пометку $[\infty, -]$, узлу 4 — пометку $[+1, 1]$, узлу 6 — пометку $[+1, 4]$, узлу 9 — пометку $[+1, 6]$, узлу 10—







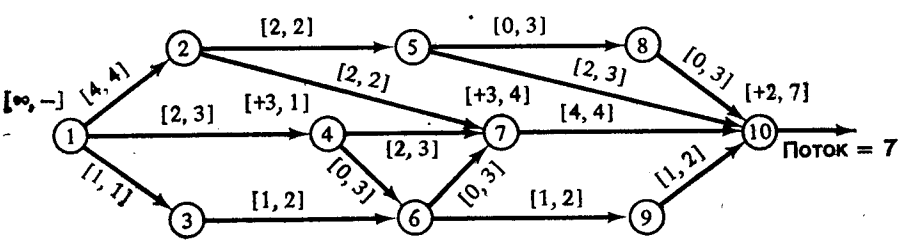


Рис. 2.53. Решение задачи распределения сточных вод.

а — первая итерация; *б* — вторая итерация; *в* — третья итерация; *г* — четвертая итерация; *д* — пятая итерация; *е* — решение.

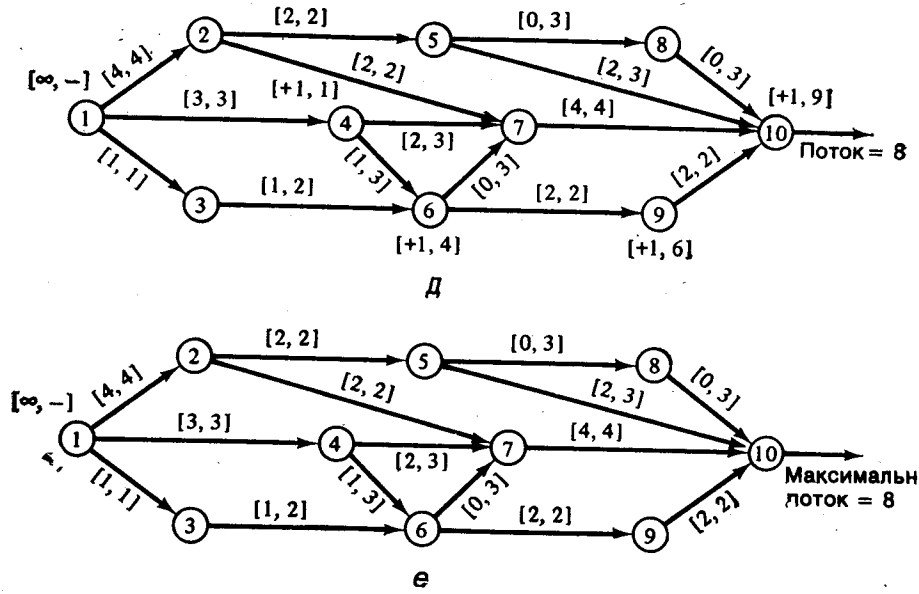


Рис. 2.53 (продолжение).

пометку [+1, 9]. Изменить потоки: $f_{14}=3$, $f_{46}=1$, $f_{69}=2$ и $f_{9,10}=2$.

Ни один из узлов сети, изображенной на рис. 2.53, *е*, не может быть помечен. Поэтому текущее решение является оптимальным, а величина максимального потока равна 8 ед.

2.14.5. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ О МАКСИМАЛЬНОМ ПОТОКЕ

Назначение: нахождение максимального потока в сети с ограниченной пропускной способностью.

Локализация: подпрограмма MAXFLOW в пакете сетевой оптимизации.

Ограничения: программа позволяет обрабатывать сети, содержащие до 50 узлов и 50 дуг. Размеры сети можно увеличить, изменив границы массивов в операторах размерности, записанных в подпрограмме MAXFLOW и основной программе.

Входные данные:

Набор 1. Одна карта с именем алгоритма MAXF в формате (A4).

Набор 2. Одна карта с числом узлов и числом дуг в сети в формате (2I10).

Набор 3. Общее число карт в данном наборе равно числу ориентированных дуг в сети. С каждой карты считываются следующие величины: 1) номер начального узла дуги, 2) номер конечного узла дуги, 3) пропускная способность дуги, 4) начальный поток по дуге (должно выполняться условие сохранения потока). Если поставлен знак пробела, то начальные потоки по всем дугам полагаются равными нулю.

Формат (4X, I6, I10, 2F10.2).

Набор 4. Данный набор состоит из одной карты, содержащей слово 'PEND' в формате (A4), которая указывает конец задачи.

Набор 5. Данный набор состоит из одной карты, содержащей слово 'EXIT' в формате (A4), которая указывает конец входных данных.

Используемые переменные: I — начальный узел дуги, J — конечный узел дуги, $A1$ — пропускная способность дуги, $A2$ — текущий поток по дуге.

Используемый метод: начиная с произвольного потока, удовлетворяющего условию сохранения, с помощью процедуры расстановки пометок определяются всевозможные аугментальные пути потока. Как только аугментальный путь потока найден, поток увеличивается на столько единиц, на сколько позволяет пропускная способность этого пути. Затем все пометки стираются. Алгоритм заканчивает работу, когда поток не может быть увеличен.

Литература: [31].

2.14.6. ЗАДАЧА О ТРАНСПОРТИРОВКЕ И ХРАНЕНИИ ЗЕРНА

Многие развивающиеся страны расходуют большие суммы денег на то, чтобы устранить транспортные заторы и уменьшить затраты на транспортировку и хранение сельскохозяйственной продукции. Для исследования данных проблем полезным оказывается сетевой подход, который позволяет учитывать ограничения на пропускную способность различных звеньев транспортных сетей и давать количественную оценку улучшениям отдельных элементов этих сетей.

Упрощенный вид транспортной сети прибрежной части одного развивающегося района показан на рис. 2.54. Объем поставок зерна, производимого в узле 1, должен соответствовать производительности портового оборудования (расположенного в узле 7), предназначенного для погрузки зерна. Узлы 2, 3, 4, 5 и 6 являются хранилищами или пунктами перепогрузки продукта. Автодорожная сеть включает в себя современные автострады, однако в данный момент она является незаконченной. Железнодорожная сеть устаревшая и состоит из нескольких от-

дельных веток. Числа, приписанные дугам сети, изображенной на рис. 2.54, соответствуют нулевому потоку, взятому в качестве начального решения, и максимально допустимым потокам по звеньям транспортной сети.

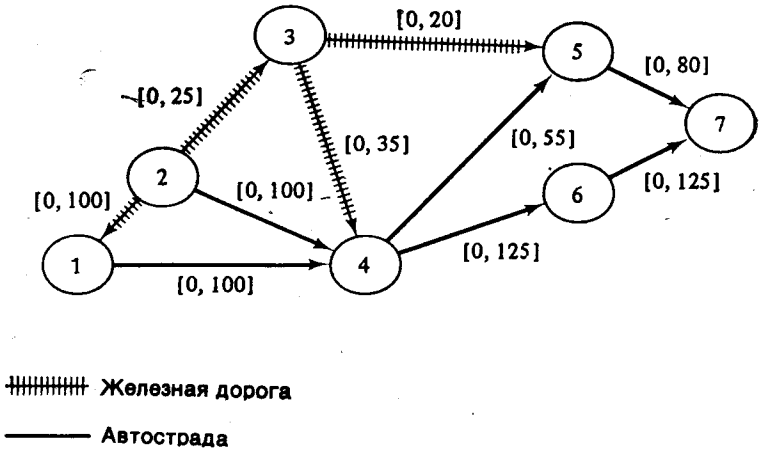


Рис. 2.54. Упрощенный вид транспортной сети, предназначенной для перевозки сельскохозяйственной продукции.

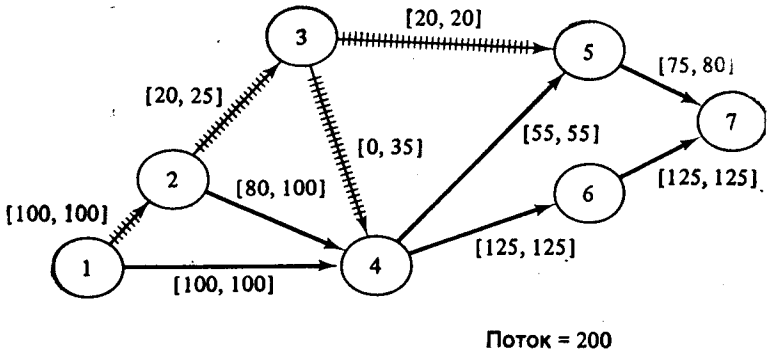


Рис. 2.55. Оптимальное решение в задаче транспортировки сельскохозяйственной продукции.

Данная сетевая модель может быть легко модифицирована с целью определения влияния пропускных способностей некоторых дуг, или звеньев, транспортной сети на величину максимального потока. Для этого достаточно изменять соответствующие параметры дуг. Новые транспортные линии обозначаются

дополнительными дугами и узлами. Аналогично при выходе из строя или закрытии транспортных линий соответствующие дуги сети удаляются.

Алгоритмы поиска максимального потока являются мощным средством при исследовании альтернативных вариантов капиталовложения в транспортную сеть и систему хранилищ, особенно в тех случаях, когда улучшение отдельных компонентов системы может повлиять на всю систему в целом. На рис. 2.55 указано оптимальное решение (максимальный поток), а решение, полученное на ЭВМ, показывает шаги вычислений.

ЗАДАЧА О МАКСИМАЛЬНОМ ПОТОКЕ
ПАРАМЕТРЫ СЕТИ И НАЧАЛЬНОЕ РЕШЕНИЕ

НАЧАЛЬНЫЙ УЗЕЛ ДУГИ	КОНЕЧНЫЙ УЗЕЛ ДУГИ	ПРОПУСКНАЯ СПОСОБНОСТЬ ДУГИ	ДУГОВОЙ ПОТОК
1	2	100,00	0,0
1	4	100,00	0,0
2	3	25,00	0,0
2	4	100,00	0,0
3	4	35,00	0,0
3	5	20,00	0,0
4	5	55,00	0,0
4	6	125,00	0,0
5	7	80,00	0,0
6	7	125,00	0,0

МАКСИМАЛЬНЫЙ ПОТОК ПО ПЕРЕЧИСЛЕННЫМ ВЫШЕ ДУГАМ = 0,0

ЗАДАЧА О МАКСИМАЛЬНОМ ПОТОКЕ
ПАРАМЕТРЫ СЕТИ И ДОПУСТИМОЕ РЕШЕНИЕ

НАЧАЛЬНЫЙ УЗЕЛ ДУГИ	КОНЕЧНЫЙ УЗЕЛ ДУГИ	ПРОПУСКНАЯ СПОСОБНОСТЬ ДУГИ	ДУГОВОЙ ПОТОК
1	2	100,00	20,00
1	4	100,00	0,0
2	3	25,00	20,00
2	4	100,00	0,0
3	4	35,00	0,0
3	5	20,00	20,00
4	5	55,00	0,0
4	6	125,00	0,0
5	7	80,00	20,00
6	7	125,00	0,0

МАКСИМАЛЬНЫЙ ПОТОК ПО ПЕРЕЧИСЛЕННЫМ ВЫШЕ ДУГАМ = 20,00

ЗАДАЧА О МАКСИМАЛЬНОМ ПОТОКЕ
ПАРАМЕТРЫ СЕТИ И ОПТИМАЛЬНОЕ РЕШЕНИЕ

НАЧАЛЬНЫЙ УЗЕЛ ДУГИ	КОНЕЧНЫЙ УЗЕЛ ДУГИ	ПРОПУСКНАЯ СПОСОБНОСТЬ ДУГИ	ДУГОВОЙ ПОТОК
1	2	100,00	100,00
1	4	100,00	100,00
2	3	25,00	20,00
2	4	100,00	80,00
3	4	35,00	0,0
3	5	20,00	20,00
4	5	55,00	55,00
4	6	125,00	125,00
5	7	80,00	75,00
6	7	125,00	125,00

МАКСИМАЛЬНЫЙ ПОТОК ПО ПЕРЕЧИСЛЕННЫМ ВЫШЕ ДУГАМ = 200,00

ЗАДАЧА О МАКСИМАЛЬНОМ ПОТОКЕ
ПАРАМЕТРЫ СЕТИ И ДОПУСТИМОЕ РЕШЕНИЕ

НАЧАЛЬНЫЙ УЗЕЛ ДУГИ	КОНЕЧНЫЙ УЗЕЛ ДУГИ	ПРОПУСКНАЯ СПОСОБНОСТЬ ДУГИ	ДУГОВОЙ ПОТОК
1	2	100,00	100,00
1	4	100,00	100,00
2	3	25,00	20,00
2	4	100,00	80,00
3	4	35,00	0,0
3	5	20,00	20,00
4	5	55,00	55,00
4	6	125,00	125,00
5	7	80,00	75,00
6	7	125,00	125,00

МАКСИМАЛЬНЫЙ ПОТОК ПО ПЕРЕЧИСЛЕННЫМ ВЫШЕ ДУГАМ = 200,00

2.15. ЗАДАЧА О МНОГОПОЛЮСНОМ МАКСИМАЛЬНОМ ПОТОКЕ

Существует большой ряд технических и экономических задач, в которых рассматриваемые системы могут быть приближенно описаны в виде детерминированных многополюсных потоковых моделей. Примерами таких систем являются: 1) транспортные сети, где автострады изображаются дугами, пропускные способности которых соответствуют максимально допустимой интенсивности движения; 2) телефонные сети, где телефонные линии представляются дугами, а пропускные способности соответствуют максимальному числу вызовов, которые могут обслуживаться в каждый момент времени; 3) электроэнергетические распределительные системы, где линии электропере-

дачи представлены дугами, а пропускные способности соответствуют максимальному объему электроэнергии, который может передаваться по линиям в единицу времени. Во всех этих задачах предполагается существование нескольких источников некоторого продукта. Предполагается также, что величина продукта, который может транспортироваться к нескольким стокам, ограничена только пропускными способностями распределительных звеньев.

Рассмотрим неориентированную сеть с ограниченной пропускной способностью, т. е. сеть, в которой потоки по дугам не должны превосходить пропускных способностей соответствующих дуг. В предыдущем разделе была описана процедура расстановки пометок для решения задачи с одним источником и одним стоком, в которой предполагалось, что из источника может поступать неограниченное количество продукта. Цель задачи состояла в нахождении максимального количества продукта, который может транспортироваться из источника в сток по дугам сети, не нарушая ограничений на пропускные способности дуг.

Несколькими математиками была рассмотрена задача нахождения максимального потока для всех пар узлов в неориентированной сети [7, 15, 25]. Данную задачу можно рассматривать как обобщение задачи с одним источником и одним стоком, и для ее решения можно воспользоваться процедурой, описанной в разд. 2.14.1, применяя ее к каждой паре узлов. Более изящный и более эффективный метод был предложен Гомори и Ху [25]. В настоящем разделе используются основные результаты, полученные в работе [25], и дается обоснование алгоритма.

Если пропускная способность каждой дуги не зависит от направления движения потока по этой дуге и если каждую пару узлов можно рассматривать как пару источник — сток, то общее число задач о максимальном потоке, которое должно быть решено, равно $n(n-1)/2$, где n — число узлов в сети. При работе алгоритма Гомори — Ху максимальный поток определяется только $n-1$ раз.

2.15.1. АЛГОРИТМ ГОМОРИ—ХУ

Пусть $G=(\mathbf{N}, \mathbf{A})$ — неориентированная сеть, где $\mathbf{N}$ — множество узлов, $\mathbf{A}$ — множество дуг. Пусть c_{ij} — пропускная способность дуги (i, j) из множества $\mathbf{A}$, и пусть $c_{ij}=c_{ji}$. Предположим также, что множество узлов задается в виде $\mathbf{N}=\{1, 2, \ldots, n\}$. При описании алгоритма будут использоваться следующие обозначения:

v_{ij} — максимальный поток между узлами i и j;

$(\mathbf{X}, \overline{\mathbf{X}})_{ij}$ — минимальный разрез, отделяющий i от j ($i \in \mathbf{X}$, $j \in \overline{\mathbf{X}}$);

$C(\mathbf{X}, \overline{\mathbf{X}})_{ij}$ — пропускная способность минимального разреза, отделяющего i от j.

Если некоторый узел s рассматривать как источник, а другой узел t — как сток, то, согласно теореме о максимальном потоке и минимальном разрезе, $v_{st} = C(\mathbf{X}, \overline{\mathbf{X}})_{st}$. Если затем в качестве источника и стока выбирается другая пара узлов (i и j соответственно), удовлетворяющих одному простому условию, то в алгоритме Гомори — Ху при определении величины v_{ij} используется решение задачи о максимальном потоке, найденное на предыдущем шаге. А именно, как доказано в работе [25], если узлы i и j выбираются таким образом, что оба они принадлежат $\mathbf{X}$ (или $\overline{\mathbf{X}}$), то множество узлов $\overline{\mathbf{X}}$ (или $\mathbf{X}$, если i и j принадлежат $\overline{\mathbf{X}}$) может быть объединено в один узел. При этом величина максимального потока из i в j будет одной и той же для исходной и конденсированной сетей.

Пусть $\overline{\mathbf{N}}_{ij}$ — множество узлов, образованное в результате конденсации всех узлов, лежащих по ту сторону разреза, где не содержатся узлы i и j. Пусть $\overline{\mathbf{A}}_{ij}$ — множество дуг, соединяющих узлы из $\overline{\mathbf{N}}_{ij}$. Тогда модифицированная сеть может быть представлена в виде $\overline{G}_{ij} = (\overline{\mathbf{N}}_{ij}, \overline{\mathbf{A}}_{ij})$. Если известны пропускные способности дуг, принадлежащих $\overline{\mathbf{A}}_{ij}$, то для нахождения величины максимального потока между узлами i и j можно воспользоваться процедурой расстановки пометок. Эти пропускные способности мы определим с помощью следующей простой процедуры. Пусть $j_1, j_2, \ldots, j_r$ — узлы из $\overline{\mathbf{X}}$, непосредственно связанные с узлом $i \in \mathbf{X}$. Если конденсируется множество $\overline{\mathbf{X}}$, то дуги $(i, j_1), (i, j_2), \ldots, (i, j_r)$ заменяются одной дугой, соединяющей узел i и конденсированный узел $\overline{\mathbf{X}}$. Пропускная способность этой дуги вычисляется следующим образом:

$$c_{i\overline{X}} = \sum_{m=1}^{r} c_{ij_m}.$$

Как отмечалось выше, величина максимального потока из i в j может быть вычислена с помощью процедуры расстановки пометок, примененной к сети $\overline{G}_{ij}$. Для определения величины v_{ij} вновь необходимо найти минимальный разрез, отделяющий i от j. Пусть $(\mathbf{X}, \overline{\mathbf{X}})_{ij}$ — соответствующий разрез с минимальной пропускной способностью. Теперь можно выбрать другую пару узлов, принадлежащую либо $\mathbf{X}$, либо $\overline{\mathbf{X}}$, и построить другую

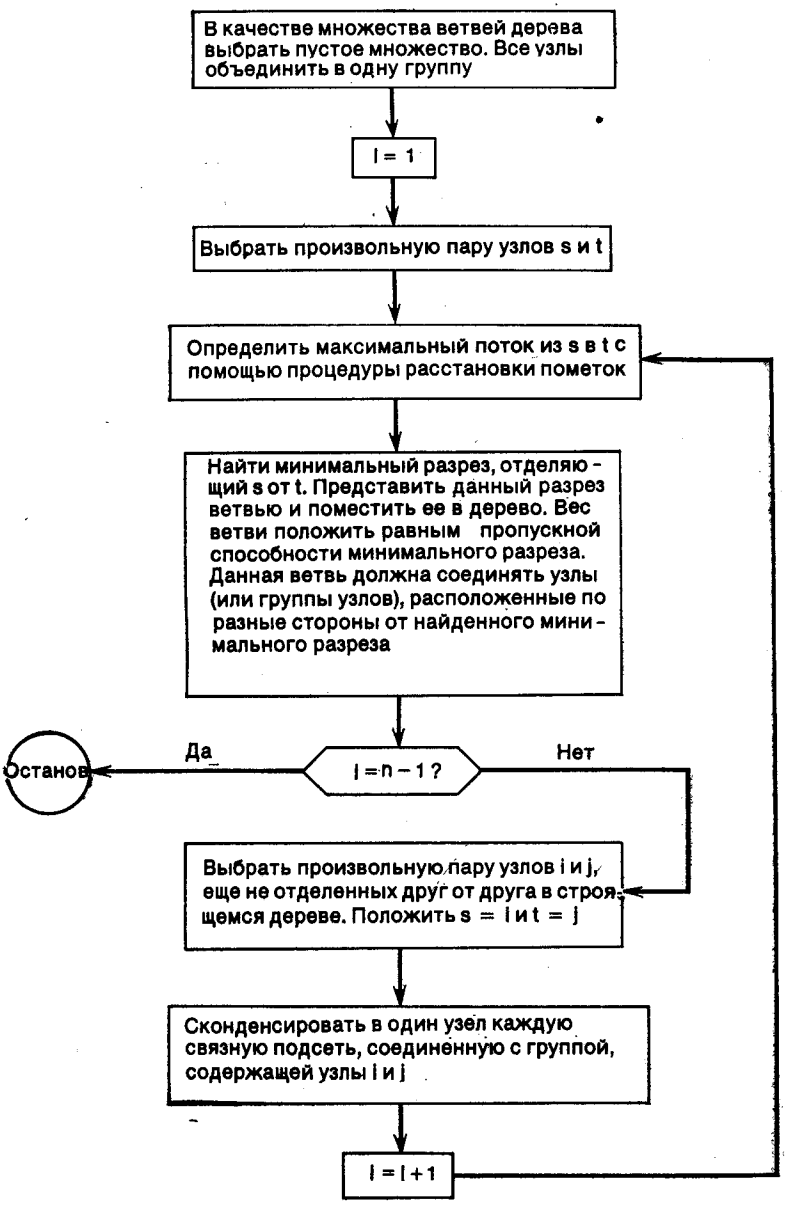


Рис. 2.56. Блок-схема алгоритма Гомори — Ху.

конденсированную сеть. В результате выполнения процедуры расстановки пометок можно будет определить другой разрез и построить новую конденсированную сеть. Можно показать, что, после того как будет выбрана n—1 пара узлов, мы определим все $n(n-1)/2$ величин максимального потока для исходной сети G.

Блок-схема алгоритма Гомори—Ху изображена на рис. 2.56. Основная идея алгоритма состоит в итеративном построении максимального остовного дерева, ветви которого соответствуют разрезам, а параметры ветвей — величинам разрезов. Ниже дается обоснование алгоритма и приводится иллюстративный пример.

2.15.2. ОБОСНОВАНИЕ АЛГОРИТМА

Пусть $G=(\mathbf{N}, \mathbf{A})$ — неориентированная сеть, и пусть пропускные способности всех дуг из $\mathbf{A}$ удовлетворяют условию $c_{ij}=c_{ji}$. Пусть $i, j, k \in \mathbf{N}$. Согласно теореме о максимальном потоке и минимальном разрезе, $v_{ij}=C(\mathbf{X}, \overline{\mathbf{X}})_{ij}$. Если $k \in \overline{\mathbf{X}}$, то $v_{ik} \leqslant C(\mathbf{X}, \overline{\mathbf{X}})_{ij}$, а если $k \in \mathbf{X}$, то $v_{kj} \leqslant C(\mathbf{X}, \overline{\mathbf{X}})_{ij}$. Следовательно, $v_{ij} \geqslant v_{ik}$ и $v_{ij} \geqslant v_{kj}$, откуда следует, что $v_{ij} \geqslant \min[v_{ik}, v_{kj}]$. Если аналогичные рассуждения повторить для v_{ik} и v_{kj}, то мы получим следующие результаты: $v_{ik} \geqslant \min[v_{ip}, v_{pk}]$, $v_{kj} \geqslant \min[v_{kq}, v_{qj}]$, где $\{i, p, k, q, j\}$ — связное множество узлов из $\mathbf{N}$. Следовательно, $v_{ij} \geqslant \min[v_{ip}, v_{pk}, v_{kq}, v_{qj}]$. В общем случае

$$v_{ij} \geqslant \min\left[v_{ii_1}, v_{i_1i_2}, v_{i_2i_3}, \ldots, v_{i_rj}\right], \tag{2.69}$$

где $\{i, i_1, i_2, \ldots, j\}$ — связное множество узлов из $\mathbf{N}$.

Перед тем как продолжить наши рассуждения, докажем следующее свойство максимального остовного дерева:

$$w_{ij} \leqslant \min\left[w_{ii_1}, w_{i_1i_2}, w_{i_2i_3}, \ldots, w_{i_rj}\right], \tag{2.70}$$

где (i, j) — произвольная дуга, *не принадлежащая* данному дереву, $\{i, i_1, i_2, \ldots, j\}$ — единственная последовательность узлов, соединяющих ветви дерева, w_{ij} — вес дуги сети. Если неравенство (2.70) не верно, то вместо любой дуги пути из i в j можно взять дугу (i, j), в результате чего будет построено дерево с бо́льшим весом. Данное противоречие доказывает справедливость неравенства (2.70).

Если веса w_{ij} дуг остовного дерева положить равными v_{ij}, то для любой дуги (i, j), не принадлежащей дереву, будет справедливо соотношение

$$v_{ij} \leqslant \min\left[v_{ii_1}, v_{i_1i_2}, \ldots, v_{i_rj}\right], \tag{2.71}$$

где $\{i, i_1, i_2, \ldots, i_r, j\}$ — связная последовательность узлов дерева, принадлежащих пути из i в j. Из неравенств (2.69) и (2.71) получаем, что для любой дуги, не принадлежащей дереву,

$$v_{ij} = \min\left[v_{ii_1}, v_{i_1i_2}, \ldots, v_{i_rj}\right]. \qquad (2.72)$$

Максимальное остовное дерево, удовлетворяющее равенству (2.72), называется *деревом разрезов* потому, что каждая его ветвь соответствует разрезу, а вес ветви равен пропускной способности разреза. Если требуется определить величину максимального потока между двумя произвольными узлами, надо в дереве найти путь, соединяющий эти два узла, и выбрать в этом пути дугу с минимальным весом. Вес этой дуги равен величине максимального потока между рассматриваемыми узлами.

2.15.3. ПРИМЕР ЗАДАЧИ О МНОГОПОЛЮСНОМ МАКСИМАЛЬНОМ ПОТОКЕ

Рассмотрим сеть, изображенную на рис. 2.57. Числа, приписанные дугам, соответствуют их пропускным способностям. Требуется для каждой пары узлов сети определить величину

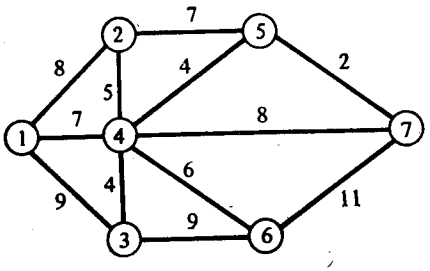


Рис. 2.57. Сеть в задаче о многополюсном максимальном потоке.

максимального потока между ними. Данная задача решается за $n-1=7-1=6$ итераций алгоритма Гомори—Ху. Если процедура расстановки пометок применялась бы к каждой паре узлов, то потребовалось бы решить 21 задачу о максимальном потоке. Разрезы, построенные на каждой итерации, состоят из дуг, остаточная пропускная способность которых равна нулю. Для простоты изложения мы опустили результаты, полученные при выполнении процедур расстановки пометок.

Итерация 1. Рассмотрим узлы $s=2$ и $t=5$. Величина максимального потока равна 13. Поэтому $v_{25}=v_{52}=13$. По разрезу с минимальной пропускной способностью мы определяем, что построение дерева разрезов можно начать с ветви, соединяющей

узел 5 и конденсированный узел, состоящий из узлов 1, 2, 3, 4, 6, 7 (рис. 2.58а). Вес данной ветви равен 13.

Итерация 2. Рассмотрим узлы $s=1$ и $t=2$. Величина максимального потока равна 19. Поэтому $v_{12}=v_{21}=19$. По минимальному разрезу мы определяем, что узлы 2 и 5 лежат по одну его сторону, а все остальные узлы сети — по другую сторону этого

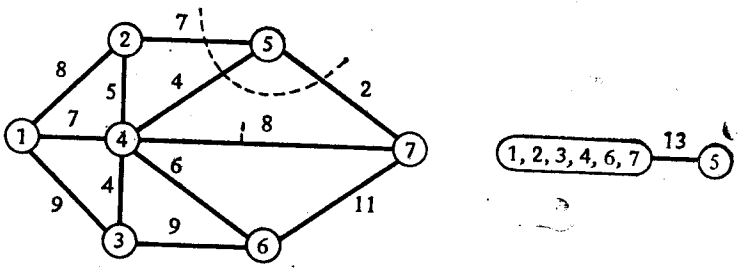


Рис. 2.58а. Задача о максимальном потоке: первая итерация.

разреза (рис. 2.58б). Вес ветви, соединяющей узел 2 и конденсированный узел, состоящий из узлов 1, 3, 4, 6 и 7, равен 19.

Итерация 3. Рассмотрим узлы 6 и 7. Величина максимального потока равна 21. Поэтому $v_{67}=v_{76}=21$. По минимальному раз-

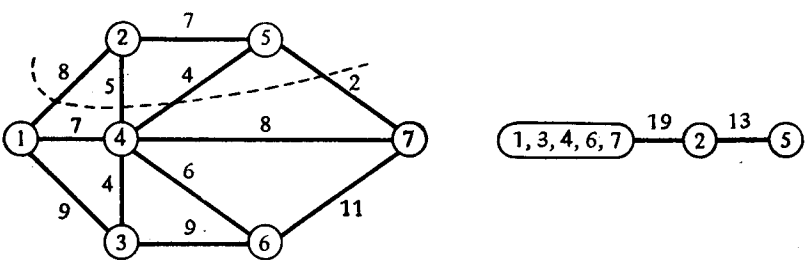


Рис. 2.58б. Вторая итерация.

резу мы определяем, что узел 7 в дереве разрезов соединяется с конденсированным узлом, состоящим из узлов 1, 3, 4, 6, дугой, вес которой равен 21. Кроме того, узлы 2 и 5 и конденсированный узел (1, 3, 4, 6) расположены по одну сторону минимального разреза, а узел 7 — по другую сторону этого разреза (рис. 2.58в).

Итерация 4. Рассмотрим узлы $s=4$ и $t=6$. Величина максимального потока равна 25. Поэтому $v_{46}=v_{64}=25$. По минимальному разрезу видно, что узлы 6 и 7 расположены в той же части дерева разрезов, что и узел (1, 3, 4) (рис. 2.58г).

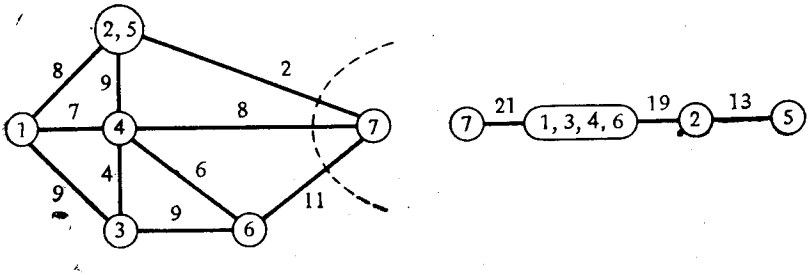


Рис. 2.58в. Третья итерация.

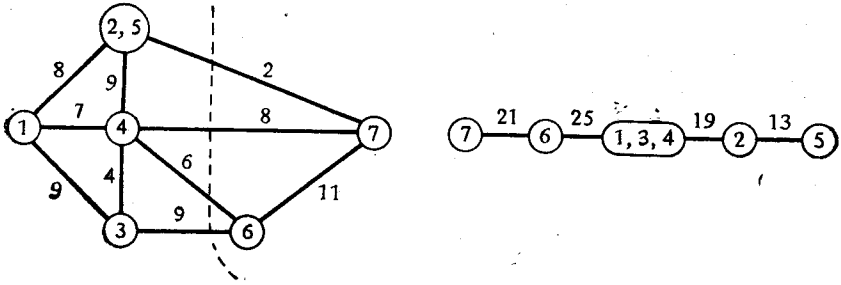


Рис. 2.58г. Четвертая итерация.

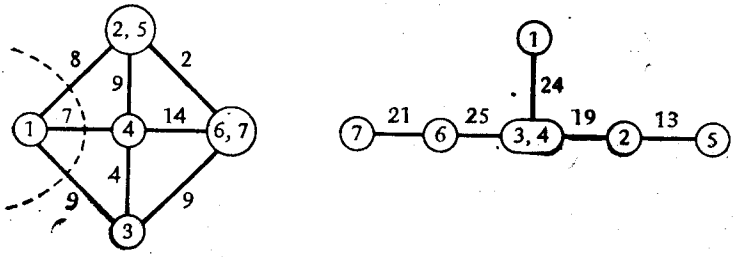


Рис. 2.58д. Пятая итерация.

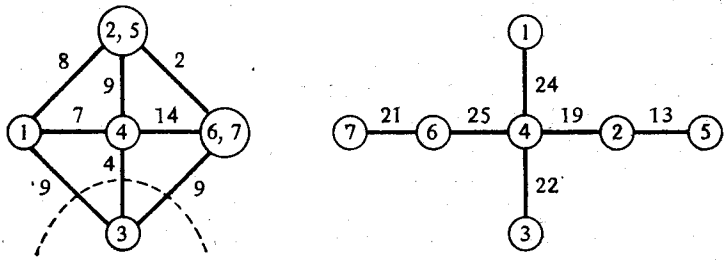


Рис 2.58е. Шестая итерация.

Итерация 5. Рассмотрим узлы $s=1$ и $t=4$. Величина максимального потока равна 24. Поэтому $v_{14}=v_{41}=24$. Определяя минимальный разрез, удаляем узел 1 из узла (1, 3, 4) и располагаем его по ту сторону узла (3, 4), где не находится ни один из оставшихся узлов (рис. 2.58д). Вес соответствующей дуги в дереве разрезов равен 24.

Итерация 6. Рассмотрим узлы $s=3$ и $t=4$. Величина максимального потока равна 22. Поэтому $v_{34}=v_{43}=22$. Найдя минимальный разрез, удаляем узел 3 из узла (3, 4) и соединяем его с узлом 4 дугой дерева разрезов, вес которой равен 22. Теперь дерево разрезов стало полным, т. е. состоит из шести дуг. Поэтому процедура заканчивается (рис. 2.58е).

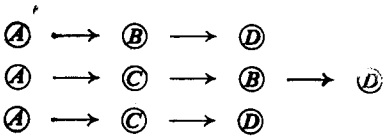
Величины максимальных потоков можно записать в виде следующей матрицы:

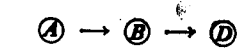
$$\mathbf{V}=\begin{bmatrix} — & 19 & 22 & 24 & 13 & 24 & 21 \\ 19 & — & 19 & 19 & 13 & 19 & 19 \\ 22 & 19 & — & 22 & 13 & 22 & 21 \\ 24 & 19 & 22 & — & 13 & 25 & 21 \\ 13 & 13 & 13 & 13 & — & 13 & 13 \\ 24 & 19 & 22 & 25 & 13 & — & 21 \\ 21 & 19 & 21 & 21 & 13 & 21 & — \end{bmatrix}.$$

2.16. ЗАДАЧА О МНОГОПОЛЮСНОЙ ЦЕПИ С МАКСИМАЛЬНОЙ ПРОПУСКНОЙ СПОСОБНОСТЬЮ

С задачей о многополюсном максимальном потоке, рассмотренной в разд. 2.15, тесно связана задача о многополюсной цепи с максимальной пропускной способностью. Алгоритм, описанный в разд. 2.15, позволяет находить максимальный поток между каждой парой узлов. Очевидно, максимальному потоку между каждой парой узлов могло соответствовать множество путей или цепей из источника в сток. В действительности в задаче о максимальном потоке рассматриваются лишь те пути или цепи, с помощью которых можно увеличить поток из одной точки в другую. Рассмотрим простую сеть, изображенную на рис. 2.59. (Числа, приписанные дугам, соответствуют верхним границам потоков по ним.) Величина максимального потока между узлами A и D равна 40, а соответствующие потоки по дугам следующие $f_{AB}=20$, $f_{AC}=20$, $f_{CB}=5$, $f_{BD}=25$, $f_{CD}=15$. Узлы A и D соединены тремя цепями, как показано ниже.

Рассмотрим следующую задачу, относящуюся к приведенной выше сети: какая цепь, ведущая из узла A в узел D, имеет



максимальную пропускную способность? Очевидно, цепь с максимальной пропускной способностью определяется последовательностью узлов  , а величина максимального потока по этой цепи равна $F_{\max}=20$. Задача, которую мы хотим рассмотреть в данном разделе, — это задача о многополюсной цепи с максимальной пропускной способностью, т. е. задача о цепи с максимальным потоком между всеми парами узлов.

Ху [31] была разработана эффективная вычислительная процедура, которая представляет собой модификацию трехместной операции, используемой при решении задачи о многополюсной кратчайшей цепи (пути). Данный алгоритм работает следующим образом.

Шаг 1. Построить *матрицу пропускных способностей* размером $n\times n$, элементы которой соответствуют пропускным способностям дуг между узлами i и j $(i, j=1, 2, ..., n)$.

$$f_{AB}=20$$
$$f_{AC}=20$$
$$f_{CB}=5$$
$$f_{BD}=25$$
$$f_{CD}=15$$

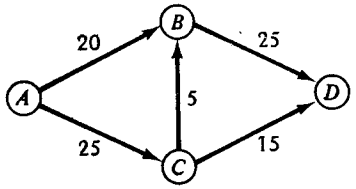


Рис. 2.59. Сеть в задаче о цепи с максимальной пропускной способностью.

Шаг 2. а. Для каждого $j=1$, 2, ..., n выполнить следующую процедуру: исключить j-ю строку и i-й столбец матрицы и над каждым оставшимся элементом d_{ik} (диагональные элементы также исключаются) выполнить трехместную операцию $d_{ik}=\max\{d_{ik};\ \min[d_{ij},\ d_{jk}]\}$ для всех $i, k\neq j$, $j=1$, 2, ..., n.

Вторая матрица, называемая *матрицей маршрутов,* необходима для определения внутренних узлов каждой цепи. Матрица маршрутов также имеет размеры $n\times n$, а k-й элемент i-й строки в ней первоначально равен k.

б. Одновременно с заменами элементов в матрице пропускных способностей выполнить замены элементов в матрице

маршрутов по следующему правилу:

$$r_{ik} = \begin{cases} j, \text{ если } d_{ik} < \min[d_{ij}, d_{jk}], \\ \text{остается неизменным, если } d_{ik} \geqslant \min[d_{ij}, d_{jk}]. \end{cases}$$

В заключение отметим, что если узлы i и j не соединены дугой (или связь между ними недопустима), то значение соответствующего элемента d_{ij} матрицы пропускных способностей полагается равным $-\infty$.

Для иллюстрации данного алгоритма рассмотрим следующую задачу.

2.16.1. ОПТИМАЛЬНЫЙ МАРШРУТ ПЕРЕВОЗКИ НЕУПАКОВАННОГО ГРУЗА

East Texas Freight Company (ETFC) управляет фирмой, производящей перевозки автотранспортом крупногабаритного оборудования. ETFC предлагают заключить специальный контракт о перевозке неупакованного груза между семью предприятиями НАСА, расположенными в Хьюстоне, шт. Техас. По этому контракту требуется перевозить несколько видов оборудования с очень большим вертикальным габаритом. ETFC должна перевозить это оборудование только по семи утвержденным маршрутам. Задача заключается в выборе маршрутов с максимально допустимыми подмостовыми зазорами. ETFC определила высоты проездов под мостами и для всех маршрутов, соединяющих пункты погрузки и разгрузки, нашла максимально допустимые вертикальные габариты груза.

Таблица 2.37. Транспортировка неупакованного груза

Из \ В	1	2	3	4	5	6	7
1	0	11	30	—	—	—	—
2	11	0	—	12	2	—	—
3	30	—	0	19	—	4	—
4	—	12	19	0	11	9	—
5	—	2	—	11	0	—	—
6	—	—	4	9	—	0	—
7	—	—	—	20	1	1	0

Каждый элемент матрицы, приведенной в табл. 2.37, равен высоте проезда под мостом (в фут.), уменьшенной на 30 фут. ETFC требуется информация о максимально допустимых вертикальных габаритах грузов, которые могут транспортироваться с каждого пункта погрузки в каждый пункт выгрузки.

Данную задачу можно решить следующим образом: для каждой пары узлов определяется соединяющий их маршрут с максимальной пропускной способностью. Ниже приводятся результаты вычислений.

Итерация 0: **Исходные матрицы**

Матрица пропускных способностей

	1	2	3	4	5	6	7
1	0	11	30	−∞	−∞	−∞	−∞
2	11	0	−∞	12	2	−∞	−∞
3	30	−∞	0	19	−∞	4	−∞
4	−∞	12	19	0	11	9	−∞
5	−∞	2	0	11	0	−∞	−∞
6	−∞	−∞	4	9	−∞	0	−∞
7	−∞	−∞	−∞	20	1	1	0

Матрица маршрутов

	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	1	2	3	4	5	6	7
3	1	2	3	4	5	6	7
4	1	2	3	4	5	6	7
5	1	2	3	4	5	6	7
6	1	2	3	4	5	6	7
7	1	2	3	4	5	6	7

Итерация 1:

Матрица пропускных способностей

	1	2	3	4	5	6	7
1	0	11	30	−∞	−∞	−∞	−∞
2	11	0	11	12	2	−∞	−∞
3	30	11	0	19	−∞	4	−∞
4	−∞	12	19	0	11	9	−∞
5	−∞	2	−∞	11	0	−∞	−∞
6	−∞	−∞	4	9	−∞	0	−∞
7	−∞	−∞	−∞	20	1	1	0

Матрица маршрутов

	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	1	2	1	4	5	6	7
3	1	1	3	4	5	6	7
4	1	2	3	4	5	6	7
5	1	2	3	4	5	6	7
6	1	2	3	4	5	6	7
7	1	2	3	4	5	6	7

Итерация 2:

Матрица пропускных способностей

	1	2	3	4	5	6	7
1	0	11	30	11	2	$-\infty$	$-\infty$
2	11	0	11	12	2	$-\infty$	$-\infty$
3	30	11	0	19	2	4	$-\infty$
4	11	12	19	0	11	9	$-\infty$
5	2	2	2	11	0	$-\infty$	$-\infty$
6	$-\infty$	$-\infty$	4	9	$-\infty$	0	$-\infty$
7	$-\infty$	$-\infty$	$-\infty$	20	1	1	0

Матрица маршрутов

	1	2	3	4	5	6	7
1	1	2	3	2	2	6	7
2	1	2	1	4	5	6	7
3	1	1	3	4	2	6	7
4	2	2	3	4	5	6	7
5	2	2	2	4	5	6	7
6	1	2	3	4	5	6	7
7	1	2	3	4	5	6	7

Итерация 3:

Матрица пропускных способностей

	1	2	3	4	5	6	7
1	0	11	30	19	2	4	$-\infty$
2	11	0	11	12	2	4	$-\infty$
3	30	11	0	19	2	4	$-\infty$
4	19	12	19	0	11	9	$-\infty$
5	2	2	2	11	0	2	$-\infty$
6	4	4	4	9	2	0	$-\infty$
7	$-\infty$	$-\infty$	$-\infty$	20	1	1	0

Матрица маршрутов

	1	2	3	4	5	6	7
1	1	2	3	3	2	3	7
2	1	2	1	4	5	3	7
3	1	1	3	4	2	6	7
4	3	2	3	4	5	6	7
5	2	2	2	4	5	3	7
6	3	3	3	4	3	6	7
7	1	2	3	3	5	6	7

Итерация 4:

Матрица пропускных способностей

	1	2	3	4	5	6	7
1	0	12	30	19	11	9	$-\infty$
2	12	0	12	12	11	9	$-\infty$
3	30	12	0	19	11	9	$-\infty$
4	19	12	19	0	11	9	$-\infty$
5	11	11	11	11	0	9	$-\infty$
6	9	9	9	9	9	0	$-\infty$
7	19	12	19	20	11	9	0

Матрица маршрутов

	1	2	3	4	5	6	7
1	1	4	3	3	4	4	7
2	4	2	4	4	4	4	7
3	1	4	3	4	4	4	7
4	3	2	3	4	5	6	7
5	4	4	4	4	5	4	7
6	4	4	4	4	4	6	7
7	4	4	4	4	4	4	7

Итерация 5:

Матрица пропускных способностей

	1	2	3	4	5	6	7
1	0	12	30	19	11	9	$-\infty$
2	12	0	12	12	11	9	$-\infty$
3	30	12	0	19	11	9	$-\infty$
4	19	12	19	0	11	9	$-\infty$
5	11	11	11	11	0	9	$-\infty$
6	9	9	9	9	9	0	$-\infty$
7	19	12	19	20	11	9	0

Матрица маршрутов

	1	2	3	4	5	6	7
1	1	4	3	3	4	4	7
2	4	2	4	4	4	4	7
3	1	4	3	4	4	4	7
4	3	2	3	4	5	6	7
5	4	4	4	4	5	4	7
6	4	4	4	4	4	6	7
7	4	4	4	4	4	4	7

Итерация 6:

Матрица пропускных способностей

	1	2	3	4	5	6	7
1	0	12	30	19	11	9	$-\infty$
2	12	0	12	12	11	9	$-\infty$
3	30	12	0	19	11	9	$-\infty$
4	19	12	19	0	11	9	$-\infty$
5	11	11	11	11	0	9	$-\infty$
6	9	9	9	9	9	0	$-\infty$
7	19	12	19	20	11	9	0

Матрица маршрутов

	1	2	3	4	5	6	7
1	1	4	3	3	4	4	7
2	4	2	4	4	4	4	7
3	1	4	3	4	4	4	7
4	3	2	3	4	5	6	7
5	4	4	4	4	5	4	7
6	4	4	4	4	4	6	7
7	4	4	4	4	4	4	7

Итерация 7:

Матрица пропускных способностей

	1	2	3	4	5	6	7
1	0	12	30	19	11	9	$-\infty$
2	12	0	12	12	11	9	$-\infty$
3	30	12	0	19	11	9	$-\infty$
4	19	12	19	0	11	9	$-\infty$
5	11	11	11	11	0	9	$-\infty$
6	9	9	9	9	9	0	$-\infty$
7	19	12	19	20	11	9	0

Матрица маршрутов

	1	2	3	4	5	6	7
1	1	4	3	3	4	4	7
2	4	2	4	4	4	4	7
3	1	4	3	4	4	4	7
4	3	2	3	4	5	6	7
5	4	4	4	4	5	4	7
6	4	4	4	4	4	6	7
7	4	4	4	4	4	4	7

Для каждой пары узлов максимально допустимый вертикальный габарит перевозимого груза можно определить *непосредственно* из последней матрицы пропускных способностей. Оптимальный маршрут также может быть построен с помощью последней матрицы маршрутов. Например, максимальная пропускная способность цепи из пункта 1 в пункт 4 задается значением элемента $d^*_{14}=19$ (подмостовой зазор равен 49 фут.). Соответствующий маршрут строится следующим образом:

$r^*_{14}=3$, двигаться из узла 1 в узел 4 через узел 3;

$r^*_{34}=4$, двигаться из узла 3 непосредственно в узел 4;

$r^*_{13}=3$, двигаться из узла 1 непосредственно в узел 3.

$$① \xrightarrow{30} ③ \xrightarrow{19} ④ \qquad d^*_{14}=\min\,[30,\ 19]=19.$$

Отметим, что дуги, соединяющей узлы 1 и 4, не существует.

2.16.2. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ О МНОГОПОЛЮСНОЙ ЦЕПИ С МАКСИМАЛЬНОЙ ПРОПУСКНОЙ СПОСОБНОСТЬЮ

Назначение: нахождение цепи с максимальной пропускной способностью для каждой пары узлов источник — сток в ориентированной сети.

Локализация: подпрограмма MAXCAP в пакете сетевой оптимизации.

Ограничения: программа позволяет обрабатывать сети, содержащие до 50 узлов и 50 дуг. Размеры сети можно увеличить, изменив границы массивов в операторах размерности, записанных в подпрограмме MAXCAP и в основной программе.

Входные данные:

Набор 1. Одна карта с именем алгоритма MCRP в формате (A4).

Набор 2. Одна карта с числом узлов и числом дуг в сети в формате (2I10).

Набор 3. Общее число карт в данном наборе равно числу дуг в сети. С каждой карты считываются следующие величины: 1) номер начального узла дуги; 2) номер конечного узла дуги; 3) пропускная способность дуги.

Формат (4X, I6, I10, F10.2).

Набор 4. Данный набор состоит из одной карты, содержащей слово 'PEND' в формате (A4), которая указывает конец задачи.

Набор 5. Данный набор состоит из одной карты, содержащей слово 'EXIT' в формате (A4), которая указывает конец входных данных.

Составные части программы. Программа состоит из следующих подпрограмм:

MAXCAP (ввод и вычисления);

PRNTRX (печать исходной матрицы);

PRNTRY (печать оптимального решения).

Используемые переменные: I — начальный узел дуги, J — конечный узел дуги, K — промежуточный узел, A — пропускная способность дуги.

Используемый метод: в данном алгоритме пропускная способность ориентированной дуги между узлами I и K принимается за начальную оценку максимальной пропускной способности цепи из I в K. На J-й итерации производится сравнение максимальной пропускной способности цепи из I в K, содержащей промежуточные узлы из множества $\{1, 2, ..., J\}$, с максимальной пропускной способностью цепи, проходящей через узел J, с промежуточными узлами из множества $\{1, 2, ..., J-1\}$. Если использование узла J позволяет получить цепь с большей пропускной способностью, то данная величина принимается за новую оценку.

Литература: [31].

2.16.3. ТРАНСПОРТИРОВКА КОСМИЧЕСКОГО КОРАБЛЯ «ШАТТЛ»

Недавно в Калифорнии был построен космический корабль «Шаттл», который затем был перевезен на подходящее место для запуска. До начала строительства группе инженеров, работающих в НАСА, поручили определить место для строительства корпуса корабля и маршрут его транспортировки к месту запуска. Изучив данную проблему, инженеры нашли три подходящих места для строительства корабля и три участка для его возможного запуска. В связи с техническими трудностями транспортировки корабля было решено выполнять перевозку только на грузовом автомобиле с безбортовой платформой. Инженеров НАСА волновал вопрос о том, какой вес груза является максимально допустимым для транспортировки с мест возможного строительства корабля к местам его возможного запуска. Было установлено, что между всеми выбранными пунктами расположено 28 мостов, максимально допустимая нагрузка для которых недостаточно высокая. После проверки данных в отделении автомобильных дорог, была составлена сеть (рис. 2.60), представляющая собой схему расположения мостов. Каждая дуга сети соответствует мосту, а число, приписанное этой дуге, — максимально допустимой нагрузке (в тоннах) для соответствующего моста. Требуется определить, какой вес груза является максимально допустимым для транспортирования с мест строи-

тельства корабля (пункты *A*, *B* и *C*) к местам его запуска (пункты *D*, *E* и *F*).

Решение данной задачи, полученное с помощью ЭВМ, выглядит следующим образом. Значения элементов *исходной матрицы* равны максимально допустимым нагрузкам для соответ-

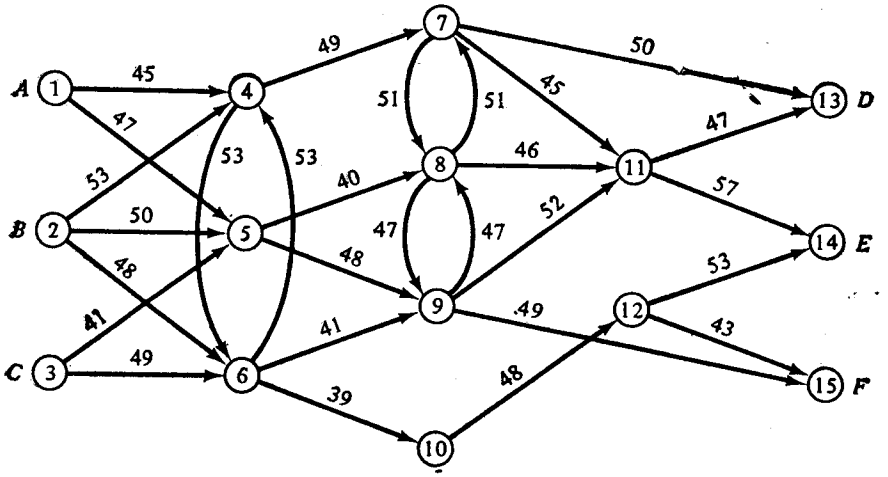


Рис. 2.60. Допустимые нагрузки на мосты.

ствующих мостов (дуг). Отметим, что запрещенным звеньям с пропускной способностью, равной —∞, приписано число 9999999,00. Элементы *матрицы пропускных способностей дуг* определяют максимальную пропускную способность маршрутов между всеми парами узлов. С помощью элементов последней матрицы, называемой *матрицей узлов*, определяются пути с максимальными пропускными способностями.

Таблица 2.38. Маршруты с максимальной пропускной способностью

Источник	Сток	Максимальная пропускная способность
1	13	47
2	13	49
3	13	49
1	14	47
2	14	48
3	14	47
1	15	47
2	15	48
3	15	47

ИСХОДНАЯ МАТРИЦА

	1	2	3	4	5
1	0.00	9999999.00	9999999.00	45.00	47.00
2	9999999.00	0.00	9999999.00	53.00	50.00
3	9999999.00	9999999.00	0.00	9999999.00	41.00
4	9999999.00	9999999.00	9999999.00	0.00	9999999.00
5	9999999.00	9999999.00	9999999.00	9999999.00	0.00
6	9999999.00	9999999.00	9999999.00	53.00	9999999.00
7	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
8	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
9	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
10	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
11	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
12	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
13	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
14	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
15	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00

	6	7	8	9	10
1	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
2	48.00	9999999.00	9999999.00	9999999.00	9999999.00
3	49.00	9999999.00	9999999.00	9999999.00	9999999.00
4	53.00	49.00	9999999.00	9999999.00	9999999.00
5	9999999.00	9999999.00	40.00	48.00	9999999.00
6	0.00	9999999.00	9999999.00	41.00	39.00
7	9999999.00	0.00	51.00	9999999.00	9999999.00
8	9999999.00	51.00	0.00	47.00	9999999.00
9	9999999.00	9999999.00	47.00	0.00	9999999.00
10	9999999.00	9999999.00	9999999.00	9999999.00	0.00
11	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
12	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
13	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
14	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
15	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00

	11	12	13	14	15
1	47.00	39.00	47.00	47.00	47.00
2	48.00	39.00	49.00	48.00	48.00
3	47.00	39.00	49.00	47.00	47.00
4	47.00	39.00	49.00	47.00	47.00
5	48.00	9999999.00	47.00	48.00	48.00
6	47.00	39.00	49.00	47.00	47.00
7	47.00	9999999.00	50.00	47.00	47.00
8	47.00	9999999.00	50.00	47.00	47.00
9	52.00	9999999.00	47.00	52.00	49.00
10	9999999.00	48.00	9999999.00	48.00	43.00
11	0.00	9999999.00	47.00	57.00	9999999.00
12	9999999.00	0.00	9999999.00	53.00	43.00
13	9999999.00	9999999.00	0.00	9999999.00	9999999.00
14	9999999.00	9999999.00	9999999.00	0.00	9999999.00
15	9999999.00	9999999.00	9999999.00	9999999.00	0.00

МАТРИЦА ПРОПУСКНЫХ СПОСОБНОСТЕЙ ДУГ

	1	2	3	4	5
1	0.00	9999999.00	9999999.00	45.00	47.00
2	9999999.00	0.00	9999999.00	53.00	50.00
3	9999999.00	9999999.00	0.00	49.00	41.00
4	9999999.00	9999999.00	9999999.00	0.00	9999999.00
5	9999999.00	9999999.00	9999999.00	9999999.00	0.00
6	9999999.00	9999999.00	9999999.00	53.00	9999999.00
7	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
8	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
9	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
10	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
11	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
12	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
13	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
14	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
15	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00

	6	7	8	9	10
1	45.00	47.00	47.00	47.00	39.00
2	53.00	49.00	49.00	48.00	39.00
3	49.00	49.00	49.00	47.00	39.00
4	53.00	49.00	49.00	47.00	39.00
5	9999999.00	47.00	47.00	48.00	9999999.00
6	0.00	49.00	49.00	47.00	39.00
7	9999999.00	0.00	51.00	47.00	9999999.00
8	9999999.00	51.00	0.00	47.00	9999999.00
9	9999999.00	47.00	47.00	0.00	9999999.00
10	9999999.00	9999999.00	9999999.00	9999999.00	0.00
11	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
12	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
13	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
14	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
15	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00

	11	12	13	14	15
1	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
2	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
3	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
4	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
5	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
6	9999999.00	9999999.00	9999999.00	9999999.00	9999999.00
7	45.00	9999999.00	50.00	9999999.00	9999999.00
8	46.00	9999999.00	9999999.00	9999999.00	9999999.00
9	52.00	9999999.00	9999999.00	9999999.00	49.00
10	9999999.00	48.00	9999999.00	9999999.00	9999999.00
11	0.00	9999999.00	47.00	57.00	9999999.00
12	9999999.00	0.00	9999999.00	53.00	43.00
13	9999999.00	9999999.00	0.00	9999999.00	9999999.00
14	9999999.00	9999999.00	9999999.00	0.00	9999999.00
15	9999999.00	9999999.00	9999999.00	9999999.00	0.00

МАТРИЧНОЕ ПРЕДСТАВЛЕНИЕ УЗЛОВ В ПУТЯХ
С МАКСИМАЛЬНОЙ ПРОПУСКНОЙ СПОСОБНОСТЬЮ

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	4	5	4	9	9	5	6	9	10	9	11	9
2	1	2	3	4	5	4	4	7	5	6	9	10	7	11	9
3	1	2	3	6	5	6	6	7	8	6	9	10	7	11	9
4	1	2	3	4	5	6	7	7	8	6	9	10	7	11	9
5	1	2	3	4	5	6	9	9	9	10	9	12	9	11	9
6	1	2	3	4	5	6	4	7	8	10	9	10	7	11	9
7	1	2	3	4	5	6	7	8	8	10	9	12	13	11	9
8	1	2	3	4	5	6	7	8	9	10	9	12	7	11	9
9	1	2	3	4	5	6	8	8	9	10	11	12	8	11	15
10	1	2	3	4	5	6	7	8	9	10	11	12	13	12	12
11	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
13	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Интерпретация выхода программы построения многополюсной цепи с максимальной пропускной способностью. Решения, представляющие для нас интерес, приводятся в табл. 2.38, а оптимальные решения — в табл. 2.39.

Таблица 2.39. Оптимальные решения

Источник	Сток	Путь	Пропускная способность оптимального маршрута
2	13	2–4–7–13	49
3	13	3–6–4–7–13	49

Следует отметить два момента. Во-первых, нами было получено два оптимальных решения, каждое из которых представляет путь, по которому можно перевозить груз весом 49 т. Во-вторых, численный метод нахождения этих путей такой, что матрица узлов может задавать узлы оптимального пути не последовательно. Это происходит по той причине, что длина пути из узла i в некоторый другой узел k сравнивается с длиной пути, содержащего промежуточный узел j (выполнение трехместной операции). Следовательно, при трассировке пути мы должны получить промежуточный узел j, который либо *следует* не-

Таблица 2.40. Процедура трассировки пути из узла 2 в узел 13

Из	В	Через узел	Путь
2	13	7	2 → 7 → 13
7	13	13	2 → 7 → 13
2	7	4	2 → 4 → 7 → 13
2	4	4	2 → 4 → 7 → 13
4	7	7	2 → 4 → 7 → 13

посредственно *за* узлом i, либо непосредственно *предшествует* узлу k, а также другие узлы оптимального пути между i и k. Например, при трассировке пути из источника 2 в сток 13 с помощью матрицы узлов мы получаем информацию, содержащуюся в табл. 2.40. Отметим, что данный путь определяется однозначно. Перейдем теперь к трассировке пути из источника 3 в сток 13. С помощью матрицы узлов мы получаем информацию, содержащуюся в табл. 2.41.

После небольшой тренировки читатель без труда справится с построением оптимального решения. В данном примере описана основная процедура поиска всех видов решений.

Таблица 2.41. Процедура трассировки пути из узла 3 в узел 13

Из	В	Через узел	Путь
3	13	7	3 → 7 → 13
7	13	13	3 → 7 → 13
3	7	6	3 → 6 → 7 → 13
6	7	4	3 → 6 → 4 → 7 → 13
4	7	7	3 → 6 → 4 → 7 → 13
3	6	6	3 → 6 → 4 → 7 → 13
6	4	4	3 → 6 → 4 → 7 → 13

2.17. ПОВРЕЖДЕНИЯ УЗЛОВ И ДУГ В СЕТЯХ

Очевидно, что с помощью сетей может быть описан широкий круг физических, экономических и управляющих систем. Большинство систем, рассмотренных выше, было связано с решением физических или экономических задач. Однако сетевой анализ может также играть большую роль при изучении управляющих систем и описании их характеристик. Ниже мы рассмотрим задачи, связанные с управляющими системами. В этих задачах изучается возможность систем эффективно ра-

ботать, или функционировать, после того, как одна или несколько компонент сети (узлы и дуги) будут «повреждены». Напомним, что сеть называется связной, если для каждой пары узлов существует по крайней мере один путь, соединяющий их. В противном случае сеть называется несвязной.

Для того чтобы определить характер задач, которые мы хотим рассмотреть, необходимо точно указать, как характеристики оперирующих систем связаны с процессом построения сетей. Сеть является *несвязной*, когда не для каждой пары узлов существует путь, соединяющий их. Отметим, что удаление одной или нескольких дуг может не привести к разрыву сети. Однако при последовательном удалении дуг, выбираемых произвольным образом, в конце концов сеть будет разорвана. Часто интерес представляет только вопрос о том, можно ли конечное множество узлов разъединить путем удаления дуг. Такой разрыв будем называть *частичным разрывом* и будем определять его только для двух или более узлов. Другой важной задачей, связанной с определением условий возникновения частичного разрыва, является задача нахождения числа узлов, удаление которых из сети приведет к тому, что не для каждой пары узлов будет существовать соединяющий их путь. Детальное изучение этой задачи было проведено Фрэнком и Фришем [18, 36], а также Клейтманом [36]. Последующий материал основан на их работах.

Первая задача, которую мы рассмотрим, состоит в нахождении минимального числа ветвей, удаление которых приводит к разрыву сети. Мы воспользуемся следующей теоремой, доказанной Фрэнком и Фришем [36].

Теорема. Если каждой ветви связной сети приписать пропускную способность, равную 1, то максимальный поток между каждой парой узлов сети равен минимальному числу ветвей, удаление которых приводит к разрыву всех путей из одного узла к другому.

Иными словами, данная теорема утверждает, что если пропускные способности всех дуг полагаются равными 1, то максимальный поток между каждой парой узлов равен числу путей между этими узлами, не содержащих общих дуг. Следовательно, если для каждой пары узлов найдена величина максимального потока между этими узлами, то число ветвей, удаление которых приводит к разрыву сети, равно минимуму среди всех величин максимальных потоков. Отметим, что если при выполнении вычислений будет использоваться алгоритм, описанный в разд. 2.15, то расчеты упростятся в результате получения всех частичных разрывов между всеми парами узлов.

Вторая задача, которую мы рассмотрим в данном разделе, состоит в нахождении минимального числа узлов, удаление ко-

торых из сети приведет к разъединению всех пар ее узлов. Данная задача может быть решена с помощью алгоритма, аналогичного тому, который описан в разд. 2.5. Выберем в качестве источника произвольный узел i, а в качестве стока — произвольный узел j. Предположим, что пропускная способность каждого узла равна 1. Эта величина соответствует штрафу за выполнение поворота, определенному для каждого узла в алгоритме из разд. 2.5. В рассматриваемом нами случае метка, приписанная каждой дуге фиктивной сети, равна 1. Будем рассматривать величину этой метки как пропускную способность.

Далее, определим максимальный поток из фиктивного источника в фиктивный сток. Величина данного потока равна максимальному числу путей в исходной сети, не имеющих общих узлов, за исключением источника и стока. Эти пути будем называть *путями, не пересекающимися по узлам.* Данная процедура выполняется для всех пар узлов сети. Число узлов, удаление которых приведет к разрыву сети, равно минимуму среди всех величин максимальных потоков.

Хотя приведенная выше процедура позволяет находить точное решение, при большом числе узлов она является очень громоздкой. Например, для того чтобы определить, приведет ли удаление не более четырех узлов сети, содержащей 1000 узлов, к полному ее разрыву, необходимо решить свыше 500 000 задач о максимальном потоке. Такие вычисления встречаются при решении задач небольшой размерности, связанных с распределением электроэнергии, в которых узлы представляют места соединений линий электропередач. Очевидно, что для решения практических задач нужен эффективный алгоритм. Клейтман [36] показал, что для решения описанной выше задачи достаточно решить не более 4000 задач о максимальном потоке с использованием следующей процедуры.

Выберем произвольный узел сети и убедимся, что максимальный поток из этого узла в любой другой узел сети составляет не менее 4 единиц. Если это так, то удалим из сети данный узел и все соединенные с ним ветви. Выберем один из узлов в полученной редуцированной сети и убедимся, что между этим узлом и всеми другими узлами редуцированной сети существует поток величиной не менее 3 единиц. Удалим из сети этот узел и соединенные с ним дуги и проверим, что между произвольно выбранным узлом полученной сети и каждым из оставшихся узлов существует поток величиной не менее 2 единиц. Выбирая четвертый узел (предварительно исключив соответствующий узел и соединенные с ним дуги), проверим, существует ли между этим узлом и всеми остальными узлами поток величиной в 1 единицу. Отметим, что при выполнении данной процедуры на каждом ее шаге между (произвольным)

источником и всеми остальными узлами строится некоторое количество путей, не пересекающихся по узлам. Если данную последовательность операций выполнить нельзя, то сеть будет разорвана в результате удаления из нее менее четырех узлов.

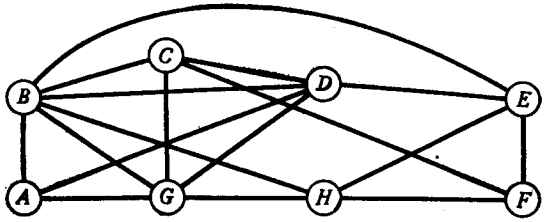


Рис. 2.61а. Пример сети к задаче о разрыве сетей.

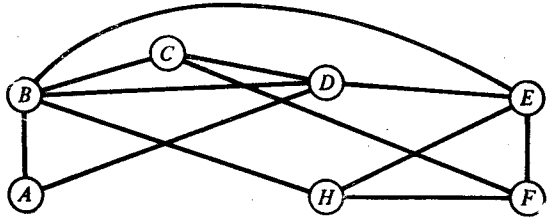


Рис. 2.61б. Сеть после удаления узла *G*.

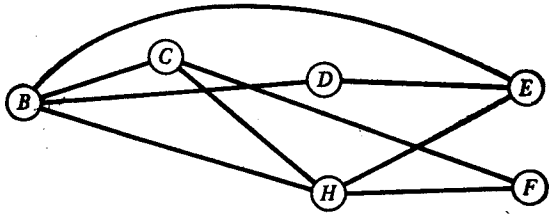


Рис. 2.61в. Сеть после удаления узлов *G* и *A*.

Например, если было найдено только два не пересекающихся по узлам пути, то на предпоследнем шаге процедуры не существовало бы требуемого потока. Обычно данная процедура является очень эффективной, поскольку на каждом ее шаге из исходной сети удаляется один узел и, возможно, большое число дуг. Работа данной процедуры показана на следующем примере.

Рассмотрим сеть, состоящую из 8 узлов и 16 дуг (рис. 2.61а). Определим, будет ли сеть разорвана в результате удаления пяти или менее дуг.

Шаг 1. Выбрать произвольный узел, например узел *G*. Можно показать, что между узлом *G* и любым другим узлом расположено не менее пяти путей, не пересекающихся по узлам.

Шаг 2. Исключить узел *G* и все соединенные с ним дуги. Редуцированная сеть изображена на рис. 2.61б. Выбрать произвольный узел, например узел *A*. Можно показать, что между

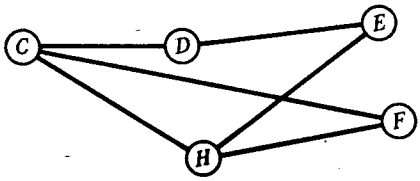


Рис. 2.61г. Сеть после удаления узлов *G*, *A* и *B*.

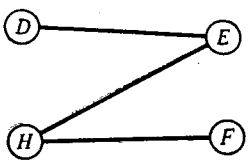


Рис. 2.61д. Сеть после удаления узлов *G*, *A*, *B* и *C*.

узлом *A* и каждым из остальных узлов существует по крайней мере четыре пути, не пересекающихся по узлам.

Шаг 3. Удалить узел *A* и все соединенные с ним дуги. Образованная в результате такой редукции сеть изображена на рис. 2.61в. Выбрать произвольный узел, например узел *B*. Можно показать, что между узлом *B* и каждым из остальных узлов существует по крайней мере три пути, не пересекающихся по узлам.

Шаг 4. Удалить узел *B* и все соединенные с ним дуги. Образованная в результате такой редукции сеть изображена на рис. 2.61г. Выбрать произвольный узел, например узел *C*. Можно показать, что между узлом *C* и каждым из остальных узлов существует не менее двух путей, не пересекающихся по узлам.

Шаг 5. Удалить узел *C* и все соединенные с ним дуги. В результате этого образуется сеть, изображенная на рис. 2.61д.

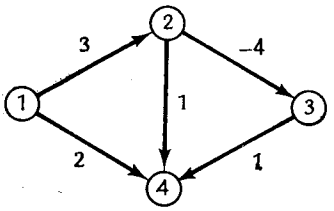
Отметим, что в сети, изображенной на рис. 2.61д, число путей, не пересекающихся по узлам, равно единице. Поскольку на шаге 5 процедура завершается, принятая гипотеза верна.

УПРАЖНЕНИЯ

1. Показать, как метод Дейкстры может быть использован для построения древовидности с заданным корнем.

2. Следует ли заново начать решение задачи, если на одной из промежуточных итераций при определении постоянной пометки была допущена ошибка, которая не была обнаружена до завершения решения задачи?

3. В изображенной ниже сети найти кратчайший путь из узла 1 в узел 4: (а) простым перебором; (б) с помощью метода Дейкстры. Почему эти два решения различаются?



4. Привести несколько примеров практического использования задач о пути максимальной длины. Указать условия, при выполнении которых задача о пути максимальной длины может быть сведена к эквивалентной задаче о кратчайшем пути.

5. При работе каких алгоритмов о кратчайшем пути из числа описанных в настоящей главе вначале определяются длины путей, а затем используются процедуры трассировки для нахождения самих путей? В чем преимущество таких алгоритмов по сравнению с алгоритмами, в которых длины путей и сами пути определяются одновременно?

6. В настоящей главе был описан вариант метода Дейкстры, в котором предполагается, что все параметры дуг неотрицательные. Модифицировать данный алгоритм таким образом, чтобы он был применим к сетям, содержащим дуги с отрицательными параметрами. Какое условие в данном случае имеет решающее значение?

7. Объяснить, почему число итераций в алгоритме Флойда равно числу узлов сети.

8. Могут ли в алгоритме Флойда и методе двойного поиска параметры дуг принимать отрицательные значения?

9. Могут ли быть получены циклы в путях при поиске K кратчайших путей с помощью метода двойного поиска? Что нужно сделать в том случае, когда определяются только пути, не содержащие циклов?

10. Объяснить, почему эффективность алгоритма Флойда и метода двойного поиска может зависеть от порядка нумерации узлов. Рассмотреть каждый из алгоритмов в отдельности.

11. Показать, что метод двойного поиска, примененный к ориентированной бесконтурной сети с дугами (i, j), где $i<j$, сводится к однократному поиску. Какой поиск при этом исключается? Рассмотреть аналогичную задачу для случая, когда дуги (i, j) такие, что $j<i$.

12. (а) Как определить, являются ли два или более решений задачи коммивояжера оптимальными? (б) В каком случае заданное звено не включается в оптимальный маршрут?

13. Объяснить, почему общую задачу коммивояжера нельзя решать непосредственно как задачу о назначениях.

14. В чем заключается роль начальных границ в алгоритме решения задачи коммивояжера?

15. При каких условиях неориентированная дуга, соединяющая узлы i и j, может быть заменена двумя ориентированными дугами (i, j) и (j, i) с той

же пропускной способностью, что и у исходной дуги? В каких случаях такая замена невозможна? Привести соответствующие примеры.

16. Почему начальное решение транспортной задачи, полученное с помощью ПМФ, как правило, лучше начального решения, полученного с помощью метода «северо-западного угла»?

17. Показать, каким образом обычная транспортная задача может быть сведена к задаче о назначениях (большей размерности).

18. Показать, каким образом задача о кратчайшем пути может быть сведена к задаче о перевозках.

19. Почему задачу со штрафом за выполнение поворота нельзя решать с помощью обычного алгоритма поиска кратчайшего пути, сложив предварительно параметр каждой дуги с величиной штрафа?

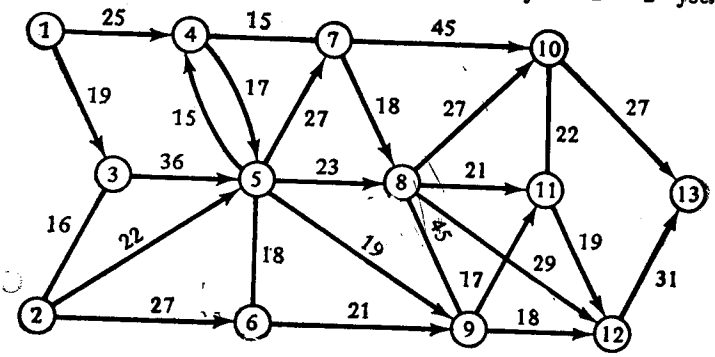
20. Дать математические постановки задач о кратчайшем пути со штрафом за выполнение поворота и без штрафов. Сравнить эти две постановки.

21. На изображенной ниже сети параметр каждой дуги соответствует стоимости единицы потока по этой дуге. Выполняя все вычисления вручную, найти следующие пути:

а) путь минимальной стоимости из узла 1 в узел 13 (используя алгоритм Дейкстры);

б) пути минимальной стоимости между всеми парами узлов;

в) $K=3$ путей минимальной стоимости из узла 2 в узел 13.

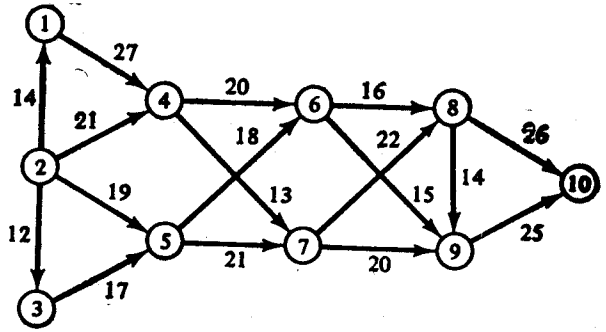


22. На изображенной ниже сети параметр каждой дуги соответствует верхней границе потока по этой дуге. Выполняя все вычисления вручную, найти:

а) максимальный поток из узла 2 в узел 10;

б) цепь с максимальной пропускной способностью из узла 2 в узел 10;

в) максимальный поток между всеми парами узлов.



23. В упражнении 22 (а) проверить правильность найденной величины максимального потока с помощью теоремы о максимальном потоке и минимальном разрезе.

24. Решить задачи из упражнения 21 с помощью соответствующих машинных программ.

25. Предположим, что в упражнении 21 стоимость единицы потока по дуге (12, 13) уменьшена вдвое. Можно ли найти путь минимальной стоимости, используя полученные ранее результаты?

26. Предположим, что в упражнении 21 дуга (4, 5) должна быть включена в путь минимальной стоимости. Найти решение данной задачи. Как изменится процедура поиска решения, если знать о данном ограничении заранее?

27. В XXV в. Бэк Роджерс отправляется в космическое путешествие. Он хочет побывать на пяти астероидах, причем на каждом из них только один раз. До каждого астероида можно долететь только по определенным космическим трассам. В приводимой ниже таблице даны расстояния (в млн. миль) между всеми астероидами, вычисленные по заданным космическим трассам.

	Земля	Черный спутник	Форт Судьбы	Замок неожиданностей	Астероид Ужасов
Земля	—	52	29	35	29
Черный спутник	46	—	62	21	44
Форт Судьбы	29	62	—	50	27
Замок неожиданностей	37	23	50	—	—
Астероид Ужасов	24	44	27	—	—

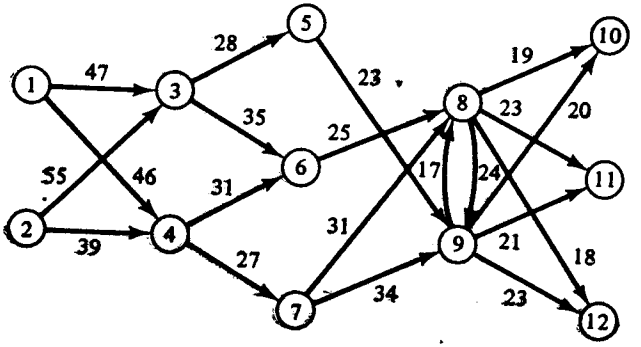
Матрица расстояний не является симметричной, поскольку возможны отклонения от курса. Найти кратчайший маршрут для Бэка, если он начинает свой путь с Земли и благополучно возвращается домой.

28. В изображенной ниже сети параметр каждой дуги соответствует стоимости единицы потока по данной дуге. Выполняя все вычисления вручную, найти:

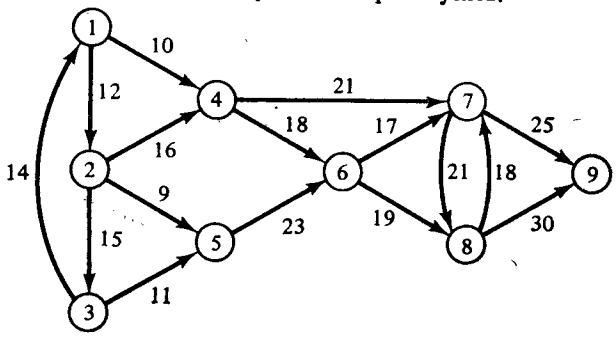
а) путь минимальной стоимости из узла 1 в узел 10;

б) пути минимальной стоимости из узлов 1 и 2 в узлы 10, 11 и 12;

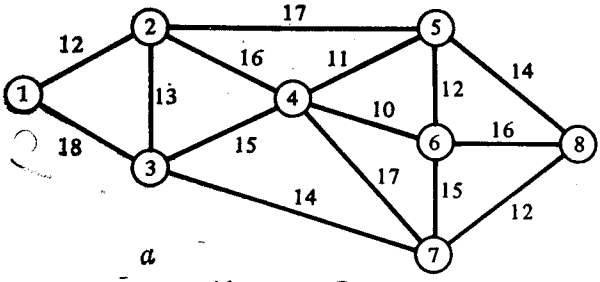
в) $K=2$ кратчайших путей из узла 1 в узел 11.



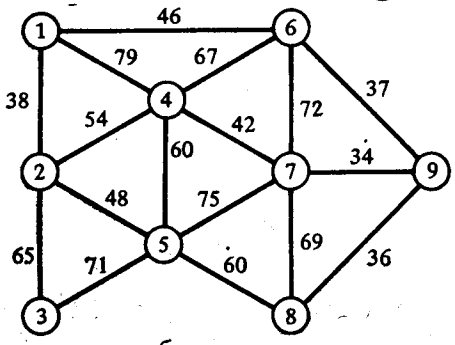
29. В изображенной ниже сети параметры дуг соответствуют верхним границам потоков по ним. Выполняя все вычисления вручную, найти:
а) максимальный поток из узла 1 в узел 9;
б) цепь с максимальной пропускной способностью из узла 1 в узел 8;
в) максимальный поток между всеми парами узлов.



30. Построить минимальные остовные деревья изображенных ниже сетей.



а

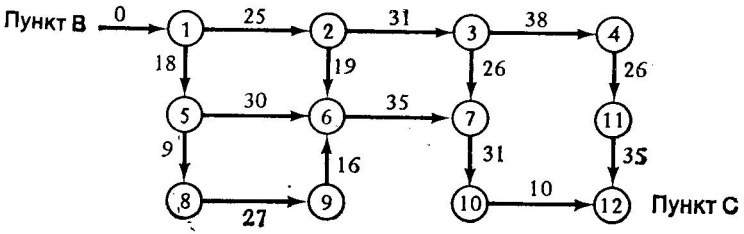


б

31. Построить древовидный остов сети из упражнения 29.

32. Крупная фирма, производящая конвейеры, проектирует систему конвейеров, которая соединит пункт B с пунктом C. Ниже дается возможная конфигурация системы. Числа, приписанные дугам, соответствуют стоимостям сооружения соответствующих участков конвейера. Если в некоторой узловой точке должен выполняться поворот, то для сооружения соответствующего узла требуются дополнительные расходы. Для узлов 1, 2, 6, 10 и 9

эти расходы составляют 1 ед., а для узлов 3, 5, 8, 7 и 4—2 ед. Найти маршрут из пункта B в пункт C, имеющий минимальную стоимость.

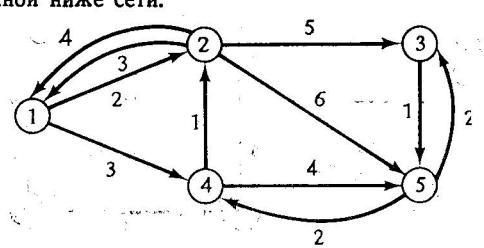


33. Resto Manufactoring Company производит обеденные столы и продает их Техасскому университету. Можно выделить следующие три основные этапа изготовления столов: сборка, окраска и сушка. Сборка может производиться в двух различных цехах — A и B. Поскольку цеха оснащены различным оборудованием, то общее время сборки в них не одинаковое. Аналогично окраска может производиться в трех различных цехах — C, D и E, которые оснащены различным красильным оборудованием. Планировка предприятия такова, что ни один из столов, собранных в цехе A, не поступает для окраски в цех E и ни один из столов, собранных в цехе B, не поступает для окраски в цех C. Средняя продолжительность (в минутах) выполнения каждой операции приводится в таблице. Требуется найти оптимальную последовательность расположения цехов, т. е. такую последовательность, при которой время изготовления одного стола будет минимальным. (Продолжительность сушки равна 60 мин.)

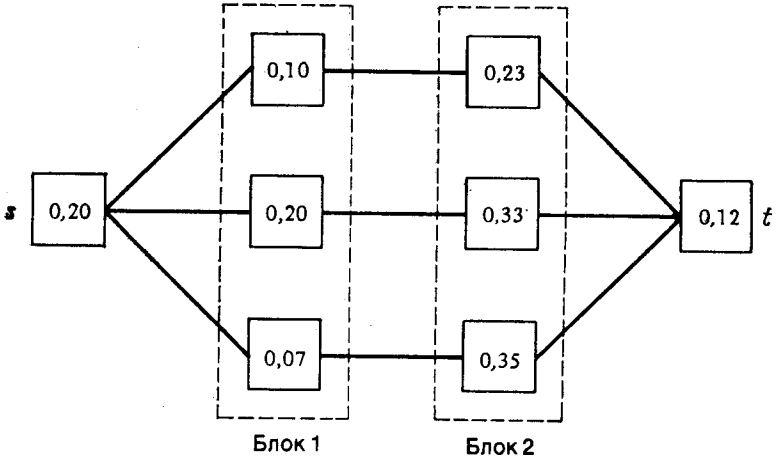
	Цеха				
Операции	*A*	*B*	*C*	*D*	*E*
Сборка	120	140	—	—	—
Окраска	—	—	50	30	20

34. ABC Taxi Service Company нужно составить план замены оборудования на ближайшие 4 года. На данный период ожидаются следующие расходы и амортизационные стоимости. Существующая цена автомобиля составляет 7000 долл. Ожидается ежегодное увеличение цен на автомобили на 10%. В первые два года автомобили изнашиваются на 25%, а в последующие два года — на 15%. Издержки на техническое обслуживание и текущий ремонт в ближайшие 4 года составят 1200, 1500, 1900 и 2400 долл. соответственно. Автомобиль можно заменять каждый год или каждые 2, 3 или 4 года. Составить план замены автомобилей, при котором общие издержки на ближайшие 4 года были бы минимальными.

35. Используя алгоритм Дейкстры, найти кратчайший путь из узла 1 в узел 5 изображенной ниже сети.

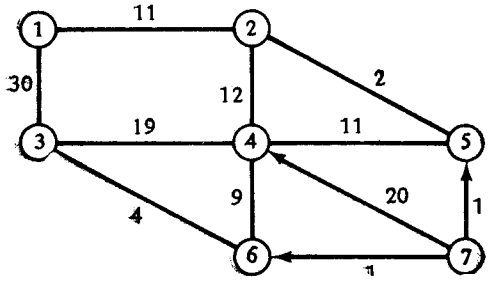


36. Устройство состоит из восьми узлов. Вероятность выхода из строя каждого узла показана на рисунке. Устройство работает правильно, если нормально функционирует каждый узел некоторого пути, ведущего из s в t. Предположим, что если после работы блока 1 используется блок 2, то блок 1 не может быть снова использован. Сформулировать задачу минимизации вероятности выхода устройства из строя как задачу о кратчайшем пути.

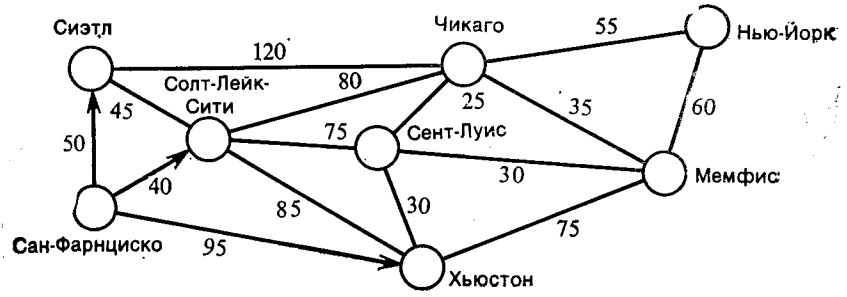


37. Промышленная компания планирует производство кондиционеров, состоящих из трех основных частей: корпуса, вентилятора и мотора. Затраты (в долл.) на оборудование предприятия станками, производящими корпуса, вентиляторы и моторы, составляют 20 000, 50 000 и 80 000 соответственно. Однако если вначале наладить выпуск вентиляторов, то затраты на оборудование предприятия станками, производящими корпуса и моторы, будут уменьшены на 5%. Если вначале пустить в производство моторы, то затраты на оснащение предприятия станками двух других типов уменьшатся на 10%. Если же в первую очередь будет налажен выпуск корпусов, то остальные затраты уменьшатся на 5%. После того как будет налажен выпуск двух компонентов, затраты на производство третьего компонента дополнительно уменьшатся на 5%. Построить сеть с одним источником и одним стоком. Указать, чему соответствуют узлы, дуги и параметры дуг. Сформулировать и решить соответствующую задачу о кратчайшем пути.

38. Найти кратчайший путь между каждой парой узлов приведенной ниже сети.

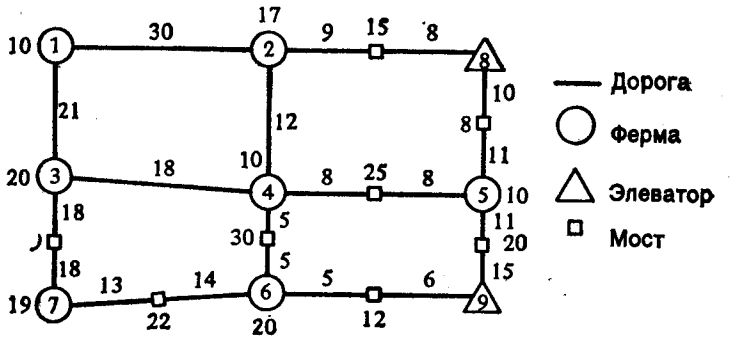


39. Почтовому ведомству США требуется определить наиболее экономные маршруты доставки писем из заданного города в каждый другой город, отмеченный на изображенной ниже карте. Числа, приписанные звеньям сети, указывают приближенные транспортные затраты на выполнение соответствующих рейсов. Найти пути минимальной стоимости.



40. В сети из упражнения 39 найти кратчайший путь, начинающийся и заканчивающийся в Хьюстоне, шт. Техас, и проходящий через каждый из остальных городов ровно один раз.

41. На изображенном ниже графе указано размещение ферм и элеваторов в одном из сельских районов. Население данного района занято главным образом в сельском хозяйстве. Поскольку на вес груза, перевозимый через мосты, накладываются ограничения, то некоторые пользователи системы не имеют возможности подъехать к элеватору кратчайшим путем. Числа, приписанные дугам, соответствуют длинам дорог. Пропускная способность каждого моста и производительность каждой фермы указаны рядом с символами, используемыми для обозначения мостов и ферм соответственно. Предложить метод, позволяющий определить, какие из мостов следует заменить на новые при условии, что бюджет позволяет провести замену двух мостов.



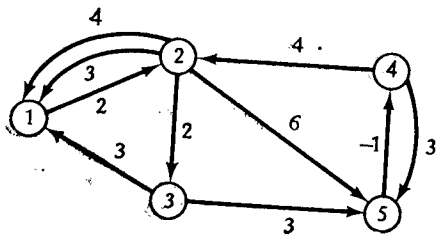
42. Пять небольших городов одного округа соединены между собой дорогами, нуждающимися в ремонте и реконструкции. Расстояния между городами (в милях) приводятся в таблице.

Начальнику окружного отдела автомобильных дорог надо определить, какие три дороги следует отремонтировать в первую очередь. В текущий момент состояние всех дорог приблизительно одинаковое. Наиболее интенсивное движение наблюдается на дорогах между городами 1 и 5, 2 и 5, 3 и 4. Общее

число поездок распределяется следующим образом: 70% — по кратчайшему пути, 20% — по второму кратчайшему пути, 10% — по третьему кратчайшему пути. Найти наиболее предпочтительное решение при условии, что стоимость ремонта одной мили дороги постоянная на всех участках.

Город i \ Город j	1	2	3	4	5
1	0	4	2	—	—
2	—	0	7	5	—
3	—	5	0	4	6
4	—	3	—	0	2
5	—	—	—	—	0

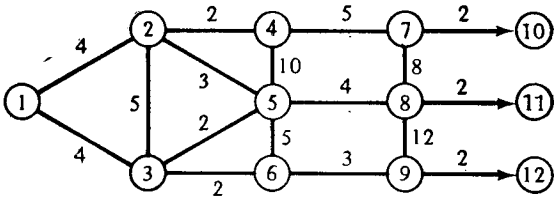
43. Найти три кратчайших пути из узла 1 в каждый из остальных узлов изображенной ниже сети.



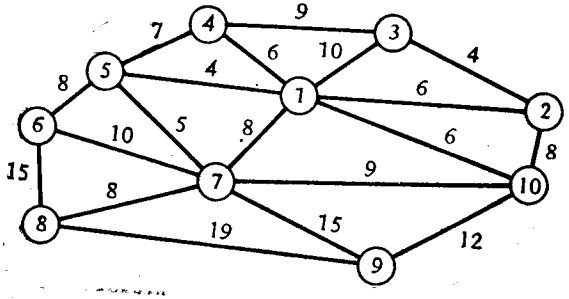
44. Проект исследований и научно-технических разработок состоит из четырех задач, темпы решения которых зависят от выделенной для этой цели суммы (см. приведенную ниже таблицу). Найти наилучшую стратегию, при которой проект будет завершен не позднее, чем через 18 месяцев, а расходы не превысят 35 000 долл. Найти вторую наилучшую стратегию

	Задача A		Задача B		Задача C		Задача D	
Темп	Длит. (в мес.)	Затраты (в долл.)	Длит. (в мес.)	Затраты (в долл.)	Длит. (в мес.)	Затраты (в долл.)	Длит. (в мес.)	Затраты (в долл.)
Низкий	6	6 000	5	8 000	6	2 000	7	10 000
Средний	4	8 000	3	9 000	3	3 000	5	12 000
Высокий	3	9 000	2	10 000	1	5 000	3	18 000

45. С помощью программы, реализующей алгоритм двойного поиска, найти четыре кратчайших пути приведенной ниже сети. Как станет видно, некоторые из этих путей содержат циклы. Модифицировать подпрограмму TRACE ФОРТРАН-программы таким образом, чтобы алгоритм завершал построение последовательности узлов, как только будет обнаружен цикл. С помощью модифицированной программы найти пути, не содержащие циклов. Узел 1 является источником, узлы 10, 11 и 12 — стоками.



46. Electrode TV Cable Company намеревается начать работы в рамках проекта «Нетвок-Сити». Компания арендовала место, обозначенное узлом 1, для основного центра связи и определила расположение дополнительных центров (узлы 2, 3, ..., 10) с учетом того, что плотность населения равна 500 человек на квадратную милю. Необходимо проложить кабель к каждому дополнительному центру, причем в целях экономии компания стремится израсходовать кабеля как можно меньше. Определить наименьшую длину кабеля, необходимую для выполнения всех работ, и построить минимальное остовное дерево. Расстояния задаются в милях.



47. Федеральное управление автомобильных дорог планирует строительство дорог, которые соединят столицы пяти соседних штатов. Все города должны быть соединены друг с другом либо непосредственно, либо дорогой, проходящей через другой город. Затраты (в тыс. долл.) на строительство дорог приводятся в таблице. Какие дороги следует построить?

Из города \ В город	*A*	*B*	*C*	*D*	*E*
A	—	10	40	60	80
B	10	—	80	70	60
C	40	80	—	20	30
D	60	70	20	—	25
E	80	60	30	25	—

48. Молодой человек намеревается вложить деньги в производительный капитал сроком на 6 лет. Он может купить акции и хранить их в течение 6 лет, или заменить их в конце пятого года новыми акциями и хранить последние еще один год, или же хранить акции 4 года, затем заменить их новыми акциями и хранить их в течение пятого и последнего годов. Он может также заменять акции в конце каждого года. Сформулировать задачу, аналогичную задаче замены оборудования, и найти оптимальное решение.

49. Директор небольшой фирмы намеревается вложить 12 000 долл. в осуществление трех проектов — 1, 2 и 3. Минимальные суммы, которые она должна вложить в эти проекты, составляют 3000, 2000 и 1000 долл. соответственно. Сверх указанных сумм можно дополнительно вкладывать любые суммы, кратные 1000 долл. Построить сетевую модель для данной задачи.

50. Rail-Riders, Inc., являющаяся основным производителем погрузочно-разгрузочного оборудования, проектирует высокоскоростную систему перевозки пассажиров между шестью различными пунктами аэропорта в Спейс-Сити, США. Значение каждого элемента матрицы равно расстоянию (в футах) между соответствующей парой пунктов, вычисленному по кратчайшему пути, соединяющему их.

	1	2	3	4	5	6
1	—	870	910	796	860	712
2	870	—	1100	560	430	630
3	900	1100	—	270	600	
4	796	560	270	—	300	850
5	850	430	600	300	—	370
6	712	630	250	850	370	—

а) Инструкции по технике безопасности не позволяют, чтобы пути высокоскоростной системы пересекались. Найти длину пути минимальной стоимости, имеющего форму цикла, проходящего через все пункты. Стоимость одного фута пути постоянна и равна 100 долл.

б) Rail-Riders намеревается спроектировать другую систему, в которой центральный распределительный пункт соединяется со всеми остальными пунктами путями с двусторонним движением. Устройство данной системы похоже на колесо велосипеда, а центральная станция играет роль «втулки». В следующей таблице даны расстояния между втулкой и каждым из шести пунктов:

	1	2	3	4	5	6
Втулка	150	175	210	180	175	190

Составить проект системы минимальной стоимости и сравнить полученные результаты с решением задачи из п. (а).

51. Задача транспортировки угля. Четыре шахты поставляют каменный уголь пяти газоперерабатывающим заводам, расположенным на юге Соединенных Штатов. Расходы на транспортировку угля, задаваемые матрицей стоимостей (в тыс. долл.), включают плату за его перевозку по железным

	Завод J					
	1	2	3	4	5	Предложение (в т)
Шахта I 1	10	9	8	9	7	500
2	4	8	12	7	9	900
3	6	3	4	2	8	700
4	7	6	5	4	3	600
Спрос (в т)	400	700	400	500	700	

и автомобильным дорогам, а также плату за погрузочно-разгрузочные работы. Построить схему транспортировки угля, имеющую минимальную стоимость.

52. Цех получил заказ на изготовление четырех изделий. В цеху имеется пять станков. Стоимость (в долл.) изготовления изделий на каждом станке задается матрицей. Перед начальником цеха стоит задача поиска такого распределения изделий по станкам, при котором минимизируются общие затраты.

Станки \ Работы	1	2	3	4
1	12	14	12	18
2	16	10	10	16
3	10	8	8	14
4	8	12	16	12
5	14	10	18	14

53. Предприятия, производящие водные лыжи, имеют четыре общегосударственные оптовые базы, с которых лыжи распределяются по пяти различным магазинам. Найти имеющую минимальную стоимость схему транспортировки лыж при условии, что базы располагают 100, 200, 300 и 400 парами лыж, а магазинам требуется 200, 250, 300, 100 и 150 пар соответственно. Затраты на транспортировку (в тыс. долл.) задаются матрицей стоимостей.

Оптовые базы \ Магазины	*A*	*B*	*C*	*D*	*E*
1	5	7	3	15	9
2	6	2	8	5	6
3	3	9	8	2	2
4	7	6	8	7	4

54. При добыче урана приблизительно 0,711% его составляет U^{235}, а остальную часть — U^{238}. С рудника уран перевозится на три предприятия, где он обогащается, т. е. процент содержания U^{235} возрастает, в результате чего образуется высокорадиоактивное вещество шестифтористый уран (UF_6). Затем его доставляют десяти предприятиям, на которых оно должно быть переработано в топливо для ядерных реакторов. Правительство, исходя из соображений безопасности, разработало десятибальную шкалу, характеризующую степень риска при транспортировке шестифтористого урана по различным маршрутам (число 10 соответствует наибольшему риску). Соответствующие значения, а также величины спроса и предложения приводятся в таблице. Найти схему транспортировки, при которой общий риск минимален.

	Предложение (в фунтах)									
	1	2	3	4	5	6	7	8	9	10
A	9	8	7	4	6	3	8	6	5	2
B	1	7	9	10	8	3	6	7	5	4
C	3	3	5	7	7	8	9	1	2	3
Спрос (в фунтах)	100	100	300	150	200	90	110	50	50	100

55. Джон Джонс прошлым летом ездил в Пуэрто-Рико. Его виза позволяла ему посетить лишь пять городов, по одному разу каждый. Поскольку раньше он никогда не был в Пуэрто-Рико, он отправился в бюро путешествий и попросил посоветовать ему, какие места лучше всего посетить. Ему ответили, что нельзя уезжать из Пуэрто-Рико, не побывав в городах Кагуас, Понсе, Маягуэс и Аресиво. Он вернулся в свою квартиру в Сан-Хуане и составил матрицу расстояний маршрутов из Сан-Хуана в каждый из перечисленных выше городов. Зная, что бензин очень дорогой, он стал искать кратчайший маршрут, проходящий через все эти города и ведущий обратно в Сан-Хуан. Определить этот маршрут. Значения элементов матрицы расстояний выражены в милях.

	С-Х	К	П	А	М
С-Х	—	17	64	48	94
К	17	—	47	58	94
П	64	47	—	52	47
А	48	58	52	—	47
М	94	94	47	47	—

56. Рассмотрим следующую задачу производственного календарного планирования. На станке изготовляют однотипные изделия различных цветов, причем необходимо изготовить по одному изделию каждого цвета. Издержки на наладочные работы включают в себя только расходы, связанные с изменением цвета выпускаемых изделий. Требуется составить такой производственный календарный план, при котором суммарные издержки на наладочные работы минимальны. Расходы, связанные с изменением цвета выпускаемого изделия, зависят от цвета изделия, которое было изготовлено последним. Стоимость (в долл.) перехода от цвета i к цвету j задается элементами c_{ij} матрицы C. Найти оптимальный производственный календарный план, т. е. план, минимизирующий расходы на перенастройку станка.

$C = [c_{ij}] =$

	1	2	3	4	5	6	7	8	9	10
1	—	1	2	3	4	5	6	7	8	9
2	2	—	3	4	5	6	7	8	9	1
3	3	4	—	4	5	6	7	8	9	1
4	4	5	6	—	7	9	8	1	2	3
5	5	6	7	8	—	9	1	2	3	4
6	6	7	8	9	1	—	2	3	4	5
7	7	8	9	1	2	3	—	4	5	6
8	8	9	1	2	3	4	5	—	6	7
9	9	1	2	3	4	5	6	7	—	8
10	1	2	3	4	5	6	7	8	9	—

57. Заготовка должна пройти пять видов обработки. Время от времени в зависимости от того, какую операцию требуется выполнить, происходит автоматическая смена режима работы режущего механизма используемого станка. Некоторые операции должны предшествовать другим операциям. Кроме того, некоторые операции требуют одних и тех же наладочных работ.

При выполнении операций в произвольной повторяющейся последовательности наладочные работы будут повторяться. Требуется минимизировать общее время обработки путем составления оптимального расписания выполнения операций, т. е. такого расписания, при котором число выполняемых наладочных работ будет минимальным. Виды обработки и способы расположения режущего инструмента указаны в табл. 1[1]. В табл. 2 даются времена проведения наладочных работ, необходимых для выполнения операции i после операции j, и указаны ограничения на последовательность выполнения операций. Операция 1 соответствует начальным наладочным работам. Найти оптимальное по времени решение.

Таблица 1

Операция	Расположение режущего инструмента			
	A	*B*	*C*	*D*
2	Фрезерование			
3	Разметка			
4		Разметка		
5		Высверливание		
6		Нарезание резьбы		
7			Разметка	
8			Высверливание	
9				Разметка Высверливание направляющего отверстия

Таблица 2

	Расположение режущего инструмента								
	A		*B*			*C*		*D*	
Операция	2	3	4	5	6	7	8	9	10
1	20	×	25	×	×	30	×	35	×
2	×	20	20	20	20	40	40	45	45
3	×	×	25	20	25	45	45	30	30
4	20	25	×	30	×	30	30	35	35
5	25	25	×	×	30	20	30	35	35
6	25	25	×	×	×	30	20	35	35
7	50	50	10	20	30	×	20	40	40
8	50	50	20	20	20	×	×	40	40
9	25	25	20	10	20	20	40	×	30
10	25	25	30	30	30	20	25	×	×

58. Решить задачу коммивояжера, в которой число городов равно 6, а плата (в долл.) за проезд из города i в город j задается следующей матрицей стоимостей:

[1] Тип обработки и расположение режущего инструмента определяют выполняемую операцию. — *Прим. перев.*

$$C = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & — & 27 & 43 & 16 & 30 & 26 \\ 2 & 7 & — & 16 & 1 & 30 & 25 \\ 3 & 20 & 13 & — & 35 & 5 & 0 \\ 4 & 21 & 16 & 25 & — & 18 & 18 \\ 5 & 12 & 46 & 27 & 48 & — & 5 \\ 6 & 23 & 5 & 5 & 9 & 5 & — \end{array}$$

59. Рассмотрим одно из промышленных предприятий, расположенных в Мичигане. Оно состоит из трех цехов (формовки, окраски и сборки), между которыми расположены дороги и подъездные пути. В цехах находится шесть заводских контор (узлы 2, 3, 4, 5, 6 и 7), в которых следует вести уборку в третью смену. Уборочная машина со всем инвентарем для уборки помещений находится в пункте 1. Инженер-технолог, выполняющий работу по учету постоянных издержек, с помощью нормативного справочника определил время, необходимое для уборки контор и перехода из одной конторы в другую. Однако будучи дипломированным инженером, он решает найти наиболее экономный маршрут. Расстояния между пунктами заданы в табл. 1. Определить наиболее экономный маршрут и его длину.

Таблица 1

Неориентированная дуга (i, j)	Расстояние (в ярдах)
(1, 2)	40
(1, 3)	90
(1, 4)	120
(1, 7)	90
(2, 3)	130
(3, 4)	80
(4, 5)	70
(4, 7)	80
(5, 6)	60
(6, 7)	140

Инженер-технолог решает, что погода в зимние месяцы в Мичигане суровая и необходимо исследовать возможность использования внешних дорог. Кроме того, правила техники безопасности ограничивают скорость передвижения уборочной машины по внутренним дорогам до 2,72 миль/ч, а по внешним дорогам можно передвигаться со скоростью 5,09 миль/ч. В табл. 2 ука-

Таблица 2

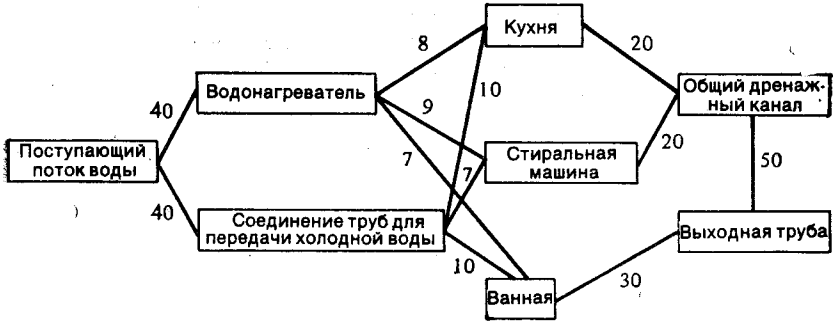
Дополнительные внешние дороги		Новые внешние дороги	
Неориентированная дуга (i, j)	Расстояние (в ярдах)	Неориентированная дуга (i, j)	Расстояние (в ярдах)
(1, 4)	140	(3, 5)	190
(1, 7)	180	(4, 6)	110
(6, 7)	150		

заны новые и дополнительные внешние дороги и приведены соответствующие расстояния. Определить, изменится ли построенный ранее наиболее экономный маршрут в результате использования новых или дополнительных дорог. Какова длина нового маршрута и время, необходимое для его прохождения?

60. С семи платформ, находящихся в море, к конечной станции нефтепровода, расположенной в 50 милях от них, перекачивается неочищенная нефть. Производительность скважин для каждой платформы различная. Управляющий конечной станцией нефтепровода дает указание установить максимально допустимый поток нефти на каждой линии, ведущей к станции. Максимальный объем (в баррелях) неочищенной нефти, который может быть перекачан с платформы i к платформе j, задается матрицей (элементы которой следует домножить на 10^5). Найти максимальный поток нефти, который можно перекачивать с платформы 1 к конечной станции T.

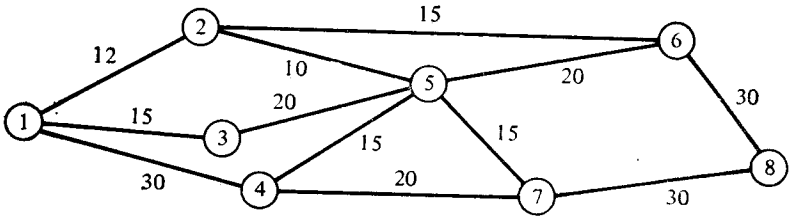
	1	2	3	4	5	6	7	T
1	—	6	4	1	—	—	—	—
2	—	—	—	—	4	—	—	—
3	—	—	—	1	1	3	—	—
4	—	—	—	—	—	4	—	—
5	—	—	—	—	—	—	4	—
6	—	—	—	—	—	—	9	4
7	—	—	—	—	—	—	—	6

61. Фирма, занимающаяся прокладкой труб, доставила необходимое количество труб для сооружения водосточной системы в одном доме. Фирме известна пропускная способность каждой трубы. Для того чтобы определить оптимальный размер еще не установленной трубы, ведущей к главной водосточной линии, и тем самым минимизировать затраты, фирме необходимо знать максимальный объем воды, который может поступать из дома. Ниже дается сетевое представление водосточной системы дома. Числа, приписанные дугам, указывают максимальную пропускную способность (в кубических дюймах) каждой трубы. Определить максимальный поток и соответствующие размеры трубы. Указать недостатки данной постановки.



62. Недавно на юге нефтедобывающей страны было освоено месторождение нефти. В результате этого в одном из ее южных городов, расположенных неподалеку от месторождения, был построен один единственный в стране

нефтеочистительный завод, из которого очищенная нефть в цистернах развозилась по всей стране. Однако последние исследования других способов транспортировки нефти показали, что в конечном итоге значительно экономичнее было бы построить еще два нефтеочистительных завода на севере и западе страны и перекачивать к ним нефть по нефтепроводам. Для того чтобы нефтеочистительные заводы работали на полную мощность, необходимо определить максимальный поток нефти, который может поступать к ним. При этом предполагается, что известны ограничения, определяемые физико-географическими и другими условиями, на пропускную способность участков нефтепровода, расположенных между соседними городами. Пропускные способности (в тыс. кубических футах) этих участков указаны на рисунке. Определить максимально возможный поток между всеми парами пунктов.



Узел 1 — месторождение нефти; узлы 2, 3, 4, 5 — города; узлы 6, 7 — нефтеочистные заводы; узел 8 — распределительный центр.

63. Семь городов используют единую систему связи. Каждый элемент матрицы равен максимальному числу вызовов, которые одновременно могут обслуживаться линией, соединяющей соответствующую пару городов. Для каждой пары городов найти максимальное число вызовов, которые могут обслуживаться одновременно.

Город i \ Город j	1	2	3	4	5	6	7
1	—	10	—	2	—	—	—
2	10	—	7	—	3	—	—
3	—	7	—	—	—	9	—
4	2	—	—	—	8	—	6
5	—	3	—	8	—	11	6
6	—	—	9	—	11	—	4
7	—	—	—	6	6	4	—

ЛИТЕРАТУРА

1. Barr R. S., Glover, F., Klingman D., An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes, *Mathematical Programming*, 7 (1), 60—87 (1974).
2. Bellmore M., Bennington G., Lubore S., A Multivechicle Tanker Scheduling Problem, *Transportation Science*, 5, 36—47 (1971).
3. Bennington G. E., An Efficient Minimal Cost Flow Algorithm, *Management Science*, 19, 1042—1051 (1973).
4. Bennington G. E., Applying Network Analysis, *Journal of Industrial Engineering* 6 (1), 17—25 (January 1974). Portions reproduced by permission of the authors and the American Institute of Industrial Engineers.

5. Bradley G. H., Survey of Deterministic Networks, *AIIE Transactions,* 7, 222—234 (1975).
6. Charnes A., Glover F., Karney D., Klingman D., Stutz J., Past, Present and Future of Large Scale Transshipment Computer Codes and Applications, Center for Cybernetic Studies, University of Texas, Report CS 131, July 1973 (rev. October 1973) (to appear in Computers and Operations Research).
7. Chien R. T., Synthesis of a Communication Net, *Journal of Research and Development,* 4, 311—320 (1960).
8. Dantzig G. G., Blattner W., Rao M. R., Finding a Cycle in a Graph with a Minimum Cost to Time Ratio with Applications to a Ship Routing Problem, Theory of Graphs International Symposium, Dunod, Paris, and Gordon and Breach, New York, 1966, pp. 77—83.
9. Dantzig G., Application of the Simplex Method to a Transportation Problem, in: Activity Analysis of Production and Allocation, ed. T. C. Koopmans, Wiley, Inc., New York, 1951.
10. Dei Rossi J. A., Heiser R. S., King N. S., A Cost Analysis of Minimum Distance TV Networking for Broadcasting Medical Information, RM-6204-NLM, RAND Corporation, Santa Monica, Calif., February 1970.
11. Dijkstra E. W., A Note on Two Problems in Connection with Graphs, *Numerishe Mathematik,* 1, 269—271 (1959).
12. Dreyfus S. E., An Appraisal of Some Shortest Path Algorithsm, *Operations Research,* 17, 395—412 (1969).
13. Dreyfus S. E., A Generalized Equipment Replacement Study, *Journal of the Society for Industrial and Applied Mathematics,* 8, 425—435 (1960).
14. Evans J. R., Network Modeling in Production Planning, Proceedings of the 1978 AIIE National Systems Conference, Montreal, Canada, 1978.
15. Ford L. R., Fulkerson D. R., Flows in Networks, Princeton University Press, Princeton, N. J., 1962. [Имеется перевод: Форд Л. Р., Фалкерсон Д. Потоки в сетях. — М.: Мир, 1966.]
16. Ford L. R., Fulkerson D. R., A Primal-Dual Algorithm for the Capacitated Hitchcock Problem, *Naval Research Logistics Quarterly,* 4, (1), 47—54 (1957).
17. Fox B., Vehicle Scheduling and Driver Run Cutting: RUCUS Package Overview, M71-58, The Mitre Corporation, McClean, Va., September 1971.
18. Frank H., Frisch I. T., Communication, Transmission and Transportation Networks, Addison-Wesley Publ. Co., Inc., Reading, Mass., 1971. [Имеется перевод: Фрэнк Г., Фриш И. Сети, связь и потолки. — М.: Связь, 1978.]
19. Garfinkel R. S., Nemhauser G. L., Integer Programming, Wiley, Inc., New York, 1972.
20. Glover F., Klingman D., New Advances in the Solution of Large Scale Network and Network-related Problems, Colloquia Mathematica Societatis Janos Bolyai, Vol. 12, North-Halland Publ. Co., Amsterdam, 1975.
21. Glover F., Klingman D., A Note on Computational Simplification in Solving Generalized Transportation Problems, *Transportation Science,* 7, 351—361 (1973).
22. Glover F., Klingman D., Capsule View of Future Developments on Large Scale Network and Network-related Problems, CCS238, University of Texas at Austin, 1975.
23. Glover F., Klingman D., Network Application in Industry and Government, *AIIE Transactions,* 9 (4), (1977).
24. Glover F., Hultz J., Klingman D., Improved Computer Based Planning Techniques — Part I, *Interfaces,* 8 (4) (August 1978).
25. Gomory R. E., Hu T. C., Multi-terminal Network Flows, *SIAM Journal of Applied Mathematics,* 9, 551—571 (1971).
26. Held M., Karp R., The Travelling Salesman Problem and Minimum Spanning Tress, *Operations Research,* 18, (6), 1138—1162 (1970).

27. Held M., Karp R., The Travelling Salesman Problem and Minimum Spanning Trees: Part II, *Mathematical Programming*, 1, 6—25 (1971).
28. Heller I., Tompkins C. B., An Extension of a Theorem of Dantzigs, in: Linear Inequalities and Related Systems, eds. H. Kuhn, A. W. Tucker, Princeton University Press, Princeton, N. J., 1956.
29. Hitchcock F. L., The Distribution of a Product from Several Sources to Numerous Localities, *Journal of Mathematics and Physics*, 20, 224—230 (1941).
30. Hoffman A. J., Winograd S., Finding All Shortest Distances in a Directed Network, *IBM Journal of Research and Development*, 16, 412—414 (1972).
31. Hu T. C., Integer Programming and Network Flows, Addison-Wesley Publ. Co., Inc., Reading, Mass., 1969. [Имеется перевод: Ху Т. Целочисленное программирование и потоки в сетях. — М.: Мир, 1974.]
32. Hu T. C., The Maximum Capacity Route Problem, *Operations Research*, 9, 898—900 (1961).
33. Jacobs W. W., The Caterer Problem, *Naval Research Logistics Quarterly*, 1, 154—165 (1954).
34. Klein M., A Primal Method Cost Flows, *Management Science*, 14 (3), 205—220 (1967).
35. Klingman D., Napier A., Stutz J., NETGEN — A Program for Generating Large Scale (Un) Capacitated Assignment, Transportation and Minimum Cost Flow Network Problems, *Management Science*, 20 (5), 814—822 (1974).
36. Kleitman D. H., reported by H. Frank, I. Frisch in: Network Analysis, *Scientific American*, 223, 94—103 (July 1970).
37. Klingman D. H., Hultz J., Solving Constrained Generalized Network Problems, Research Report CCS 257, Center for Cybernetic Studies, The University of Texas at Austin, 1976.
38. Koopmans T. C., Optimum Utilization of the Transportation System, Proceedings of the International Statistical Conferences, Washington, D. C., 1947.
39. Kuhn H. W., The Hungarian Method for the Assignment Problem, *Naval Research Logistics Quarterly*, 2, 83—97 (1955).
40. Lawler E. L., A Solvable Case of the Traveling Salesman Problem, *Mathematical Programming*, 1, 267—269 (1971).
41. Little J. D. C., et al., An Algorithm for the Traveling Salesman Problem, *Operations Research* 11 (5), 972—989 (1963).
42. Iri M., Network Flow, Transportation and Scheduling, Academic Press, Inc., New York, 1969.
43. Minieka E. T., Shier D. R., A Note on an Algebra for the K Best Routes in a Network, *Journal of the IMA*, 11, 145—149 (1973).
44. Pollack M., The Maximum Capacity Route through a Network, *Operations Research*, 8, 733—736 (1960).
45. Ross G. T., Soland R. M., A Branch and Bound Algorithm for the Generalized Assignment Problem, *Mathematical Programming*, 8, 91—104 (1975).
46. Saltman R. G., Bolotsky G. R., Ruthberg Z. G., Heuristic Cost Optimization of the Federal Telpak Network, National Bureau of Standards Technical Note 787, 1973.
47. Shier D. A., Iterative Methods for Determining the K Shortest Paths in a Network, *Networks*, 6, 205—230 (1976).
48. Shier D. A., Computational Experience with an Algorithm for Finding the K Shortest Paths in a Network, *Journal of Research, National Bureau of Standards*, 78B, 139—165 (July—September 1974).
49. Smythe W. R., Johnson L., Introduction to Linear Programming with Applications, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1966.
50. Williams T. A., White G. P., A Note on Yen's Algorithm for Finding the Length of All Shortest Paths in N-Node Nonnegative Distance Networks,

Journal of the Association of Computing Machinery, **20** (3), 389—390 (July 1973).
51. Veinott A. F., Jr., Wagner H. M., Optimal Capacity Scheduling I, *Operations Research,* **10**, 518—522 (1962).
52. Zadeh N., Theoretical Efficiency and Partial Equivalence of Minimum Cost Flow Algorithms: A Bad Network Problem for the Simplex Method, Operations Research Center, University of California at Berkeley, ORC 72-7, March 1972.
53. Zahn C. T., Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters, *IEEE Transactions on Computers,* **C-20** (1), 68—86 (January 1971).
54. Zangwill W., A Backlogging Model and a Multi-echelon Model of a Dynamic Economic Lot Size Production System — A Network Approach, *Management Science,* **15**, 506—527 (1969).

Глава 3

АЛГОРИТМ ДЕФЕКТА. ОБОБЩЕННЫЙ АНАЛИЗ ДЕТЕРМИНИРОВАННЫХ СЕТЕЙ С ОГРАНИЧЕННОЙ ПРОПУСКНОЙ СПОСОБНОСТЬЮ

Несомненно существует лучший
способ сделать это, и если искать
достаточно долго, то мы обязательно найдем его.

Из высказываний исследователя

После изучения большого числа различных алгоритмов, описанных в гл. 2, читателю может показаться, что для решения рассмотренных выше задач нет более эффективных методов. Однако в области системного анализа и оптимизации дело обстоит таким образом, что в результате разработки более эффективных алгоритмов могут возникнуть усовершенствованные подходы к решению задач. В теории сетевого анализа значительное место занимает алгоритм дефекта. Он является наиболее общим алгоритмом на детерминированных сетях с ограниченной пропускной способностью и имеет широкую область применения. Поэтому несмотря на то что иногда он уступает по эффективности некоторым специальным алгоритмам, ему уделяют особое внимание.

Описание алгоритма дефекта разбито на три части. В первой части дано теоретическое обоснование алгоритма и описана его работа на одном простом примере. Во второй части показана многогранность области его применения и описаны процедуры построения сети для большого числа классических сетевых задач. В третьей части формулируются и решаются некоторые практические задачи.

ЧАСТЬ I. ОПТИМИЗАЦИЯ ПОТОКА, ОСНОВАННАЯ НА ПРИМЕНЕНИИ АЛГОРИТМА ДЕФЕКТА. ОПИСАНИЕ ТЕОРИИ

Основная цель первой части состоит в описании теоретической основы алгоритма дефекта, который был разработан для оптимизации на сетях с ограниченной пропускной способностью. Алгоритм дефекта основан на теории двойственности линейного программирования и условиях дополняющей нежесткости. Фор-

мальная схема работы алгоритма описана в общих чертах, однако помимо нее даются логическое обоснование метода и описание ряда решающих правил поиска решения задачи. Затем весь алгоритм описывается в виде пяти основных шагов, выполнение которых осуществляется с помощью табличных решающих правил. В заключение, для того чтобы проиллюстрировать применение алгоритма дефекта при решении потоковых задач с сетями с ограниченной пропускной способностью, рассмотрен и решен вручную модельный пример.

3.1. ОСНОВНЫЕ ПОНЯТИЯ

Перед тем как перейти к описанию алгоритма, определим основные понятия, используемые в данном разделе. Некоторые определения были даны выше, однако для наглядности изложения мы вновь остановимся на них.

1. *Узел* — это основной элемент сетевой диаграммы. Обычно он служит для обозначения физического объекта, являющегося начальным или конечным пунктом (например, предприятие, магазин, источник рабочей силы и т. д.). Узел, который порождает

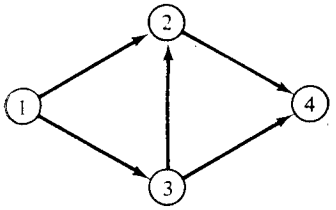


Рис. 3.1. Ориентированная сеть.

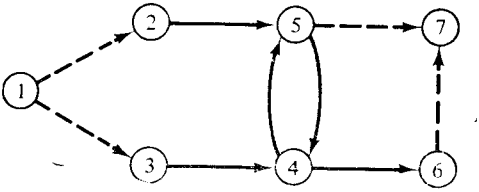


Рис. 3.2. Главный источник и главный сток.

поток (например, склад в некоторых задачах), называется *источником*. Узел, который поглощает поток (например, предприятие-заказчик), называется *стоком*. Множество узлов сети будем обозначать через N.

2. *Дугами*, или ветвями, называются линии, соединяющие различные узлы сети. Дуга является ориентированной, если поток по ней может протекать только в одном заданном направлении. Множество дуг сети будем обозначать через S.

3. *Сеть* — это связное множество дуг и узлов. Обычно она используется для описания физического процесса, в котором единицы потока движутся из источника (или источников) в сток (или стоки). На рис. 3.1 изображена сеть, в которой узел 1 является источником, а узел 4 — стоком. Сеть может содержать несколько источников и стоков. На рис. 3.2 изображена сеть, в которой узлы 2 и 3 являются источниками, а узлы 5 и

6 — стоками. К этой сети с помощью пунктирных дуг добавлены узлы 1 и 7, называемые *главным источником* и *главным стоком* соответственно.

4. Если поток по дуге может принимать только определенные значения, например если заданы верхняя и (или) нижняя границы потока, то говорят, что дуга имеет *ограниченную пропускную способность*.

5. При описании алгоритма удобно воспользоваться понятиями прямой и обратной дуги. Если направление движения по

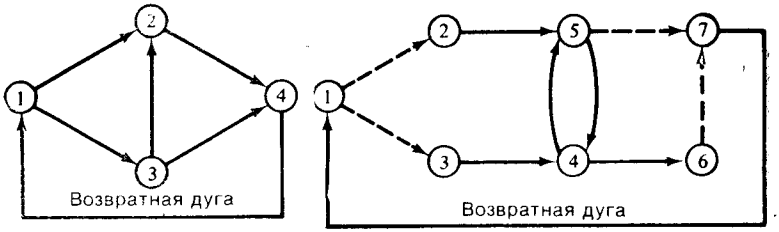


Рис. 3.3. Замкнутые сети.

дуге совпадает с ее ориентацией, то дуга называется *прямой*. Если направление движения по дуге противоположно ее ориентации, то дуга называется *обратной*.

6. Сеть, содержащая дуги с ограниченной пропускной способностью, называется *сетью с ограниченной пропускной способностью*.

7. *Циркуляцией* называется поток по дугам сети, для которого в каждом узле выполняется условие сохранения, т. е. суммарный поток, входящий в узел, равен суммарному потоку, выходящему из узла. Для работы алгоритма дефекта требуется существование циркуляции; поэтому исходная сеть, как правило, соответствующим образом модифицируется. Так, для сетей, изображенных на рис. 3.1 и 3.2, необходимо ввести дополнительную дугу, чтобы соединить сток с источником. Эта дуга называется *возвратной* (рис. 3.3). Способы построения возвратной дуги зависят от вида сети. Подробно о них будет рассказано во второй части настоящей главы.

8. Для сети с ограниченной пропускной способностью всегда заданы верхние и нижние границы (которые могут быть равными нулю или бесконечности) потоков по всем дугам. Величина потока по каждой дуге должна быть заключена между верхним и нижним пределами, и это ограничение, как и все остальные, не должно нарушаться. В настоящей главе будут использованы следующие обозначения: f_{ij} — поток по дуге (i, j), L_{ij} — нижняя пропускная способность дуги (i, j), U_{ij} — верхняя пропускная

способность дуги (i, j), c_{ij} — стоимость прохождения единицы потока из узла i в узел j.

9. Алгоритм дефекта является итеративной процедурой, выполняющей для заданной сети с ограниченной пропускной способностью поиск циркуляции, минимизирующей суммарную стоимость потоков по всем дугам.

3.2. СВЕДЕНИЕ ИСХОДНОЙ ЗАДАЧИ К ЗАДАЧЕ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Рассмотрим ориентированную дугу, соединяющую узел i с узлом j. Пусть f_{ij} — величина потока по этой дуге. Будем предполагать, что:

1. $f_{ij}=0$, если поток по дуге отсутствует;
2. $f_{ij}>0$, если поток протекает из узла i в узел j.

С помощью введенных обозначений на рис. 3.4 изображена часть некоторой сети.

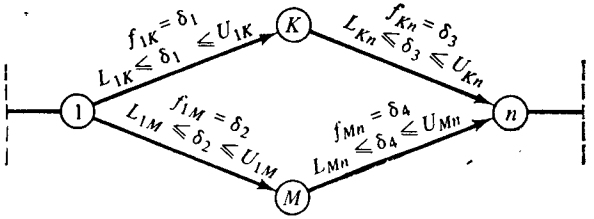


Рис. 3.4. Обозначения для сетей с ограниченной пропускной способностью.

Рассматриваемая потоковая задача может быть сформулирована в виде специальной задачи линейного программирования. Поскольку стоимость прохождения единицы потока по дуге (i, j) равна c_{ij}, то задача минимизации суммарной стоимости формулируется следующим образом:

$$\text{минимизировать} \sum_{(i,\,j)\in S} c_{ij}f_{ij} \tag{3.1}$$

при ограничениях на пропускные способности дуг

$$\begin{aligned} f_{ij} &\leqslant U_{ij}, \quad (i,j)\in S,\\ f_{ij} &\geqslant L_{ij}, \quad (i,j)\in S. \end{aligned} \tag{3.2}$$

Для того чтобы количество продукта, поступающего в узел, равнялось количеству продукта, выходящего из этого узла, требуется выполнение условия сохранения потока, т. е.

$$\sum_{j\in N} f_{ji} - \sum_{j\in N} f_{ij} = 0 \quad \text{для всех } i\in N,\ i\neq j. \tag{3.3}$$

Кроме того, требуют, чтобы потоки были неотрицательными:

$$f_{ij} \geqslant 0 \quad \text{для всех дуг } (i, j).$$

Задача нахождения циркуляции минимальной стоимости представляется в виде специальной задачи линейного программирования (3.1)—(3.3). Именно на основе этой основной формулировки будет описан алгоритм дефекта. При этом используются условия оптимальности, которые следуют из теории двойственности линейного программирования. Перепишем (3.1)—(3.3) в более удобной форме:

$$\text{максимизировать} \sum_{(i,\, j)\, \in S} -c_{ij} f_{ij}$$

при условии, что

$$\sum_{j \in N} f_{ij} - \sum_{j \in N} f_{ji} = 0 \quad \text{для всех } i \in \mathbf{N} \text{ (условие сохранения потока),}$$

$$f_{ij} \leqslant U_{ij} \quad \text{(ограничения на потоки сверху),}$$

$$-f_{ij} \leqslant -L_{ij} \quad \text{(ограничения на потоки снизу),}$$

$$f_{ij} \geqslant 0 \quad \text{(условие неотрицательности потока).}$$

В данной формулировке для удобства последующего описания алгоритма целевая функция умножена на —1. При этом задача минимизации трансформировалась в задачу максимизации, которую будем называть *прямой задачей*. Согласно известному в линейном программировании результату, для любой прямой задачи существует соответствующая ей *двойственная задача*. В рассматриваемом случае двойственная задача формулируется следующим образом:

$$\text{минимизировать} \sum_{(i,\, j)\, \in S} U_{ij}\alpha_{ij} - L_{ij}\delta_{ij}$$

при условии, что

$$\pi_i - \pi_j + \alpha_{ij} - \delta_{ij} \geqslant -c_{ij} \quad \text{для всех } (i, j) \in \mathbf{S},$$

$$\pi_i \quad \text{не имеют ограничений по знаку для всех } i \in \mathbf{N},$$

$$\alpha_{ij} \geqslant 0 \quad \text{для всех } (i, j) \in \mathbf{S},$$

$$\delta_{ij} \geqslant 0 \quad \text{для всех } (i, j) \in \mathbf{S}.$$

В данной формулировке переменные π соответствуют ограничениям, описывающим условие сохранения потока для прямой задачи, и могут принимать произвольные значения, поскольку эти ограничения имеют вид равенств. Переменные α в двойственной задаче соответствуют ограничениям сверху на потоки по

дугам в прямой задаче, а переменные δ — ограничениям снизу. Каждой переменной f_{ij} в прямой задаче соответствует некоторое ограничение в двойственной задаче.

В качестве примера предположим, что из источника 1 в сток 4 по сети, изображенной на рис. 3.5, требуется доставить три единицы продукта. Каждой дуге (i, j) сети приписана трой-

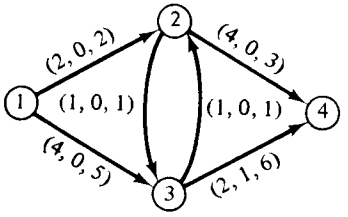


Рис. 3.5. Открытая сеть с ограниченной пропускной способностью.

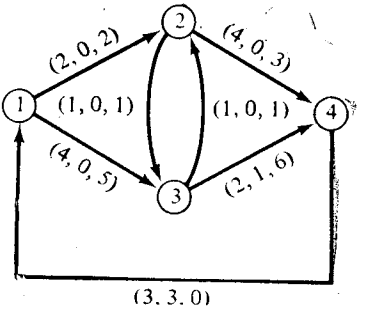


Рис. 3.6. Замкнутая сеть с ограниченной пропускной способностью.

ка (U_{ij}, L_{ij}, c_{ij}). Для данной задачи необходимо «замкнуть» исходную сеть, добавив к ней возвратную дугу (4, 1). Величина потока, который требуется доставить из источника 1 в сток 4, будет равна величине потока по дуге (4, 1) и поэтому определить ее можно, полагая $L_{41}=U_{41}=3$. Стоимость единицы потока по дуге (4, 1) должна быть нулевой, т. е. $c_{41}=0$. Полная сеть циркуляции изображена на рис. 3.6. Прямая и двойственная задачи для данного примера сформулированы в табл. 3.1 и 3.2.

Таблица 3.1. Прямая задача

f_{12}	$+f_{13}$					$-f_{41}$	$=$	0		
$-f_{12}$		$+f_{23}$	$+f_{24}$	$-f_{32}$			$=$	0		Узлы
	$-f_{13}$	$-f_{23}$		$+f_{32}$	$+f_{34}$		$=$	0		
			$-f_{24}$		$-f_{34}$	$+f_{41}$	$=$	0		
f_{12}							$\leqslant$	2		
	f_{13}						$\leqslant$	4		
		f_{23}					$\leqslant$	1		
			f_{24}				$\leqslant$	4		Верхние
				f_{32}			$\leqslant$	1		границы
					f_{34}		$\leqslant$	2		
						f_{41}	$\leqslant$	3		
					$-f_{34}$		$\leqslant$	-1		Нижние
						$-f_{41}$	$\leqslant$	-3		границы
Максимизировать	$-2f_{12}-5f_{13}-f_{23}-3f_{24}-f_{32}-6f_{34}$,								$f_{ij}\geqslant 0$	

Таблица 3.2. Двойственная задача

π_1	$-\pi_2$			$+\alpha_{12}$									$\geq$	-2
π_1		$-\pi_3$			$+\alpha_{13}$								$\geq$	-5
	π_2	$-\pi_3$				$+\alpha_{23}$							$\geq$	-1
	π_2		$-\pi_4$				$+\alpha_{24}$						$\geq$	-3
	$-\pi_2$	$+\pi_3$						$+\alpha_{32}$					$\geq$	-1
		π_3	$-\pi_4$						$+\alpha_{34}$		$-\delta_{34}$		$\geq$	-6
$-\pi_1$			$+\pi_4$							$+\alpha_{41}$		$-\delta_{41}$	$\geq$	0

Минимизировать $2\alpha_{12} + 4\alpha_{13} + 1\alpha_{23} + 4\alpha_{24} + 1\alpha_{32} + 2\alpha_{34} + 3\alpha_{41} - \delta_{34} - 3\delta_{41}$

$\pi_1, \pi_2, \pi_3, \pi_4$ не имеют ограничений по знаку

$\alpha_{ij} \geq 0 \qquad (i, j) \in \mathbf{S}$

$\delta_{ij} \geq 0 \qquad (i, j) \in \mathbf{S}$

Обратимся к рассмотрению прямой и двойственной задач с тем, чтобы разработать метод, который позволял бы:

1. Всегда находить оптимальные решения (если они существуют).

2. Вычислять оптимальные решения более эффективно, чем это позволяют сделать общие алгоритмы решения задачи линейного программирования.

В следующем разделе с помощью теории двойственности в линейном программировании будут получены условия оптимальности.

3.3. ОСНОВНЫЕ ТЕОРЕМЫ

Решения прямой и двойственной задач являются оптимальными в том и только в том случае, если:

1. Оба решения допустимы.

2. Для любой положительной двойственной *переменной* соответствующее ограничение в прямой задаче является жестким[1]).

3. Для любого *ограничения* в двойственной задаче, не являющегося жестким, значение соответствующей *переменной* в прямой задаче равно нулю.

Последние два условия называются *условиями дополняющей нежесткости* и вместе с условиями допустимости они составляют необходимые и достаточные *условия оптимальности* решения задачи о циркуляции минимальной стоимости [7].

Условия допустимости для прямой задачи

P_1: $\sum_j f_{ij} - \sum_j f_{ji} = 0$ для всех $i \in N$ (сохранение потока),

P_2: $L_{ij} \leqslant f_{ij} \leqslant U_{ij}$ для всех $(i, j) \in S$ (ограничения на пропускную способность).

Условия допустимости для двойственной задачи

D_1: $\pi_i - \pi_j + \alpha_{ij} - \delta_{ij} \geqslant -c_{ij}$ для всех $(i, j) \in S$,

D_2: $\alpha_{ij} \geqslant 0$ для всех $(i, j) \in S$,

D_3: $\delta_{ij} \geqslant 0$ для всех $(i, j) \in S$.

Условия дополняющей нежесткости

C_1: если $\pi_i - \pi_j + \alpha_{ij} - \delta_{ij} > -c_{ij}$, то $f_{ij} = 0$,

---

[1]) Жестким называется такое ограничение, что при заданном решении левая его часть равна правой.

C_2: если $\alpha_{ij} > 0$, то $f_{ij} = U_{ij}$,
C_3: если $\delta_{ij} > 0$, то $f_{ij} = L_{ij}$.

Эквивалентная форма условий оптимальности задается следующими соотношениями [3]:

I. Если $\pi_j - \pi_i > c_{ij}$, то $\alpha_{ij} > 0$ и $f_{ij} = U_{ij}$.

II. Если $\pi_j - \pi_i < c_{ij}$, то $\delta_{ij} > 0$ и $f_{ij} = L_{ij}$.

III. Если $\pi_j - \pi_i = c_{ij}$, то $U_{ij} \geqslant f_{ij} \geqslant L_{ij}$

при условии, что

IV. $\alpha_{ij} = \max[0, \pi_j - \pi_i - c_{ij}]$,

V. $\delta_{ij} = \max[0, -\pi_j + \pi_i + c_{ij}]$,

VI. $\sum_j f_{ij} - \sum_j f_{ji} = 0$ для всех i.

Данные условия позволяют проводить очень эффективный поиск оптимального решения путем последовательных изменений вектора решения, поскольку требуется проверять только условия I, II и III, которые не содержат двойственных переменных α и δ.

Предполагая, что условия IV и V выполнены, и вводя обозначение $\bar{c}_{ij} = c_{ij} + \pi_i - \pi_j$, перепишем условия I, II, III и VI в более удобной форме:

k_1: если $\bar{c}_{ij} < 0$, то $f_{ij} = U_{ij}$;
k_2: если $\bar{c}_{ij} > 0$, то $f_{ij} = L_{ij}$,
k_3: если $\bar{c}_{ij} = 0$, то $L_{ij} \leqslant f_{ij} \leqslant U_{ij}$,
k_4: условие сохранения потока.

Если для узлов i и j и дуги, соединяющей их, выполнено одно из условий оптимальности k_1, k_2 или k_3, то говорят, что дуга является *бездефектной*. Если же одно из условий k_1, k_2 или k_3 нарушено, то дуга называется *дефектной*. Решение является оптимальным, если устранены дефекты всех дуг и выполнено условие сохранения потока (условие k_4). Если же таких потоков по дугам не существует, то допустимого решения задачи не существует.

3.4. АЛГОРИТМ ДЕФЕКТА ДЛЯ РЕШЕНИЯ ЗАДАЧИ О ЦИРКУЛЯЦИИ МИНИМАЛЬНОЙ СТОИМОСТИ

При работе алгоритма дефекта определяются значения π_i и f_{ij}, для которых выполнены условия оптимальности k_1, k_2, k_3 и k_4. Работу алгоритма можно начать, приписывая дугам произвольные потоки, удовлетворяющие условию сохранения, а узлам — произвольные величины π_i. При работе алгоритма каждая дуга может находиться в одном из девяти взаимно исклю-

Таблица 3.3. Возможные состояния дуги

Состояние	$\bar{c}_{ij}$	f_{ij}	Отсутствует ли дефект?
α	$\bar{c} > 0$	$f = L$	Да
β	$\bar{c} = 0$	$L \leq f \leq U$	Да
δ	$\bar{c} < 0$	$f = U$	Да
α_1	$\bar{c} > 0$	$f < L$	Нет
β_1	$\bar{c} = 0$	$f < L$	Нет
δ_1	$\bar{c} < 0$	$f < U$	Нет
α_2	$\bar{c} > 0$	$f > L$	Нет
β_2	$\bar{c} = 0$	$f > U$	Нет
δ_2	$\bar{c} < 0$	$f > U$	Нет

чающих состояний, связанных с условиями оптимальности (табл. 3.3).

Проверяя состояния дуг, можно изменять потоки по ним до тех пор, пока не будут выполнены условия оптимальности. По значению $\bar{c}_{ij}$ можно однозначно определить, является ли дуга дефектной или нет, а также выявить, что нужно делать — увеличивать или уменьшать поток по дуге, для того чтобы она перестала быть дефектной. Однако не все так просто, как кажется, поскольку условия оптимальности включают в себя также условие сохранения потока. Поэтому если поток по некоторой дуге (i, j) увеличивается (или уменьшается), то в инцидентных ей узлах нарушается условие сохранения потока. А для того чтобы оно не нарушалось, необходимо найти *другой* путь из узла j в узел i (или из i в j, если поток уменьшается). Дуга (i, j) и этот путь из j в i вместе образуют цикл. Изменение потока по циклу не влияет на сохранение потока ни для какого узла. Путь из j в i следует выбрать так, чтобы, во-первых, ни одна бездефектная дуга не стала бы дефектной и, во-вторых, ни одна из дефектных дуг не стала бы более дефектной. Таким образом, необходимо иметь процедуру, позволяющую определить, каким образом следует изменить потоки по всем рассматриваемым дугам и какой путь следует выбрать. Описание такой процедуры, называемой *процедурой расстановки пометок*, приводится ниже.

3.5. ПРОЦЕДУРА РАССТАНОВКИ ПОМЕТОК

1. Если для того чтобы дуга (i, j) перестала быть дефектной, поток по ней следует *увеличить*, то она находится в одном из состояний β_1, δ_1 или α_1. Приписать узлу j пометку $[q_j, i^+]$, кото-

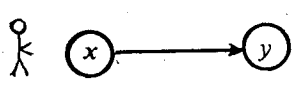
рая означает, что узел j может *получить* q_j дополнительных единиц потока *из* узла i. Если дуга находится в состоянии α_1, то q_j определить равным $\min[q_i, L_{ij}-f_{ij}]$, а если в состоянии β_1 или δ_1[1], то q_j определить равным $\min[q_i, U_{ij}-f_{ij}]$[2].

2. Если поток по дуге (i, j) следует уменьшить, то она находится в одном из состояний β_2, δ_2 или α_2. Приписать узлу i пометку $[q_i, j^-]$, которая означает, что поток, выходящий из узла i и входящий в узел j, может быть уменьшен на величину q_i. Если дуга находится в состоянии α_2 или β_2[3], то q_i определить равным $\min[q_j, f_{ij}-L_{ij}]$, а если — в состоянии δ_2, то q_i определить равным $\min[q_j, f_{ij}-U_{ij}]$.

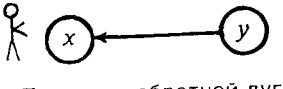
3. Если дуга (i, j) находится в одном из состояний α, β или δ, то она не является дефектной, и поток по ней изменять не надо. Исключение составляет состояние β, для которого поток можно увеличить или уменьшить таким образом, что ни одно из условий не будет нарушено.

Как только обнаруживается, что дуга (i, j) является дефектной, узел i или j помечается по правилу 1 или 2. Если изменение потока по дуге, указанное с помощью пометки, возможно, то дуга может стать бездефектной. Однако, как отмечалось выше, для того чтобы не нарушалось условие сохранения потока, необходимо найти дополнительный путь из i в j (или из j в i). Для определения приращений потоков дополнительного пути следует пометить каждый его внутренний узел.

Рассмотрим произвольную дугу (x, y) дополнительного пути. Эта промежуточная дуга будет находиться в одном из перечисленных выше девяти взаимно исключающих состояний. Если она является дефектной, то поток следует изменить таким образом, чтобы ее дефект не увеличился. Аналогично поток по бездефектной дуге должен измениться так, чтобы она не стала дефектной. Предположим теперь, что мы остановились на некотором помеченном узле x[4] и хотим пройти по дуге из узла x (помеченного) в узел y (непомеченный).



Просмотр прямой дуги



Просмотр обратной дуги

[1] Поскольку $\bar{c}=0$, то, для того чтобы дуга перестала быть дефектной, поток по ней можно увеличить как до нижней, так и до верхней границы.

[2] В результате применения данного правила дуга (i, j) перестает быть дефектной, а дефекты остальных дуг последовательно уменьшаются.

[3] Справедливо замечание, аналогичное замечанию 1.

[4] Поскольку этот процесс всегда начинается в заданном узле, то первоначально один узел всегда является помеченным.

Из помеченного узла мы пытаемся пройти по всем прямым и обратным дугам, инцидентным ему. При некоторых состояниях дуги узел на другом ее конце может быть помечен. Аналогично из одного узла можно пометить несколько различных узлов, хотя для продолжения процедуры достаточно пометить только один узел. После завершения процесса расстановки пометок из данного узла этот узел помечается как *просмотренный.* Затем движемся к помеченному, но *непросмотренному* узлу. Хотя вы помечаете *непросмотренный* узел на инцидентных дугах, вам

Таблица 3.4. Процедура расстановки пометок для прямой дуги

$(x, y) \in \mathbf{S}$ π_x c_{xy} π_y

f_{xy}

$\bar{c}_{xy} = c_{xy} + \pi_x - \pi_y$

Предположим, что узлу x приписана пометка $[q_x, Z^+]$. Можно ли пометить узел y? Если *увеличение потока* f_{xy} приведет к увеличению дефекта дуги (x, y), то y не может быть помечен из x.

Состояние дуги (x, y)	$\bar{c}_{xy}$	f_{xy}	Отсутствует ли дефект?	Может ли y быть помечен?	Почему?
α	$\bar{c} > 0$	$f = L$	Да	Нет	В результате увеличения потока дуга станет дефектной
β	$\bar{c} = 0$	$f < U$	Да	Да	Поток может быть увеличен до U
		$f = U$	Да	Нет	Поток не может быть увеличен
δ	$\bar{c} < 0$	$f = U$	Да	Нет	Поток не может быть увеличен
α_1	$\bar{c} > 0$	$f < L$	Нет	Да	Поток может быть увеличен до L
β_1	$\bar{c} = 0$	$f < L$	Нет	Да	Поток может быть увеличен до U
δ_1	$\bar{c} < 0$	$f < U$	Нет	Да	Поток может быть увеличен до U
α_2	$\bar{c} > 0$	$f > L$	Нет	Нет	Увеличение потока приведет к увеличению дефекта дуги
β_2	$\bar{c} = 0$	$f > U$	Нет	Нет	Увеличение потока приведет к увеличению дефекта дуги
δ_2	$\bar{c} < 0$	$f > U$	Нет	Нет	Увеличение потока приведет к увеличению дефекта дуги

Резюме: приписать узлу y пометку $[q_y, x^+]$, если $\bar{c}_{xy} > 0$ и $f_{xy} < L_{xy}$;
$q_y = \min [q_x, L_{xy} - f_{xy}]$
или если $\bar{c}_{xy} \leq 0$ и $f_{xy} < U_{xy}$;
$q_y = \min [q_x, U_{xy} - f_{xy}]$

следует спросить: «Как изменить поток по этой дуге — уменьшить или увеличить?» После ответа на этот вопрос можно указать приращение потока по этой дуге. Затем вы можете продолжить движение из просмотренного узла в помеченный, но непросмотренный узел. Величина приращения потока по дуге определяется по состоянию этой дуги посредством решающих правил, описанных в табл. 3.4 и 3.5.

Таким образом, должна быть решена следующая задача. Выбирается дефектная дуга (i, j), поток по которой следует из-

Таблица 3.5. Процедура расстановки пометок для обратной дуги

$(y,x) \in S$ π_x c_{yx} π_y

f_{yx}

$\bar{c}_{yx} = c_{yx} + \pi_y - \pi_x$

Предположим, что узлу x приписана пометка $[q_x, Z^+]$. Можно ли пометить узел y? Если *уменьшение потока* f_{yx} приведет к увеличению дефекта дуги (y,x), то y не может быть помечен из x

Состояние дуги (y, x)	$\bar{c}_{yx}$	f_{yx}	Отсутствует ли дефект?	Может ли y быть помечен?	Почему?
α	$\bar{c} > 0$	$f = L$	Да	Нет	В результате уменьшения потока дуга станет дефектной
β	$\bar{c} = 0$	$L \leq f \leq U$	Да	Да	Поток может быть уменьшен на величину $f - L$
		$f = L$	Да	Нет	Поток не может быть уменьшен
δ	$\bar{c} < 0$	$f = U$	Да	Нет	Поток не может быть уменьшен
α_1	$\bar{c} > 0$	$f < L$	Нет	Нет	Уменьшение потока приведет к увеличению дефекта дуги
β_1	$\bar{c} = 0$	$f < L$	Нет	Нет	Уменьшение потока приведет к увеличению дефекта дуги
δ_1	$\bar{c} < 0$	$f < U$	Нет	Нет	Уменьшение потока приведет к увеличению дефекта дуги
α_2	$\bar{c} > 0$	$f > L$	Нет	Да	Поток может быть уменьшен на величину $f - L$
β_2	$\bar{c} = 0$	$f = U$	Нет	Да	Поток может быть уменьшен на величину $f - L$
δ_2	$\bar{c} < 0$	$f > U$	Нет	Да	Поток может быть уменьшен на величину $f - U$

Резюме:
приписать узлу y пометку $[q_y, x^-]$, если $\bar{c}_{yx} \geqslant 0$ и $f_{yx} > L_{yx}$; $q_y = \min [q_x, f_{yx} - L_{yx}]$
или если $\bar{c}_{yx} < 0$ и $f_{yx} > U_{yx}$; $q_y = \min [q_x, f_{yx} - U_{yx}]$

менить так, чтобы она перестала быть дефектной. Для выполнения условия сохранения потока определяется дополнительный путь из узла j в узел i (или из i в j), и затем потоки по дугам этого пути изменяются в соответствии с последней пометкой q_i (или q_j). Допустим, что мы передвигаемся из узла j в узел i, проходя по прямым и обратным дугам через просмотренные узлы. Напомним, что изменение потока по дуге не допускается, если оно приведет к тому, что (1) бездефектная дуга станет дефектной или (2) дефект дуги увеличится. Следовательно, может возникнуть такая ситуация, что мы достигнем некоторого узла, из которого дальнейшее движение невозможно. Может показаться, что в этом случае нельзя построить путь из j в i и, следовательно, решения исходной потоковой задачи не существует. При этом бесполезно переходить к другой дефектной дуге и начинать процесс заново, поскольку в конечном счете, до того как

может быть найдено оптимальное решение, вышеуказанная дуга должна быть выбрана. Данная ситуация называется *непрорывом.* К счастью, при возникновении непрорыва существует еще один способ поиска оптимального потока. Напомним, что состояние дуги однозначно определяется величиной $\bar{c}_{ij}=c_{ij}+\pi_i-\pi_j$ и поэтому оно может измениться в результате изменения значений π. По определению двойственной задачи, которое было дано выше, каждому узлу ставится в соответствие некоторая переменная π. Поэтому для потоковой задачи с n узлами существует ровно n двойственных переменных π. При возникновении непрорыва напрашивается следующий вопрос: значения каких переменных π следует изменить, чтобы построить путь из узла j в узел i (или из i в j, если поток уменьшается)? Напомним, что мы начали движение из узла j и проходили по прямым и обратным дугам через просмотренные узлы, пытаясь достичь узла i. По мере нашего продвижения узлы, через которые мы проходим, помечаются по описанным выше правилам[1]. Следовательно, при возникновении непрорыва существуют два непересекающихся множества узлов: 1) просмотренные помеченные узлы и 2) непомеченные узлы. Очевидно, что в ситуации непрорыва нас интересуют только те узлы, с помощью которых мы сможем завершить продвижение из узла j в узел i. Для продолжения процесса мы должны передвигаться из просмотренного помеченного узла (поскольку из него можно по крайней мере продвинуться дальше) в непомеченный узел. Поэтому необходимо рассмотреть только те числа π, которые соответствуют дугам, соединяющим помеченные узлы с непомеченными. *Обозначим множество всех помеченных узлов через* $\mathbf{A}$, *а множество всех непомеченных узлов — через* $\bar{\mathbf{A}}$.

Если мы остановимся на некотором просмотренном помеченном узле x и посмотрим в сторону непомеченного узла y, то увидим либо прямую, либо обратную дугу. В первом случае поток может протекать из $\mathbf{A}$ в $\bar{\mathbf{A}}$, во втором — из $\bar{\mathbf{A}}$ в $\mathbf{A}$.

Случай I. Пусть $\mathbf{B}$ — множество всех дуг, которые исходят из узлов, принадлежащих $\mathbf{A}$, входят в узлы из $\bar{\mathbf{A}}$ и для которых $\bar{c}>0$, а поток не превосходит верхней границы.

Случай II. Пусть $\bar{\mathbf{B}}$ — множество всех дуг, которые исходят из узлов, принадлежащих $\bar{\mathbf{A}}$, входят в узлы из $\mathbf{A}$ и для которых $\bar{c}<0$, а поток не меньше нижней границы. Поскольку значение $\bar{c}$ можно вычислить для любой дуги из множеств $\mathbf{B}$ и $\bar{\mathbf{B}}$, то процесс нужно продолжить следующим образом:

1. Случай I: $\bar{c}>0$.

---

[1] Либо узел i, либо j принадлежит этому множеству помеченных узлов, поскольку один из них был помечен первоначально.

Определить $\zeta_1=\min\limits_{\bar{\mathbf{B}}}[\bar{c}_{xy}]$, если $\mathbf{B}\neq\emptyset$, и $\zeta_1=\infty$ в противном случае.

2. Случай II: $\bar{c}<0$.

Определить $\zeta_2=\min\limits_{\bar{\mathbf{B}}}[-\bar{c}_{yx}]$, если $\bar{\mathbf{B}}\neq\emptyset$, и $\zeta_2=\infty$ в противном случае.

3. Пусть $\zeta=\min[\zeta_1, \zeta_2]$.

4. Для каждого узла $k\in\bar{\mathbf{A}}$ изменить соответствующее узловое число π_k, прибавляя к нему величину ζ.

5. Сохранить все предыдущие пометки.

Затем, возвращаясь к процедуре расстановки пометок, продолжаем выполнение процесса.

Если в результате выполнения описанных выше шагов изменится состояние по крайней мере одной дуги, ведущей из множества всех помеченных узлов в множество всех непомеченных узлов, то процедуру расстановки пометок следует продолжить до тех пор, пока (1) не возникнет прорыв или (2) вновь не возникнет непрорыв. Если же после выполнения описанных выше шагов состояние ни одной дуги, ведущей из $\mathbf{A}$ в $\bar{\mathbf{A}}$, не изменится то допустимого решения не существует. Если имеет место прорыв, то дополнительный путь из j в i (или из i в j) найден. В этом случае остается только пройти по данному пути в обратном направлении и изменить поток по каждой его дуге на величину q_i (или q_j) последней (среди всех пометок, приписанных узлам этого пути) пометки. Затем все пометки стираются, выбирается другая дефектная дуга и вновь выполняется указанная процедура. Оптимальное решение исходной потоковой задачи будет найдено тогда, когда все дуги станут бездефектными.

Если прорыв не возникает, то вновь определяются множества $\mathbf{B}$ и $\bar{\mathbf{B}}$ и вновь изменяются узловые числа согласно описанным выше правилам. Процедура расстановки пометок повторяется до тех пор, пока либо дуга (i, j) не станет бездефектной, либо не возникнет непрорыв при $\zeta\neq\infty$. В случае когда $\zeta=\infty$, оптимального решения задачи не существует и алгоритм завершает работу[1]. Отметим, что если $\bar{c}_{ij}=0$ для всех дуг, то потоки по всем прямым дугам равны U_{ij}, а потоки по всем обратным дугам равны L_{ij}. В этом случае дополнительный путь построить нельзя.

Итак, алгоритм дефекта может быть описан следующим образом. Строится сеть и определяется начальная циркуляция, удовлетворяющая условию сохранения потока. Нулевая цирку-

[1] Необходимые и достаточные условия сходимости здесь не приводятся. Их можно найти в работах [7] и [13].

ляция всегда удовлетворяет этому условию. Затем узлам приписываются произвольные числа π и выполняется процедура расстановки пометок. В случае возникновения прорыва потоки по дугам изменяются, в противном случае определяются новые узловые числа и данная процедура повторяется. В следующем разделе будет дана графическая интерпретация описанных выше процедур.

3.6. ГРАФИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ АЛГОРИТМА ДЕФЕКТА

В настоящем разделе описан графический подход к выполнению корректирующих операций в алгоритме дефекта. Логичность и простота данной процедуры делают этот мнемонический

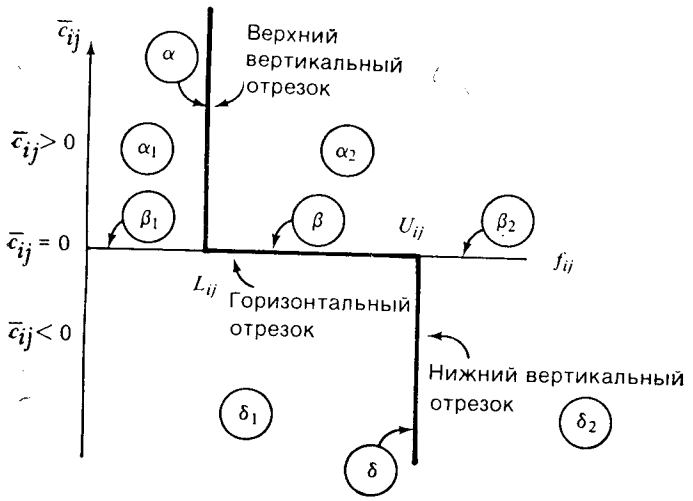


Рис. 3.7. Мнемонический прием для алгоритма дефекта.

прием особенно привлекательным с вычислительной точки зрения.

На любой фазе алгоритма дефекта решения могут быть представлены точками $(f_{ij}, \bar{c}_{ij})$. Состояние каждой дуги (i, j) определяется расположением соответствующей точки относительно границ L_{ij} и U_{ij} (рис. 3.7).

Если дуга не является дефектной, то точка $(f_{ij}, \bar{c}_{ij})$ принадлежит жирной линии, состоящей из трех отрезков (рис. 3.7). Верхний вертикальный отрезок соответствует бездефектному состоянию α, нижний вертикальный отрезок — бездефектному состоянию δ и горизонтальный отрезок — бездефектному состоянию β.

Если дуга является дефектной, то соответствующая точка не принадлежит жирной линии и следует выполнять корректирующие действия. Существуют два типа корректирующих действий:

1. Модификация потоков (f_{ij}).

2. Модификация скорректированных стоимостей ($\bar{c}_{ij}$).

Модификация потоков (процедура расстановки пометок) соответствует горизонтальным перемещениям, указанным на рис. 3.7,

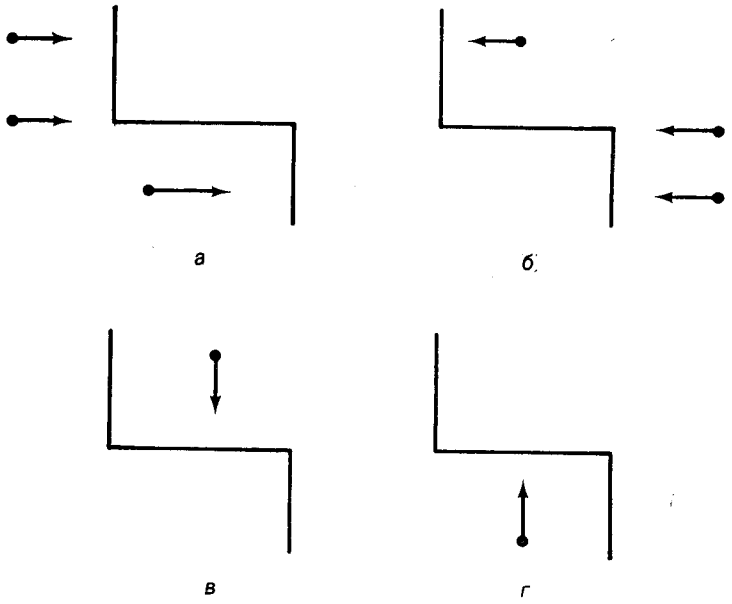


Рис. 3.8. Допустимые направления горизонтальных и вертикальных перемещений.

а модификация скорректированных стоимостей (изменение значений π) — вертикальным перемещениям. Направления допустимых горизонтальных перемещений для каждого из шести состояний дефекта показаны на рис. 3.8, *а, б*. Допустимые вертикальные перемещения показаны на рис. 3.8, *в, г*.

Следующие правила могут быть использованы для определения величин горизонтального и вертикального смещений. Правила горизонтальных перемещений эквивалентны правилам процедуры расстановки пометок. Правила вертикального перемещения эквивалентны правилам изменения значений двойственных переменных π.

3.6.1. ГОРИЗОНТАЛЬНЫЕ ПЕРЕМЕЩЕНИЯ

Рассмотрим произвольную дефектную дугу (i, j). Пусть ей соответствует точка $(f_{ij}, \bar{c}_{ij})$. Эта точка может располагаться слева или справа от каждого из трех отрезков линии порядка.

1. *Горизонтальные перемещения вправо* (увеличение потока). Если точка расположена слева от линии, то соответствующая дуга является дефектной и допустимым горизонтальным перемещением является только перемещение вправо, как это показано на рис. 3.8, *а*. Любая модификация потока, соответствующая горизонтальным перемещениям влево, недопустима, поскольку она ухудшает текущее состояние дуги. В связи с процедурой расстановки пометок это означает, что узел i не может быть помечен из узла j, если (i, j) — обратная дуга. Следующие правила применяются к прямым дугам.

а. Если $\bar{c}_{ij}>0$, то переместить точку так, чтобы она находилась как можно ближе к верхнему вертикальному отрезку линии. Если величина потока достаточна, для того чтобы достичь линии, то дуга перестает быть дефектной. В противном случае дуга продолжает оставаться дефектной, но соответствующая ей точка перемещается в горизонтальном направлении ближе к линии. Приращение потока всегда равно минимуму из величины потока, который может быть получен из узла i, и величины смещения, необходимого для достижения верхнего вертикального отрезка. Пусть q_i — величина потока, который может быть получен из узла i. Приписать узлу j пометку $[q_j, i^+]$, где $q_j=\min[q_i, \bar{L}_{ij}-f_{ij}]$.

б. Если $\bar{c}_{ij}=0$, то переместить точку вправо настолько, насколько это позволяет сделать имеющийся поток, не выходя при этом за верхнюю границу U_{ij}. В этом случае узлу j может быть приписана пометка $[q_j, i^+]$, где $q_j=\min[q_j, U_{ij}-f_{ij}]$.

в. Если $\bar{c}_{ij}<0$, то переместить точку вправо настолько, насколько это позволяет сделать имеющийся поток, не выходя при этом за верхнюю границу U_{ij}. Если нижний вертикальный отрезок достигается, то дуга перестает быть дефектной. В противном случае она продолжает оставаться дефектной, но соответствующая ей точка перемещается в горизонтальном направлении ближе к линии. В этом случае узлу j приписывается пометка $[q_j, i^+]$, где $q_j=\min[q_i, U_{ij}-f_{ij}]$.

2. *Горизонтальные перемещения влево* (уменьшение потока). Если точка расположена справа от линии, то соответствующая дуга является дефектной и допустимым горизонтальным перемещением является только перемещение влево, как это показано на рис. 3.8, *б*. Любая модификация потока, соответствующая горизонтальным перемещениям вправо, недопустима, поскольку она ухудшает текущее состояние дуги. В связи с процедурой

расстановки пометок это вновь означает, что узел j не может быть помечен из узла i, если (i, j) — прямая дуга. Следующие правила применяются к обратным дугам.

а. Если $c_{ij}>0$, то переместить точку так, чтобы она находилась как можно ближе к верхнему вертикальному отрезку. Это эквивалентно тому, что поток послан из j в i в направлении, противоположном направлению дуги (i, j). Если величина потока, который можно получить из узла j, достаточна, для того чтобы в результате горизонтальных перемещений влево точка достигла линии, то дуга перестает быть дефектной. В противном случае она продолжает оставаться дефектной, но соответствующая ей точка перемещается в горизонтальном направлении ближе к линии. Величина, на которую поток уменьшается (т. е. величина потока противоположного направления), всегда равна минимуму из величины потока, который может быть получен из узла j, и расстояния в горизонтальном направлении от точки до линии порядка. Поэтому узел i может быть помечен из узла j как $[q_i, j^-]$, где $q_i=\min[q_j, f_{ij}-L_{ij}]$.

б. Если $c_{ij}=0$, то переместить точку влево настолько, насколько это позволяет сделать имеющийся поток, не переходя при этом за другую сторону вертикального отрезка линии. В этом случае узлу i может быть приписана пометка $[q_i, j^-]$, где $q_i=\min[q_j, f_{ij}-L_{ij}]$.

в. Если $c_{ij}<0$, то переместить точку влево настолько, насколько это позволяет сделать имеющийся поток, не переходя при этом за другую сторону нижнего вертикального отрезка. Если этот отрезок достигается, то дуга перестает быть дефектной. В противном случае она продолжает оставаться дефектной, но соответствующая ей точка перемещается в горизонтальном направлении ближе к линии. В этом случае узлу i приписывается пометка $[q_i, j^-]$, где $q_i=\min[q_j, f_{ij}-U_{ij}]$.

3.6.2. ВЕРТИКАЛЬНЫЕ ПЕРЕМЕЩЕНИЯ

Как видно из рис. 3.8, *в*, *г*, вертикальные перемещения приносят пользу только в том случае, когда они позволяют расположить точку на горизонтальном отрезке линии порядка. Предположим, что узел i помеченный, а узел j непомеченный. Тогда точка с положительной ординатой соответствует ориентированной дуге, направленной из i в j. Если ордината отрицательна, то точка соответствует ориентированной дуге, направленной из j в i. Поэтому достаточно рассмотреть два случая:

1. $c_{ij}>0$ и $f_{ij}<U_{ij}$;
2. $c_{ji}<0$ и $f_{ji}\geqslant L_{ji}$.

В случае 1 в результате вертикального перемещения вниз точка расположится ближе к горизонтальному отрезку линии, как это показано на рис. 3.8, *в*. Напомним, что по определению $\bar{c}_{ij}=c_{ij}+\pi_i-\pi_j$. Следовательно, $0=c_{ij}+\pi_i-(\pi_j+\bar{c}_{ij})$. Это означает, что, для того чтобы точка достигла линии, узловое число π_j следует увеличить на величину $\bar{c}_{ij}$.

В случае 2 в результате вертикального перемещения вверх точка расположится ближе к горизонтальному отрезку линии, как это показано на рис. 3.8, *г*. Поскольку по определению $\bar{c}_{ji}=c_{ji}+\pi_j-\pi_i$, то $0=c_{ji}+(\pi_j-c_{ji})-\pi_i$. Это означает, что, для того чтобы в результате вертикального перемещения точка достигла линии, узловое число π_j следует уменьшить на величину $\bar{c}_{ji}$.

Пусть **В** — множество точек, соответствующих дугам, которые удовлетворяют условиям случая 1. Все эти точки требуется переместить вертикально вниз так, чтобы ордината ни одной из них не стала бы отрицательной. Для этого можно к каждому π_j прибавить минимальную ординату рассматриваемых точек, т. е. к каждому π_j прибавить величину $\zeta_1=\min\limits_{\mathbf{B}}[\bar{c}_{ij}]$.

Аналогично пусть $\bar{\mathbf{B}}$ — множество точек, соответствующих дугам, которые удовлетворяют условиям случая 2. Все эти точки требуется переместить вертикально вниз так, чтобы ордината ни одной из них не стала бы положительной. Для этого можно из каждого узлового числа π_j вычесть максимальную ординату рассматриваемых точек, т. е. к каждому π_j прибавить величину $\zeta_2=-\max[\bar{c}_{ji}]=\min\limits_{\bar{\mathbf{B}}}[-\bar{c}_{ji}]$.

Если узловые числа всех непомеченных узлов должны быть изменены посредством сложения их с одной и той же величиной, то к каждому π_j следует прибавить величину $\zeta=\min[\zeta_1, \zeta_2]$. И наконец, следует отметить, что в результате вертикального перемещения всех точек из **В** и $\bar{\mathbf{B}}$ на величину, равную минимальному расстоянию их от линии, все они расположились ближе к этой линии. Данная процедура гарантирует, что если множества **В** и $\bar{\mathbf{B}}$ не пустые, то по крайней мере одна дефектная дуга станет бездефектной.

3.7. ОПИСАНИЕ ШАГОВ АЛГОРИТМА

Шаг 1. Найти дефектную дугу (i, j). Если ее не существует, то алгоритм завершает работу.

Шаг 2. Определить, как следует изменить поток по этой дуге — увеличить или уменьшить, для того чтобы она перестала быть дефектной. Если поток следует увеличить, то перейти на шаг 3, а если уменьшить, то на шаг 4.

Шаг 3. С помощью процедуры расстановки пометок найти путь из j в i, по которому можно пропустить поток, не увеличивая дефекта ни одной из дуг этого пути. Если такой путь найден, то скорректировать поток по нему и увеличить поток по дуге (i, j). Если дуга (i, j) стала бездефектной, то перейти на шаг 1. Если она по-прежнему является дефектной, то повторить шаг 3. Если такого пути не существует, то перейти на шаг 5.

Шаг 4. Найти путь из i в j, по которому можно пропустить поток, не увеличивая дефекта ни одной из дуг этого пути. Если такой путь найден, то скорректировать поток по нему и уменьшить поток по дуге (i, j). Если дуга (i, j) стала бездефектной, то перейти на шаг 1. Если она по-прежнему является дефектной, то повторить шаг 4. Если такого пути не существует, то перейти на шаг 5.

Шаг 5. Изменить значения переменных π и, сохраняя пометки всех помеченных узлов, повторить шаг 2. Если хотя бы одно узловое число стало равным ∞, то алгоритм завершает работу. Допустимого потока не существует.

3.8. ЧИСЛОВОЙ ПРИМЕР

В настоящем разделе с помощью алгоритма дефекта будет завершено решение задачи, данной на рис. 3.6. Для применения алгоритма необходимо выполнить два шага: (1) выбрать начальные значения двойственных переменных и (2) выбрать начальный поток в сети, удовлетворяющий условию сохранения. При этом выбор допустимого решения не является обязательным, однако необходимо, чтобы эти переменные были явно определены. Для простоты значения переменных π выберем равными нулю: $\pi_1=\pi_2=\pi_3=\pi_4=0$. Потоки, удовлетворяющие условию (2), зададим следующим образом: $f_{12}=0$, $f_{13}=2$, $f_{23}=0$, $f_{32}=2$, $f_{24}=2$, $f_{34}=0$, $f_{41}=2$[1].

3.8.1. ЧИСЛОВОЙ ПРИМЕР ДЛЯ АЛГОРИТМА ДЕФЕКТА

Итерация 1.

1. Выберем дефектную дугу $(4, 1)$[2].

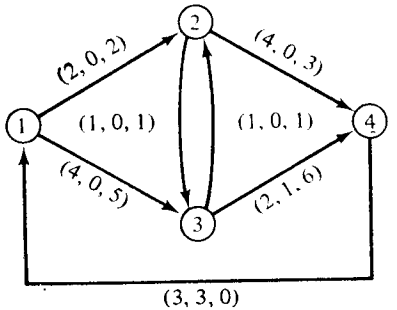
2. Дуга $(4, 1)$ находится в состоянии β_1, поэтому увеличим поток по ней на 1 единицу, делая его равным нижней границе, т. е. 3.

3. С помощью процедуры расстановки пометок находим путь из узла 1 в узел 4.

---

[1] Интуитивно понятно, что если начальные решения выбрать близкими к оптимальным, то алгоритм завершит работу намного быстрее. В рассматриваемом примере решение было выбрано только для иллюстративных целей. Отметим, что оно не является допустимым.

[2] На данном этапе можно выбрать произвольную дефектную дугу.



$$\bar{c}_{xy} = c_{xy} + \pi_x - \pi_y$$

$\bar{c}_{12} = 2$
$\bar{c}_{13} = 5$
$\bar{c}_{23} = 1$
$\bar{c}_{32} = 1$
$\bar{c}_{24} = 3$
$\bar{c}_{41} = 0$
$\bar{c}_{34} = 6$

Начальные условия: $\pi_1 = \pi_2 = \pi_3 = \pi_4 = 0$

$f_{12} = 0 \quad f_{13} = 2$
$f_{23} = 0 \quad f_{32} = 2$
$f_{24} = 2 \quad f_{34} = 0$
$f_{41} = 2$

Дуга (x, y)	$\bar{c}_{xy}$	f_{xy}	Состояние дуги (x, y)		Отсутствует ли дефект?
(1, 2)	0	0	$\bar{c} = 0 \quad f = L$	β	Да
(1, 3)	3	2	$\bar{c} > 0 \quad f > L$	α_2	Нет
(2, 3)	1	0	$\bar{c} > 0 \quad f = L$	α	Да
(2, 4)	3	2	$\bar{c} > 0 \quad f > L$	α_2	Нет
(3, 2)	1	2	$\bar{c} > 0 \quad f > L$	α_2	Нет
(3, 4)	6	0	$\bar{c} > 0 \quad f < L$	α_1	Нет
(4, 1)	2	2	$\bar{c} > 0 \quad f < L$	α_1	Нет

Процедура расстановки пометок.

Узел	Пометка
1	[1, 4+]. Теперь узел 1 является помеченным
2	Не может быть помечен (отсутствие дефекта)
3	Не может быть помечен (увеличение потока приведет к увеличению дефекта дуги)

Возник непрорыв.

Множества узлов: $\mathbf{A} = \{1\}$ $\quad \bar{\mathbf{A}} = \{2, 3, 4\}$

Множества дуг: $\mathbf{B} = \{(1, 2), (1, 3)\}$ $\quad \bar{\mathbf{B}} = \varnothing$

$\zeta_1 = \min\limits_{\mathbf{B}} [2, 5] = 2$

$\zeta_2 = \infty$

$\therefore \zeta = 2$

Новые значения двойственных переменных: $\pi_1 = 0, \ \pi_3 = 2$
$\pi_2 = 2, \ \pi_4 = 2$

Итерация 2.

Дуга (x, y)	$\bar{c}_{xy}$	f_{xy}	Состояние дуги (x, y)		Отсутствует ли дефект?
(1, 2)	2	0	$\bar{c} > 0 \quad f = L$	α	Да
(1, 3)	5	2	$\bar{c} > 0 \quad f > L$	α_2	Нет
(2, 3)	1	0	$\bar{c} > 0 \quad f = L$	α	Да
(2, 4)	3	2	$\bar{c} > 0 \quad f > L$	α_2	Нет
(3, 2)	1	2	$\bar{c} > 0 \quad f > L$	α_2	Нет
(3, 4)	6	0	$\bar{c} > 0 \quad f < L$	α_1	Нет
(4, 1)	0	2	$\bar{c} = 0 \quad f < L$	β_1	Нет

1. Дуга (4, 1) все еще является дефектной.
2. Дуга (4, 1) находится в состоянии α_1, поэтому увеличим поток до нижней границы, равной 3.
3. С помощью процедуры расстановки пометок находим путь из узла 1 в узел 4.

Процедура расстановки пометок.

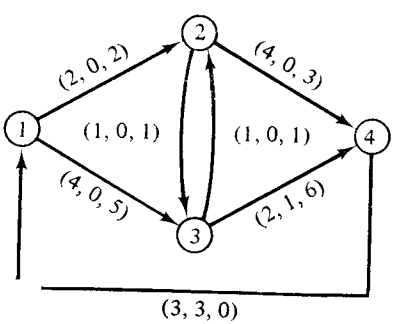
Узел	Пометка
1	[1, 4+]
2	[1, 1+]
3	[1, 2−] Поток можно уменьшить до верхней границы, не увеличивая дефекта дуги
4	[1, 3+]

Возник прорыв.

Изменим потоки по дугам так, как это было описано выше.

Итерация 3.

1. Выберем дефектную дугу, например (1, 3).
2. Дуга (1, 3) находится в состоянии α_2, поэтому уменьшим поток на 2 единицы, делая его равным нижней границе.



$f_{12} = 1$
$f_{13} = 2$
$f_{24} = 2$
$f_{23} = 0$
$f_{32} = 1$
$f_{34} = 1$
$f_{41} = 3$

Дуга (x, y)	$\bar{c}_{xy}$	f_{xy}	Состояние дуги (x, y)		Отсутствует ли дефект?
(1, 2)	0	1	$\bar{c} = 0$ $L < f < U$	β	Да
(1, 3)	3	2	$\bar{c} > 0$ $f > L$	α_2	Нет
(2, 3)	1	0	$\bar{c} > 0$ $f = L$	α	Да
(2, 4)	3	2	$\bar{c} > 0$ $f > L$	α_2	Нет
(3, 2)	1	1	$\bar{c} > 0$ $f = U$	α_2	Нет
(3, 4)	6	1	$\bar{c} > 0$ $f = L$	α	Да
(4, 1)	2	3	$\bar{c} > 0$ $f = L$	α	Да

3. С помощью процедуры расстановки пометок находим путь из узла 1 в узел 3.

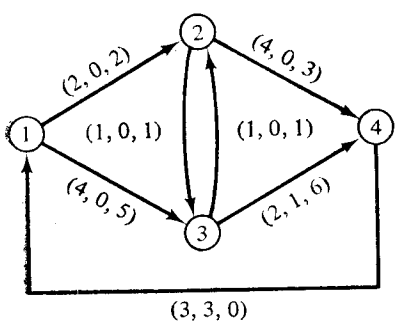
Процедура расстановки пометок.

Узел	Пометка	
1	[2, 3⁻]	
2	[1, 1⁺]	Несмотря на то что дефект отсутствует, поток по дуге (1, 2) может быть увеличен на 1 единицу, и она при этом не станет дефектной
3	[1, 2⁻]	

Возник прорыв.

Изменим потоки по дугам в соответствии с результатами работы процедуры расстановки пометок

Дуга (x, y)	$\bar{c}_{xy}$	f_{xy}	Состояние дуги (x, y)		Отсутствует ли дефект?
(1, 2)	0	2	$\bar{c} = 0$ $f = U$	β	Да
(1, 3)	3	1	$\bar{c} > 0$ $f > L$	α_2	Нет
(2, 3)	1	0	$\bar{c} > 0$ $f = L$	α	Да
(2, 4)	3	2	$\bar{c} > 0$ $f > L$	α_2	Нет
(3, 2)	1	0	$\bar{c} > 0$ $f = L$	α	Да
(3, 4)	6	1	$\bar{c} > 0$ $f = L$	α	Да
(4, 1)	2	3	$\bar{c} > 0$ $f = L$	α	Да



$f_{12} = 2$
$f_{13} = 1$
$f_{24} = 2$
$f_{23} = 0$
$f_{32} = 0$
$f_{34} = 1$
$f_{41} = 3$

Две дуги все еще являются дефектными.

Итерация 4.

1. Выберем дефектную дугу, например (1, 3).
2. Дуга (1, 3) находится в состоянии α_2, поэтому уменьшим поток на 1 единицу, делая его равным нижней границе.
3. С помощью процедуры расстановки пометок находим путь из узла 1 в узел 3.

Процедура расстановки пометок.

Узел	Пометка
1	[1, 3−]
2	Не может быть помечен
4	Не может быть помечен

Возник непрорыв.

Множества узлов: $\mathbf{A}=\{1\}$, $\overline{\mathbf{A}}=\{2, 3, 4\}$

Множества дуг: $\mathbf{B}=\{(1, 3)\}$, $\overline{\mathbf{B}}=\varnothing$

$\zeta_1=\min[3]=3$

$\zeta_2=\infty$

$\therefore \zeta=3$

Новые значения двойственных переменных: $\pi_1=0,\ \pi_2=5$, $\pi_3=5,\ \pi_4=5$

Итерация 5.

Дуга (x, y)	$\bar{c}_{xy}$	f_{xy}	Состояние дуги (x, y)			Отсутствует ли дефект?
(1, 2)	−3	2	$\bar{c}<0$	$f=U$	δ	Да
(1, 3)	0	1	$\bar{c}=0$	$U>f>L$	β	Да
(2, 3)	1	0	$\bar{c}>0$	$f=L$	α	Да
(2, 4)	3	2	$\bar{c}>0$	$f>L$	α_2	Нет
(3, 2)	1	0	$\bar{c}>0$	$f=L$	α	Да
(3, 4)	6	1	$\bar{c}>0$	$f=L$	α	Да
(4, 1)	5	3	$\bar{c}>0$	$f=L$	α	Да

На данном этапе только одна дуга является дефектной. Читателю предлагается проверить, что аугментальный путь потока вновь не может быть построен. Выполняя при данных условиях две последовательные итерации, получаем новые значения двойственных переменных: $\pi_1=1$, $\pi_2=5$, $\pi_3=6$ и $\pi_4=8$. Теперь выполнены следующие условия:

Дуга (x, y)	$\tilde{c}_{xy}$	f_{xy}	Состояние дуги (x, y)			Отсутствует ли дефект?
(1, 2)	−2	2	$\tilde{c} < 0$	$f = U$	δ	Да
(1, 3)	0	1	$\tilde{c} = 0$	$U > f > L$	β	Да
(2, 3)	0	0	$\tilde{c} = 0$	$f = L$	β	Да
(2, 4)	0	2	$\tilde{c} = 0$	$U > f > L$	β	Да
(3, 2)	2	0	$\tilde{c} > 0$	$f = L$	α	Да
(3, 4)	4	1	$\tilde{c} > 0$	$f = L$	α	Да
(4, 1)	7	3	$\tilde{c} > 0$	$f = L$	α	Да

Поскольку все дуги сети стали бездефектными, то условия оптимальности выполнены.

Оптимальная (имеющая минимальную стоимость) циркуляция следующая: $f^*_{41}=3$, $f^*_{12}=2$, $f^*_{32}=0$, $f^*_{13}=1$, $f^*_{34}=1$, $f^*_{23}=0$, $f^*_{24}=2$. Минимальная стоимость равна 21.

3.9. ЗАКЛЮЧЕНИЕ

Алгоритм дефекта был описан на эвристической основе с использованием хорошо известной теории двойственности в линейном программировании. Были описаны все шаги поиска оптимального решения и с целью показать их назначение было дано логическое обоснование алгоритма. Для описания всевозможных состояний, возникающих при работе алгоритма, приводились соответствующие таблицы. Кроме того, было дано пошаговое описание процедур, используемых при решении задачи. Рассмотрен числовой пример, решение которого было получено с помощью алгоритма дефекта.

Основная цель настоящего раздела заключалась в том, чтобы описать с точки зрения логики и организации вычислений теорию и методологию, лежащие в основе алгоритма. Хотя для сетей больших размерностей процедура поиска решения может стать очень громоздкой, все операции в ней четко определены и могут быть запрограммированы. Описание программы, реализующей алгоритм дефекта, приводится в части III. Несмотря на то что алгоритм дефекта применим к широкому классу задач, процедуры поиска решения и критерий оптимальности для различных задач одни и те же, а изменяется лишь конфигурация сети. Таким образом, алгоритм дефекта обладает двумя важными достоинствами:

1. Он позволяет эффективно решать широкий класс потоковых задач.

2. Для начала работы процедуры оптимизации не требуется допустимого решения. Необходимо только, чтобы начальное решение удовлетворяло условию сохранения потока. (В качестве начального решения всегда можно выбрать нулевой вектор.)

Кроме того, алгоритм дефекта обладает достоинством, присущим всем потоковым алгоритмам. А именно, он допускает наглядную интерпретацию, что не свойственно для процедур линейного программирования при числе измерений, большем двух.

ЧАСТЬ II. ОПТИМИЗАЦИЯ ПОТОКА, ОСНОВАННАЯ НА ПРИМЕНЕНИИ АЛГОРИТМА ДЕФЕКТА. ПОСТРОЕНИЕ МОДЕЛЕЙ

Основная цель настоящего раздела состоит в том, чтобы показать возможности применения алгоритма дефекта при решении обобщенных потоковых задач. Рассматриваются следующие потоковые задачи:

1. Транспортная задача.
2. Задача о назначениях.
3. Задача о максимальном потоке.
4. Задача о дереве кратчайших цепей.
5. Задача о перевозках.

Для каждой из этих задач дана сетевая постановка, к которой применим алгоритм дефекта. Кроме того, с помощью алгоритма дефекта формулируется и решается задача производственного планирования. При этом дается экономическая интерпретация решения двойственной задачи. В части III дается описание программы, применимой для решения задач средней размерности (с числом узлов и числом дуг, не превосходящими 500), и с ее помощью решается один пример.

3.10. РЕШЕНИЕ ЗАДАЧИ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМА ДЕФЕКТА

Для применения алгоритма дефекта необходимо выполнить две процедуры:

1. Сформулировать исходную задачу в виде потоковой задачи с замкнутой сетью, имеющей ограниченную пропускную способность.

2. Задать начальные значения двойственных переменных π_k (узловых чисел) и начальную циркуляцию, удовлетворяющую условию сохранения потока.

Что касается первой процедуры, то рассмотрим сначала конфигурацию сети с одним источником и одним стоком (которые в действительности могут являться главным источником и главным стоком). Данная конфигурация изображена на рис. 3.9.

Для построения системы типа замкнутой петли необходимо ввести дугу, соединяющую узел t с узлом s. Будем называть эту дугу возвратной дугой. Для нее определяются такие же харак-

теристики, как и для всех остальных дуг, а именно ей приписываются тройка пропускная способность-стоимость (U, L, c) и двойственные переменные π_s и π_t[1]. Конфигурация полной сети, образованной после введения возвратной дуги, дана на рис. 3.10.

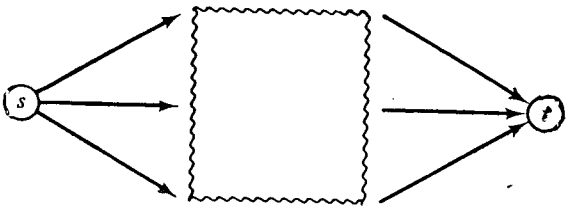


Рис. 3.9. Сеть с одним источником и одним стоком.

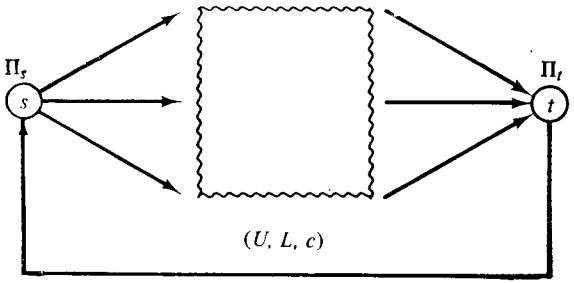


Рис. 3.10. Структура и изображение сети.

Физический или экономический смысл величин U, L, c, π_S и π_t для специальных примеров, решаемых с помощью алгоритма дефекта, будет пояснен ниже.

3.11. ТРАНСПОРТНАЯ ЗАДАЧА

Предположим, что имеется m складов и n магазинов. Из складов в магазины должен быть доставлен некоторый товар. Предложение каждого склада и спрос каждого магазина известны. Задача заключается в нахождении такой схемы перевозки товара из складов в магазины, при которой общие затраты на транспортировку минимальны и спрос каждого магазина удовлетворяется.

Пусть f_{ij} — количество единиц товара, транспортируемого из i-го склада в j-й магазин, a_i — предложение i-го склада, b_j — спрос j-го магазина, c_{ij} — затраты на транспортировку единицы товара из i-го склада в j-й магазин.

[1] В действительности переменные π_s и π_t являются не характеристиками дуг, а единственными характеристиками узлов. Они были введены здесь только для ясности и полноты изложения.

Задача формулируется следующим образом:

$$\text{минимизировать} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} f_{ij}$$

при условии, что

$$\sum_{j=1}^{n} f_{ij} \leqslant a_i, \quad i = 1,2,\ldots,m,$$

$$\sum_{i=1}^{m} f_{ij} \geqslant b_j, \quad j = 1,2,\ldots,n,$$

$$f_{ij} \geqslant 0 \quad \text{для всех } i \text{ и } j.$$

Для того чтобы решить транспортную задачу, используя алгоритм дефекта, нужно выполнить следующие процедуры.

1. Существует ровно m источников и n стоков, или m начальных узлов и n пунктов назначения. Из каждого источника

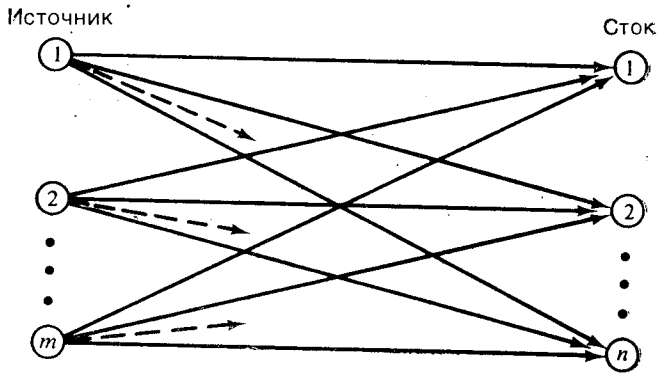


Рис. 3.11. Открытая транспортная сеть.

во все стоки может быть доставлено в общей сложности не более a_i единиц потока. Конфигурация сети для данной задачи изображена на рис. 3.11. Такая сеть называется *двусторонней*.

2. Для каждой дуги в качестве тройки пропускная способность-стоимость (U, L, c) взять тройку $(\infty, 0, c_{ij})$.

3. Ввести главный источник s и главный сток t и завершить построение сети согласно следующим правилам:

а. Для каждого источника i построить дугу, ведущую из главного источника в i. Определить для этой дуги тройку пропускная способность-стоимость $(U, L, c) = (a_i, 0, 0)$.

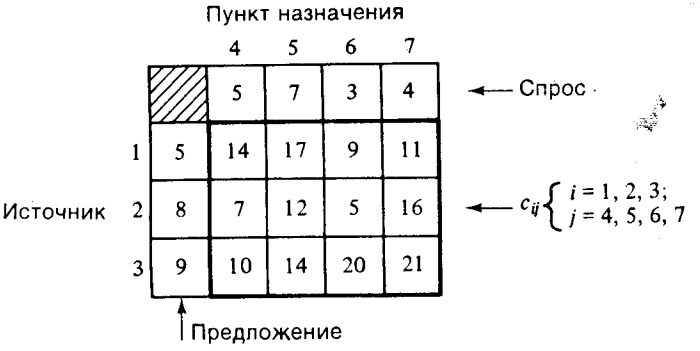
6. Для каждого пункта назначения j построить дугу, ведущую из j в главный сток. Определить для этой дуги тройку $(U, L, c)=(\infty, b_j, 0)$.

4. Построить дугу (t, s) и определить для нее тройку $(U, L, c)=(\sum_{j=1}^{n} b_j, \sum_{j=1}^{n} b_j, 0)$.

5. В качестве начальных значений всех потоков и двойственных переменных взять $f_{ij}=0$ и $\pi_k=0$ $(k=1, 2, \ldots, (m+n+2))$. Отметим, что возвратная дуга (t, s) находится в состоянии β_1, так как $\bar{c}_{ts}=0$ и поток по ней меньше нижней границы. Следовательно, в начале работы алгоритма возвратная дуга всегда является дефектной.

3.11.1. ПРИМЕР ПОСТАНОВКИ ТРАНСПОРТНОЙ ЗАДАЧИ

Предположим, что задана следующая матрица условий:



На рис. 3.12 изображена сеть для поставленной задачи. Значения искомых переменных f_{ij} могут быть найдены с помощью алгоритма дефекта.

3.12. ЗАДАЧА О НАЗНАЧЕНИЯХ

Задача о назначениях возникает в том случае, когда имеется q пунктов назначения, в каждый из которых требуется доставить по 1 единице продукта из источников, число которых также равно q. Из каждого источника в некоторый пункт назначения можно послать только 1 единицу продукта. Стоимость транспортировки единицы продукта из каждого источника в каждый пункт назначения известна. Задача заключается в минимизации общих затрат на транспортировку.

Пусть f_{ij} — число единиц продукта, транспортируемых из источника i в пункт назначения j, s_i — предложение i-го источни-

ка, b_j — спрос j-го пункта назначения, c_{ij} — затраты на транспортировку 1 единицы продукта из i-го источника в j-й пункт назначения.

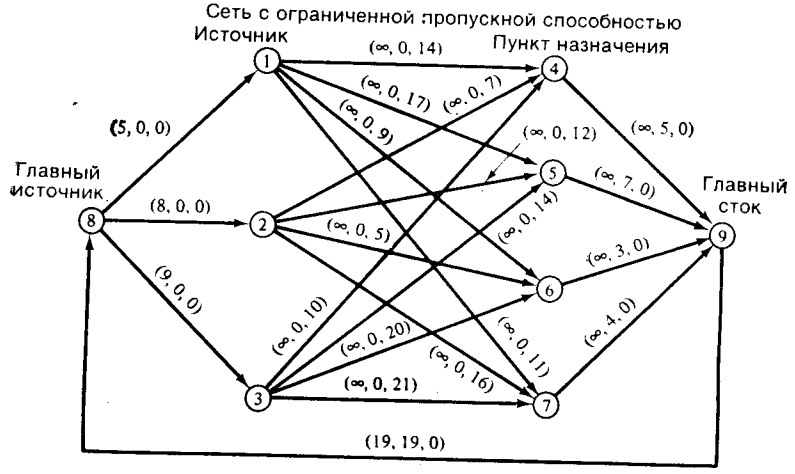


Рис. 3.12. Полная транспортная сеть.

Задача линейного программирования в данном случае выглядит следующим образом:

$$\text{минимизировать} \sum_{i=1}^{q} \sum_{j=1}^{q} f_{ij} c_{ij}$$

при условии, что

$$\sum_{j=1}^{q} f_{ij} \leqslant s_i = 1, \quad i = 1,2,\ldots,q,$$

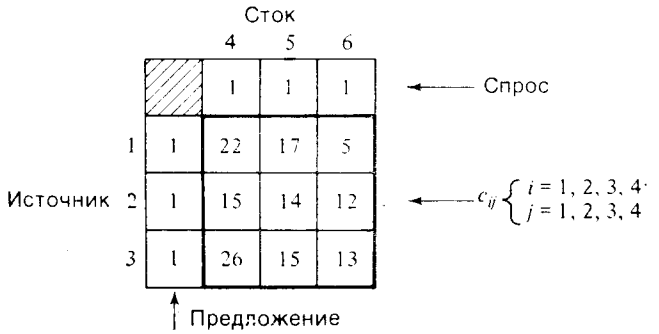
$$\sum_{i=1}^{q} f_{ij} \geqslant b_j = 1, \quad j = 1,2,\ldots,q,$$

$$f_{ij} \geqslant 0 \quad \text{для всех } i, j.$$

Фактически задача о назначениях является специальной транспортной задачей, в которой $m = n = q$ и $a_i = b_j = 1$ для всех источников и всех пунктов назначения. В данной задаче матрица условий всегда является квадратной, а общий спрос равен общему предложению. Следовательно, всегда существует максимальный поток. Правила построения сети такие же, как и для транспортной задачи при данных ограничениях.

3.12.1. ПРИМЕР ПОСТАНОВКИ ЗАДАЧИ О НАЗНАЧЕНИЯХ

Рассмотрим задачу о назначениях со следующей матрицей условий:



На рис. 3.13 изображена сеть для данной задачи. Искомые переменные f_{ij} могут быть найдены с помощью алгоритма дефекта.

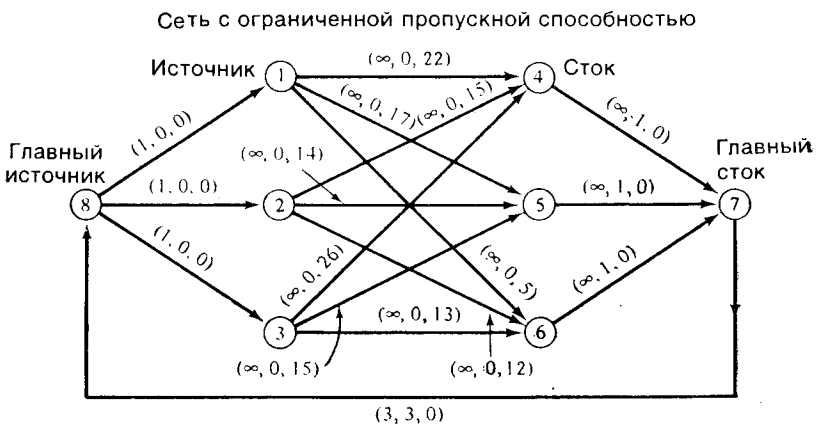


Рис. 3.13. Сеть в задаче о назначениях.

3.13. МАКСИМАЛЬНЫЙ ПОТОК В СЕТЯХ С ОГРАНИЧЕННОЙ ПРОПУСКНОЙ СПОСОБНОСТЬЮ

Предположим, что для сети с ограниченной пропускной способностью требуется определить максимально возможный поток из источника s в сток t. Верхняя и нижняя границы потока по каждой дуге заданы. Для решения этой задачи с помощью алгоритма дефекта необходимо выполнить следующие процедуры:

1. Построить дугу (t, s), для которой $(U, L, c) = (\infty, 0 -\infty)$.

2. Нижние границы для всех оставшихся дуг положить равными нулю, а верхние границы — максимальным пропускным способностям дуг: $L_{ij}=0$ для всех $i\neq t$ и всех $j\neq s$; U_{ij} равна максимальной пропускной способности дуги (i, j) для всех $j\neq s$.

3. Стоимости всех оставшихся дуг положить равными нулю: $c_{ij}=0$ для всех $i\neq t$, $j\neq s$.

4. Потоки по всем дугам и значения всех двойственных переменных положить равными нулю: $f_{ij}=0$ для всех i, j; $\pi_k=0$ для всех k. Отметим, что поскольку потоки по всем дугам нулевые, то каждая дуга, кроме (t, s), находится в состоянии β (является бездефектной). Дуга (t, s) находится в состоянии δ_1 (является дефектной).

5. Выполнять алгоритм обычным образом.

3.14. ЗАДАЧА О КРАТЧАЙШЕЙ ЦЕПИ

Иногда требуется найти кратчайшую цепь из источника s в сток t. При этом «стоимость» дуги соответствует времени или расстоянию. Эта задача может быть решена следующим образом.

Пусть d_{ij} — расстояние от узла i до узла j (или соответствующее время).

1. Построить новую дугу (t, s), для которой $(U, L, c)=(1, 1, 0)$.

2. Для всех остальных дуг положить $(U, L, c)=(1, 0, d_{ij})$.

3. Для всех i, j и k положить $f_{ij}=0$, $\pi_k=0$. Кратчайшая цепь из s в t определяется по окончании работы алгоритма дефекта. Она состоит из всех дуг, поток по которым равен 1.

Возможна также другая формулировка задачи о кратчайшей цепи в виде задачи о потоке минимальной стоимости. Для этого нужно:

1. Ввести дугу (t, s), для которой положить $(U, L, c)=(1, 0, -\infty)$.

2. Для всех остальных дуг положить $(U, L, c)=(1, 0, d_{ij})$.

3. Для всех i, j и k положить $f_{ij}=0$, $\pi_k=0$. В результате работы алгоритма через сеть будет проходить только 1 единица потока. Кратчайшая сеть будет состоять из дуг, поток по которым равен 1.

3.15. ЗАДАЧА О ДЕРЕВЕ КРАТЧАЙШИХ ЦЕПЕЙ

В этой задаче определяются кратчайшие цепи из произвольного узла k во все остальные узлы сети или в некоторое заданное множество узлов. Задача может быть решена с помощью следующих процедур:

1. Для каждого узла, отличного от k, построить дугу, ведущую из этого узла в узел k.

2. Для каждой такой дуги положить $(U, L, c)=(1, 1, 0)$.

3. Для каждой дуги (i, j) сети положить $(U, L, c)=(\infty, 0, d_{ij})$, где d_{ij} — расстояние от узла i до узла j.

4. Положить $f_{ij}=0$ для всех дуг и $\pi_k=0$ для всех узлов.

5. Построить возвратную дугу, для которой $(U, L, c)=(\phi, \phi, 0)$, где ϕ — число дуг, добавленных к узлу k. Данный метод позволяет находить дерево кратчайших цепей. Цепь из узла k в произвольный узел $i\neq k$, принадлежащая этому дереву, короче любой другой цепи из k в i. Дугами этого дерева являются все дуги, по которым поток, полученный в результате работы алгоритма дефекта, положителен.

Данная задача отличается от задачи о кратчайшем остове, в которой минимизируется сумма длин дуг дерева.

Следует отметить, что описанная процедура позволяет находить дерево кратчайших цепей, но, вообще говоря, не позволяет минимизировать общую стоимость.

3.16. ЗАДАЧА О ПЕРЕВОЗКАХ

Рассмотрим систему, в которой имеется m складов и n магазинов. Предложение i-го склада равно a_i $(i=1, 2, ..., m)$, а спрос j-го магазина равен b_j $(j=1, 2, ..., n)$. Из каждого склада товар может быть перевезен в любой магазин либо непосредственно, либо по дополнительным маршрутам, включающим промежуточные склады. Кроме того, товар можно перевозить вначале из одного склада в другой, а затем — в магазин. На первый взгляд может показаться, что данная задача решается непосредственно с помощью алгоритма решения транспортной задачи. Однако в этом случае величины «предложения» и «спроса» каждого магазина были бы равны нулю и, следовательно, никакие перевозки не были бы возможны [6]. Кроме того, возможно, что стоимость транспортировки 1 единицы товара из i-го источника в j-й магазин не равна стоимости транспортировки 1 единицы товара из j-го магазина в i-й источник, что имеет место, например, при перевозках по улицам с односторонним движением.

Для решения первой задачи выберем достаточно большое число θ, которое превосходило бы объем всех возможных перевозок, и прибавим его ко всем величинам первоначальных предложений и спросов. Как было показано в разд. 2.11, наиболее разумно выбрать в качестве θ величину $\min[\sum_{i=1}^{m} a_i, \sum_{=1}^{n} b_j]$. Соответствующие затраты на транспортировку должны определяться из существующих условий экономического характера и постановки задачи.

Рассмотрим задачу о перевозках с двумя источниками снабжения и двумя магазинами. Матрица стоимостей задается следующим образом:

		s_1	s_2	o_1	o_2	
		θ	θ	$3+\theta$	$6+\theta$	Спрос
s_1	$5+\theta$	1	4	6	2	
s_2	$4+\theta$	2	0	8	3	
o_1	θ	6	7	0	3	
o_2	θ	5	2	9	1	
Предложение						

В матрице $\theta=\min[\Sigma a_i, \Sigma b_j]=\min[9, 9]=9$.

Данная задача о перевозках свелась к классической транспортной задаче, которая может быть решена с помощью алгоритма дефекта так, как описывалось выше. Следует отметить, что в рассмотренном примере (а) $m=n$ и (б) $\sum_{i=1}^{m} a_i=\sum_{j=1}^{n} b_j$, однако в общем случае это не так. Если не выполнено условие (а), то это просто означает, что матрица стоимостей не будет квадратной. Если не выполнено условие (б), то для существования решения должно выполняться неравенство $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$. Отметим, что вводить фиктивный пункт назначения нет необходимости. Нужно только приписать возвратной дуге метку $(\sum_j [b_j+\theta], \sum_j [b_j+\theta], 0)$.

3.17. НЕЛИНЕЙНЫЕ СТОИМОСТИ

Некоторые задачи с нелинейными стоимостями дуг могут быть решены с помощью алгоритма дефекта посредством кусочно-линейной аппроксимации. На рис. 3.14 изображена выпуклая нелинейная функция стоимости дуги (x, y) и аппроксимирующая ее кусочно-линейная функция.

Линеаризованной функции стоимости в сети соответствуют три дуги (1), (2), (3) (рис. 3.15), ведущие из x в y. Поскольку исходная функция стоимости выпуклая, то величины наклонов отрезков прямых удовлетворяют соотношению $a_1<a_2<a_3$. Таким образом, при увеличении потока, протекающего из x в y, вначале он приписывается дуге (1). После насыщения дуги (1)

поток приписывается дуге (2). А после насыщения дуги (2) поток приписывается дуге (3). Результирующей кривой стоимости потока является ломаная линия, изображенная на рис. 3.14.

Если функция стоимости потока является вогнутой (рис. 3.16), то исходная сеть не может быть аппроксимирована

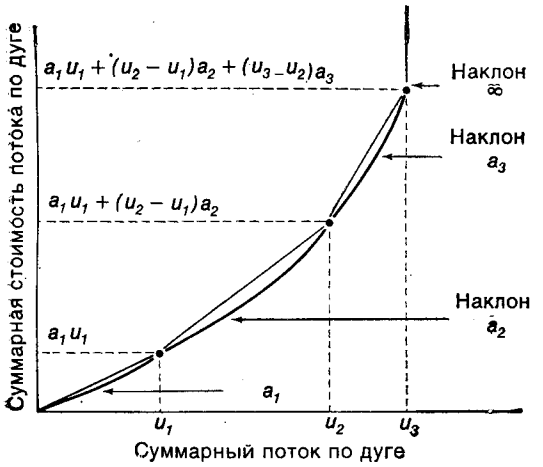


Рис. 3.14. Нелинейные стоимости.

линейной сетью и, следовательно, в этом случае нельзя воспользоваться алгоритмом дефекта. Это является основным ограничением алгоритма дефекта. Для изучения моделей с вогнутыми

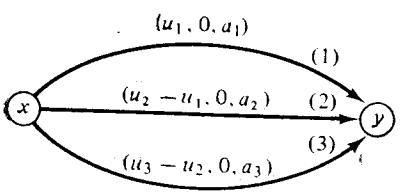


Рис. 3.15. Сетевая интерпретация кусочно-линейной аппроксимации.

Рис. 3.16. Вогнутая нелинейная функция стоимости.

функциями стоимостей потоков читателю предлагается обратиться к работе Йенсена и Барнеса [5].

В качестве последнего примера, иллюстрирующего применение алгоритма дефекта, будет рассмотрена и решена следующая задача производственного планирования [8].

3.18. ЗАДАЧА ПРОИЗВОДСТВЕННОГО ПЛАНИРОВАНИЯ (ФИЛЛИПС И ЙЕНСЕН)

Компания, изготавливающая стулья, владеет четырьмя фабриками, расположенными в различных городах страны. Стоимость изготовления одного стула, не считая стоимости древесины, а также минимальная и максимальная месячные выработки даны в табл. 3.6. Для изготовления одного стула требуется 20 фунтов древесины. Компания получает древесину у двух поставщиков, каждый из которых может продавать ее в любом количестве. Однако по условию контракта фирма должна закупать

Таблица 3.6. Стоимости продукции и уровни производства

Фабрика	Стоимость производства одного стула (долл.)	Максимальная производительность	Минимальная производительность
1	5	500	0
2	7	750	400
3	3	1000	500
4	4	250	250

у каждого поставщика не менее 8 т древесины. Стоимости древесины следующие:

Поставщик 1: 10 цент/фунт,

Поставщик 2: 7,5 цент/фунт.

Затраты (в центах) на транспортировку одного фунта древесины от поставщиков на фабрики задаются следующей таблицей:

Источник поставки древесины	Фабрика 1	2	3	4
1	1	2	4	4
2	4	3	2	2

Стулья продаются главным образом в четырех городах: Нью-Йорке, Чикаго, Сан-Франциско и Остине. Затраты (в долларах на стул) на транспортировку стульев с фабрики в эти города приводятся в следующей таблице:

Фабрика	Город Н-Й	О	С-Ф	Ч
1	1	1	2	0
2	3	6	7	3
3	3	1	5	3
4	8	2	1	4

Таблица 3.7. **Данные о ценах и спросе**

Город	Продажная цена (долл.)	Максимальный спрос	Минимальный спрос
Нью-Йорк	20	2000	500
Остин	15	400	100
Сан-Франциско	20	1500	500
Чикаго	18	1500	500

В табл. 3.7 приводятся максимальный и минимальный спрос, а также цена одного стула в указанных городах.

С помощью алгоритма дефекта мы определим: 1. где следует закупать древесину каждой фабрике; 2. сколько стульев следует изготавливать каждой фабрике; 3. сколько стульев следует продавать в каждом городе; 4. куда следует каждой фабрике отправлять свою продукцию.

На рис. 3.17 изображена сеть для данной задачи, предназначенная для работы алгоритма дефекта. Метки, приписанные дугам сети, выписаны в табл. 3.8. Отметим, что все стоимости задаются в центах на стул, а верхние и нижние границы — в стульях. Оптимальное (имеющее минимальную стоимость) решение данной задачи, полученное с помощью алгоритма дефекта, дается в табл. 3.9.

Теперь на поставленные выше вопросы можно дать ответы в терминах искомых переменных f_{ij}.

1. Где следует закупать древесину каждой фабрике?

а. Фабрика 1 должна закупать древесину для 500 стульев (т. е. 10 000 фунтов) у поставщика 1 и для 0 стульев (т. е. 0 фунтов) у поставщика 2.

б. Фабрика 2 должна закупать древесину для 300 стульев (т. е. 6000 фунтов) у поставщика 1 и для 450 стульев (т. е. 9000 фунтов) у поставщика 2.

в. Фабрика 3 должна закупать древесину для 0 стульев (т. е. 0 фунтов) у поставщика 1 и для 1000 стульев (т. е. 20 000 фунтов) у поставщика 2.

г. Фабрика 4 должна закупать древесину для 0 стульев (т. е. 0 фунтов) у поставщика 1 и для 250 стульев (т. е. 5000 фунтов) у поставщика 2.

2. Сколько стульев следует изготавливать каждой фабрике?

а. Фабрика 1 должна изготавливать 500 стульев.

б. Фабрика 2 должна изготавливать 750 стульев.

в. Фабрика 3 должна изготавливать 1000 стульев.

г. Фабрика 4 должна изготавливать 250 стульев.

3. Сколько стульев следует продавать в каждом городе?

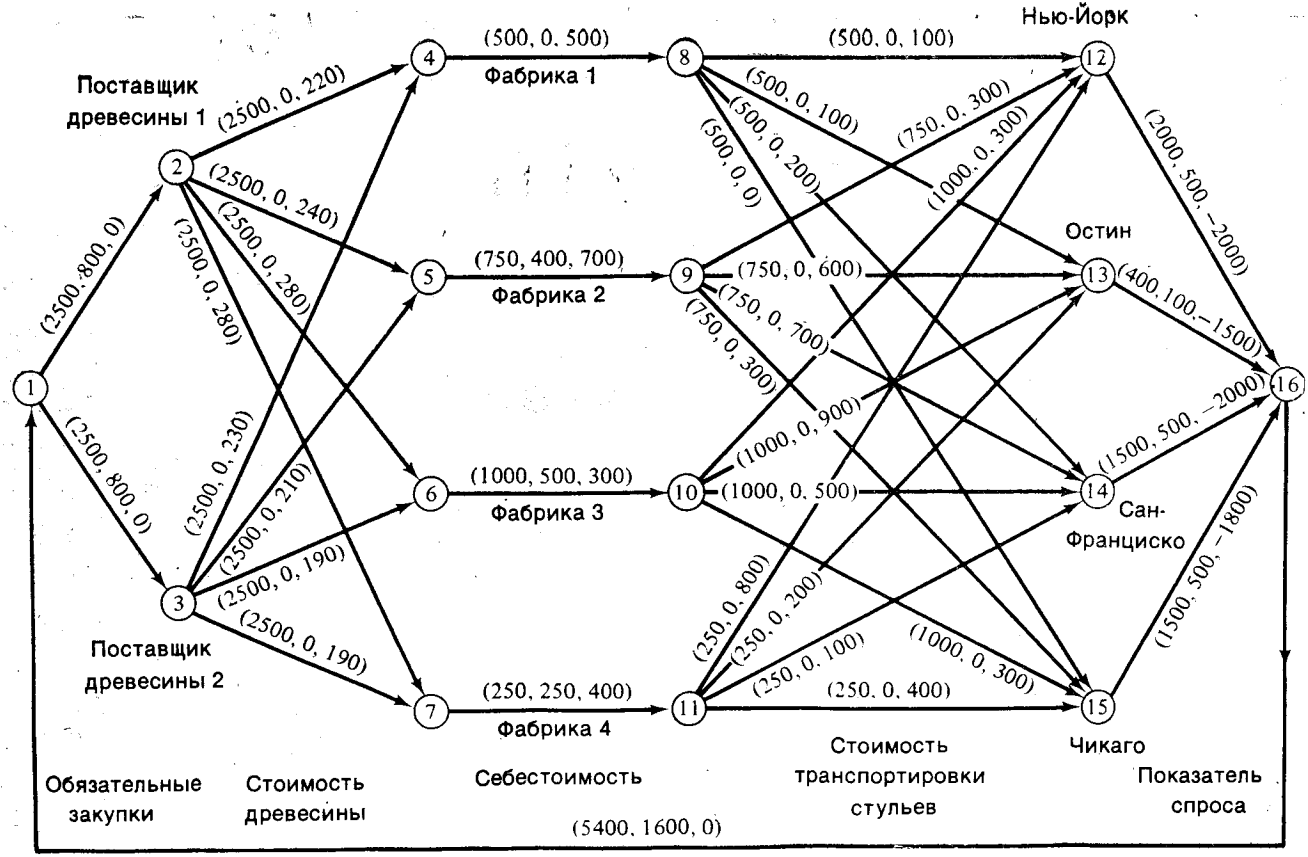


Рис. 3.17. Сетевая модель производства и распределения.

Таблица 3.8. Характеристики дуг сети

Дуга	Из узла	В узел	Верхняя граница	Нижняя граница	Стоимость
1	1	2	2500	800	0
2	1	3	2500	800	0
3	2	4	2500	0	220
4	2	5	2500	0	240
5	2	6	2500	0	280
6	2	7	2500	0	280
7	3	4	2500	0	230
8	3	5	2500	0	210
9	3	6	2500	0	190
10	3	7	2500	0	190
11	4	8	500	0	500
12	5	9	750	400	700
13	6	10	1000	500	300
14	7	11	250	250	400
15	8	12	500	0	100
16	8	13	500	0	100
17	8	14	500	0	200
18	8	15	500	0	0
19	9	12	750	0	300
20	9	13	750	0	600
21	9	14	750	0	700
22	9	15	750	0	300
23	10	12	1000	0	300
24	10	13	1000	0	100
25	10	14	1000	0	500
26	10	15	1000	0	300
27	11	12	250	0	800
28	11	13	250	0	200
29	11	14	250	0	100
30	11	15	250	0	400
31	12	16	2000	500	−2000
32	13	16	400	100	−1500
33	14	16	1500	500	−2000
34	15	16	1500	500	−1800
35	16	1	5400	1600	0

Число узлов равно 16
Число дуг равно 35

В рассматриваемом примере потоки по дугам 1—10 и 35 не ограничены сверху. Для удобства соответствующие верхние границы полагаются равными произвольному достаточно большому числу. Теоретически эти величины равны $+\infty$.

Таблица 3.9. Оптимальное решение

Узел k	π_k	Дуга (i, j)	f_{ij}	Дуга (i, j)	f_{ij}
1	1230	1	800	19	750
2	1200	2	1700	20	0
3	1230	3	500	21	0
4	1420	4	300	22	0
5	1440	5	0	23	650
6	1420	6	0	24	100
7	1420	7	0	25	250
8	3230	8	450	26	0
9	2930	9	1000	27	0
10	2930	10	250	28	0
11	3330	11	500	29	250
12	3230	12	750	30	0
13	3030	13	1000	31	1400
14	3430	14	250	32	100
15	3230	15	0	33	500
16	1230	16	0	34	500
		17	0	35	2500
		18	500		

а. В Нью-Йорке надо продавать 1400 стульев.
б. В Остине надо продавать 100 стульев.
в. В Сан-Франциско надо продавать 500 стульев.
г. В Чикаго надо продавать 500 стульев.

4. Куда следует каждой фабрике отправлять свою продукцию?

а. Фабрика 1 отправляет 500 стульев в Чикаго.
б. Фабрика 2 отправляет 750 стульев в Нью-Йорк.
в. Фабрика 3 отправляет 650 стульев в Нью-Йорк, 100 стульев — в Остин и 250 стульев — в Сан-Франциско.
г. Фабрика 4 отправляет 250 стульев в Сан-Франциско.

Оптимальное решение данной задачи производственного планирования было получено посредством решения эквивалентной ей двойственной задачи и обратного перехода к соответствующим прямым переменным. С теоретической и практической точки зрения интересно дать экономическую интерпретацию двойственным переменным. Напомним, что в алгоритме дефекта используются условия дополняющей нежесткости двойственной задачи линейного программирования и, в частности, двойственные переменные α_{ij} и δ_{ij} выбираются так, что [3, 4]

$$\alpha_{ij} = \max [0,\ \pi_j - \pi_i - c_{ij}],$$

$$\delta_{ij} = \max [0,\ \pi_i - \pi_j + c_{ij}].$$

Используя найденное решение и рис. 3.17, можно построить следующий сегмент сети, содержащий узлы 1, 2, 3 и 5.
Для данной сети имеют место следующие результаты:

дуга (1, 2) $\longrightarrow \pi_2 - \pi_1 < c_{12} \Rightarrow f_{12} = L_{12}$,
дуга (2, 5) $\longrightarrow \pi_5 - \pi_2 = c_{25} \Rightarrow L_{25} \leqslant f_{25} \leqslant U_{25}$,
дуга (1, 3) $\longrightarrow \pi_3 - \pi_1 = c_{13} \Rightarrow L_{13} \leqslant f_{13} \leqslant U_{13}$,
дуга (3, 5) $\longrightarrow \pi_5 - \pi_3 = c_{35} \Rightarrow L_{35} \leqslant f_{35} \leqslant U_{35}$.

Как следует из рис. 3.18, значения двойственных переменных, соответствующих дуге (1, 2), равны $\alpha_{12} = \max[0, 1200 - 1230 -$

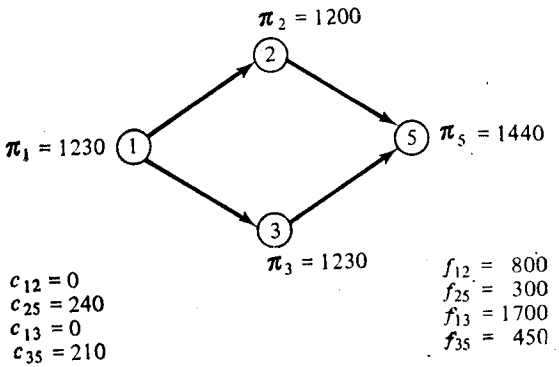


Рис. 3.18. Подсеть.

$- 0] = 0$, $\delta_{12} = \max[0, 1230 - 1200 + 0] = 30$ центов. Поскольку двойственные переменные α соответствуют ограничениям на поток сверху в прямой задаче, а двойственные переменные δ соответствуют ограничениям на поток снизу, то выражение $\delta_{12} = 0{,}30$ долл. означает, что в результате уменьшения нижней границы потока по дуге (1, 2) (и самого потока по этой дуге) на 1 единицу будет сэкономлено 0,30 долл. в расчете на один стул. Это следует из того, что в результате уменьшения потока из узла 1 в узел 2 на 1 единицу будет сэкономлено 0,00 долл. Уменьшение потока из узла 2 в узел 5 на 1 единицу даст экономию 2,40 долл. Для сохранения потока, вытекающего из узла 1 и втекающего в узел 5, следует увеличить на 1 единицу поток из узла 1 в узел 3 и из узла 3 в узел 5. В первом случае затраты на увеличение составят 0,00 долл., во втором — 2,10 долл. Следовательно, чистый выигрыш составит 0,30 долл., как было получено при исследовании двойственной переменной δ_{12}.

В качестве второго примера рассмотрим следующий сегмент сети.

Для данной сети имеют место следующие результаты:

дуга (9, 13) $\longrightarrow \pi_{13} - \pi_9 < c_{9,13} \Rightarrow f_{9,13} = L_{9,13}$,

дуга (13, 16) $\longrightarrow \pi_{16} - \pi_{13} < c_{13,16} \Rightarrow f_{13,16} = L_{13,16}$,

дуга (9, 15) $\longrightarrow \pi_{15} - \pi_9 = c_{9,15} \Rightarrow L_{9,15} \leqslant f_{9,15} \leqslant U_{9,15}$,

дуга (15, 16) $\longrightarrow \pi_{16} - \pi_{15} < c_{15,16} \Rightarrow f_{15,16} = L_{15,16}$.

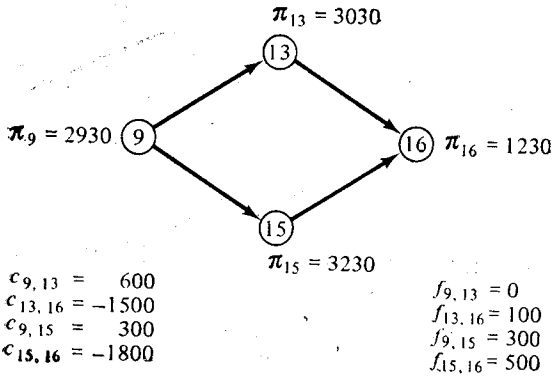


Рис. 3.19. Вторая подсеть.

Как следует из рис. 3.19, значения двойственных переменных, соответствующих дуге (9, 13), равны

$$\alpha_{9,13} = \max[0,\ 3030 - 2930 - 600] = 0,$$

$$\delta_{9,13} = \max[0,\ 2930 - 3030 + 600] = 500 \text{ центов.}$$

Для дуги (9, 15):

$$\alpha_{9,15} = \max[0,\ 3230 - 2930 - 300] = 0,$$

$$\delta_{9,15} = \max[0,\ 2930 - 3230 + 300] = 0.$$

Для дуги (13, 16):

$$\alpha_{13,16} = \max[0,\ 1230 - 3030 - (-1500)] = 0,$$

$$\delta_{13,16} = \max[0,\ 3030 - 1230 + (-1500)] = 300 \text{ центов.}$$

Для дуги (15, 16):

$$\alpha_{15,16} = \max[0,\ 1230 - 3230 - (-1800)] = 0,$$

$$\delta_{15,16} = \max[0,\ 3230 - 1230 + (-1800)] = 200 \text{ центов.}$$

Из этих соотношений для двойственных переменных следует, что в результате уменьшения нижних границ потока по дугам (9, 13) и (13, 16) на 1 единицу чистый выигрыш составит 800 центов в расчете на один стул. С другой стороны, для сохранения потока в узлах 9 и 16 потоки по дугам (9, 15) и (15, 16) следует увеличить на 1 единицу. Соответствующие затраты

составят 200 центов (если уменьшение потока на 1 единицу дает выигрыш в 200 центов, то увеличение потока на 1 единицу приводит к дополнительным затратам, равным 200 центам). Следовательно, в результате одновременного уменьшения нижних границ потока по дугам (9, 13) и (13, 16) на 1 единицу будет сэкономлено 600 центов. Этот результат может быть получен другим способом. В результате уменьшения потоков по дугам (9, 13) и (13, 16) на 1 единицу будет сэкономлено 600 центов и дополнительно затрачено 1500 центов соответственно. Аналогично в результате увеличения потоков по дугам (9, 15) и (15, 16) на 1 единицу будет дополнительно затрачено 300 центов и получена прибыль 1800 центов. Чистый выигрыш при этом составляет (600—1500—300+1800)=600 центов.

Такая экономическая интерпретация двойственных переменных в случае, когда стоимости дуг выражены в долларах или центах, явилась причиной того, что двойственные переменные часто называют *векторами цен.*

3.19. ЗАКЛЮЧЕНИЕ

Алгоритм дефекта применим к любой задаче, которая может быть сформулирована в виде задачи о циркуляции минимальной стоимости. При этом не требуется, чтобы в самом физическом процессе присутствовала циркуляция. Например, в транспортной задаче товар транспортируется только из источников в пункты назначения. Циркуляция образуется в результате построения «возвратной дуги» (t, s), поток по которой может не иметь физического смысла.

Не обязательно также, чтобы поток в задаче присутствовал явно. Например, задача о назначениях и задача о кратчайшей цепи являются комбинаторными, а не потоковыми. Циркуляция же используется как средство нахождения оптимального решения. Читателю следует внимательно изучить задачи, в которых поток или циркуляция не присутствуют явно и которые могут быть решены с помощью алгоритма дефекта.

Ключом к решению задачи с помощью алгоритма дефекта является сеть. С помощью сетей, по-видимому, может быть описано так много задач, что читатель скорее столкнется с необходимостью исправлять ошибки при применении этого алгоритма, чем искать задачи, где он может быть использован.

Следует помнить, что единственным средством моделирования для применения алгоритма дефекта является построение ориентированной сети. Каждая дуга описывается тремя числами: стоимостью единицы потока, нижней границей потока и верхней границей потока. Если заданы некоторые дополнительные условия, связывающие потоки по различным дугам (исклю-

чая условие сохранения потока), то для решения задачи алгоритм дефекта не может быть использован. Часто такие задачи могут быть описаны в виде задач линейного программирования и решены с помощью соответствующих программ. Кроме того, они могут быть сведены к задачам, для которых применим алгоритм дефекта.

Алгоритм дефекта может быть использован для решения широкого класса задач, и поэтому будущим инженерам следует вооружиться активными знаниями в области теории и программной реализации этого алгоритма.

ЧАСТЬ III. ПРИЛОЖЕНИЯ АЛГОРИТМА ДЕФЕКТА[1]

3.20. ПРОБЛЕМА УЗКИХ МЕСТ В ЗАДАЧЕ О НАЗНАЧЕНИЯХ

В качестве первого примера рассмотрим задачу, которую в исследовании операций часто называют *проблемой узких мест в задаче о назначениях*[2]. Опишем вначале наиболее часто встречающуюся задачу о назначениях. Имеются группа рабочих и некоторое количество станков. Известна эффективность работы каждого рабочего за каждым станком. Задача заключается в том, чтобы назначить на каждый станок ровно одного рабочего (или назначить рабочих на рабочие места, распределить учителей по классам и т. д.) и при этом максимизировать суммарную эффективность выполнения работ. Отметим, что, просто назначая каждого рабочего на тот станок, на котором эффективность его работы максимальна, мы, вообще говоря, не получим решения задачи, поскольку на одном и том же станке с наибольшей эффективностью могут работать несколько рабочих, а по условию задачи требуется, чтобы на каждый станок был назначен ровно один рабочий. Таким образом, возможен случай, когда при максимизации суммарной эффективности некоторые рабочие будут назначены на станки, на которых эффективность их работы не является максимальной. Как показано в разд. 3.12, эта задача может быть сформулирована и решена как потоковая задача. Оптимальный поток определит назначение, которое максимизирует суммарную эффективность. На рис. 3.20 изображена сеть, описывающая все возможные варианты назначения трех человек на три станка. Узлы 1, 2 и 3 соответствуют рабочим, а узлы 4, 5 и 6 — станкам.

---

[1] В части III использован материал, опубликованный в *Industrial Engineering* с разрешения Американского института инженеров-технологов. Примеры предложены Суонсоном, Вулсеем и Хиллисом [11].

[2] Далее проблему узких мест в задаче о назначениях будем называть кратко задачей на узкие места. — *Прим. ред.*

Перейдем теперь к задаче на узкие места, которая отличается от только что рассмотренной задачи о назначениях лишь тем, что в ней требуется максимизировать минимальную эффективность, определяемую назначением. Такая постановка является вполне реалистичной. Например, для линии последовательной сборки рабочий с минимальной производительностью (узкое место) определяет эффективность всего конвейера. Поэтому

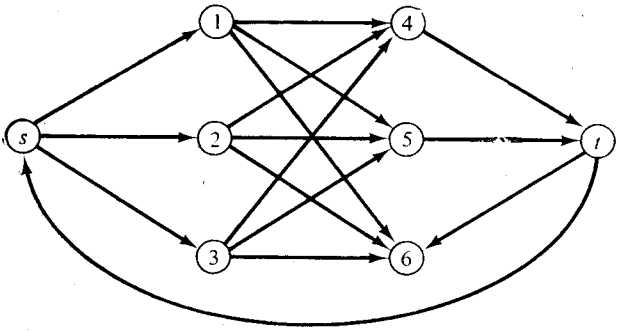


Рис. 3.20. Сеть в задаче о назначениях.

в данном случае в отличие от задачи нахождения назначения, максимизирующего суммарную эффективность, было бы желательно миксимизировать минимальную производительность, что определяет оптимальное назначение в данной задаче на узкие места. Задача на узкие места является сетевой и может быть решена последовательным решением соответствующих задач о назначении. Алгоритм описывается следующим образом:

1. Решить исходную задачу на узкие места как обычную задачу о назначениях. Это может быть легко сделано с помощью алгоритма дефекта. Поскольку алгоритм дефекта является *минимизирующим*, то следует минимизировать суммарную неэффективность (отрицательную эффективность), что эквивалентно максимизации эффективности.

2. Рассматривая оптимальное решение, полученное на шаге 1, определить назначение с максимальной неэффективностью (минимальной эффективностью). Если существует несколько таких назначений, то выбрать одно из них произвольным образом.

3. Исключить из сети, использованной на шаге 1, дуги, каждой из которых соответствует неэффективность (затраты), не меньшая неэффективности, определенной на шаге 2 (т. е. наименьшей эффективности, или наибольшей неэффективности, назначения, полученного на шаге 1).

4. Перейти на шаг 1, используя новую сеть, построенную на шаге 3.

5. Продолжать данный процесс до тех пор, пока нельзя будет произвести нужное число назначений (в результате того, что из исходной сети периодически исключается некоторое число дуг). Это станет видно тогда, когда в результате работы алгоритма дефекта обнаружится, что допустимого решения не существует. Таким образом, последнее из найденных назначений является оптимальным решением задачи на узкие места. Обоснованием этого утверждения является тот факт, что с помощью алгоритма дефекта решается последовательность задач о назначениях, в которой минимальная эффективность постоянно увеличивается (в результате исключения дуг). Данный алгоритм иллюстрируется на модельном примере, описанном в табл. 3.10.

Таблица 3.10. Матрица эффективности

	Рабочее место				
Рабочий	1	3	2	6	0
	4	2	3	8	3
	8	4	1	5	0
	3	5	4	8	8
	2	6	9	5	2

Элемент a_{ij} матрицы равен эффективности (числу деталей, изготавливаемых за 1 ч) выполнения задания i-м рабочим при назначении его на j-е рабочее место. Нулевые значения элементов a_{15} и a_{35} указывают на то, что рабочие 1 и 3 не имеют нужной квалификации для выполнения работы 5.

Решая задачу на узкие места как обычную задачу о назначениях (с 12 узлами и 36 дугами), с помощью алгоритма дефекта получаем следующее назначение:

Назначение 1

Рабочий 1⟶работа 2 с эффективностью 3
Рабочий 2⟶работа 4 с эффективностью 8
Рабочий 3⟶работа 1 с эффективностью 8
Рабочий 4⟶работа 5 с эффективностью 8
Рабочий 5⟶работа 3 с эффективностью 9

Суммарная эффективность равна 36; минимальная эффективность равна 3. Поэтому исключаем те элементы матрицы (и соответствующие им дуги сети), значение которых не превосходит 3. Теперь сеть состоит из 12 узлов и 24 дуг. Решая с помощью алгоритма дефекта задачу о назначениях для построенной сети, получаем следующее решение:

Назначение 2

Рабочий 1 ⟶ работа 4 с эффективностью 6
Рабочий 2 ⟶ работа 1 с эффективностью 4
Рабочий 3 ⟶ работа 2 с эффективностью 4
Рабочий 4 ⟶ работа 5 с эффективностью 8
Рабочий 5 ⟶ работа 3 с эффективностью 9

Суммарная эффективность теперь уменьшилась до 31, однако минимальная эффективность увеличилась до 4. Исключим далее те дуги сети, которым соответствует эффективность, не превосходящая 4. Теперь сеть содержит 12 узлов и 21 дугу. С помощью алгоритма дефекта определяем, что допустимого решения для данной сети не существует. Следовательно, назначение 2 является оптимальным в исходной задаче на узкие места. По сравнению с первоначальным назначением получен выигрыш на $33\frac{1}{3}\%$ (узкое место стало шире), хотя *суммарная* эффективность уменьшилась.

Это улучшение решения связано с тем, что из сети несколько раз удаляются дуги и задача решается заново. Таким образом, мы не только получаем оптимальное решение задачи на узкие места о назначениях, но и еще раз подтверждаем общеизвестное правило: больше работаешь — больше получаешь.

3.21. СОСТАВЛЕНИЕ ГРАФИКА ВЫПОЛНЕНИЯ ЗАДАНИЙ С ИЗВЕСТНЫМИ ВРЕМЕННЫМИ ХАРАКТЕРИСТИКАМИ

Рассмотрим задачу определения максимального числа рабочих, необходимого для выполнения фиксированного плана частично параллельных заданий. В другой постановке этой задачи требуется определить минимальное число станков, необходимое для выполнения плана заданий, если известно время наладки каждого станка, или определить минимальное число самолетов, необходимое для соблюдения заданного графика движения [1]. Решение данной задачи, найденное с помощью алгоритма дефекта, позволило определить для одного из районов минимальное число автобусов, требуемое для осуществления перевозок, маршрут и расписание которых были установлены местными властями [4]. Сетевая постановка задачи станет очевидной при рассмотрении следующего примера (принадлежащего Полю Йенсену, конструкторско-технологический отдел Техасского университета в г. Остине).

Должны быть выполнены десять заданий. Моменты начала и завершения выполнения каждого задания, а также отрезки времени, необходимые для перехода с одних рабочих мест на другие (время на подготовку), приведены в табл. 3.11. Требуется найти минимальное число рабочих, необходимое для выполнения всех заданий. Используя табл. 3.11, можно определить,

Таблица 3.11. Матрица времен выполнения заданий

Задание	Начало	Конец	1	2	3	4	5	6	7	8	9	10
1	13.00	13.30	—	60	10	230	180	20	15	40	120	30
2	18.00	20.00	10	—	40	75	40	5	30	60	5	15
3	22.30	23.00	70	30	—	0	70	30	20	5	120	70
4	16.00	17.00	0	50	75	—	20	15	10	20	60	10
5	16.00	19.00	200	240	150	70	—	15	5	240	90	65
6	12.00	13.00	20	15	20	75	120	—	30	30	15	45
7	14.00	17.00	15	30	60	45	30	15	—	10	5	0
8	23.00	24.00	20	35	15	120	75	30	45	—	20	10
9	20.10	21.00	25	60	15	10	100	70	80	60	—	120
10	13.45	15.00	60	60	30	30	120	40	50	60	70	—

какие задания являются «связанными». Два задания будем называть связанными, если выполнение одного из них и подготовку второго можно завершить до запланированного начала выполнения второго задания. Например, работа 1 связана с каждой из работ 2, 3, 7, 8 и 9, но не связана с работами 4, 5, 6 и 10 (работа 1 завершается в 13.30, а время подготовки для работы 4 равно 230 мин. или 3 ч. 50 мин. Складывая эти времена и учитывая, что работа 4 должна начаться в 16.00, получаем, что работы 1 и 4 не связаны). Аналогичным образом можно показать, что

Работа 1 связана с работами 2, 3, 7, 8, 9.
Работа 2 связана с работами 3, 8, 9.
Работа 3 не связана ни с одной работой.
Работа 4 связана с работами 2, 3, 8, 9.
Работа 5 связана с работами 3, 8.
Работа 6 связана с работами 2, 3, 4, 5, 7, 8, 9, 10.
Работа 7 связана с работами 2, 3, 8, 9.
Работа 8 не связана ни с одной работой.
Работа 9 связана с работами 3, 8.
Работа 10 связана с работами 2, 3, 4, 8, 9.

Отметим, что минимально допустимое число рабочих равно максимальному числу (несвязанных) заданий, никакие два из которых не могут быть выполнены одним и тем же человеком. Для решения данной задачи с использованием алгоритма дефекта мы построим «сеть назначений», с помощью которой одни задания будут «назначаться» на другие задания (эти назначенные задания объединяются таким образом, что они могут быть выполнены одним человеком). Кроме источника и стока, в сети должны быть два «столбца» узлов; число узлов в каждом столбце равно числу заданий. Наличие в сети дуги, соединяющей два заданных узла, зависит от того, являются ли эти узлы

связанными или нет. Так, узел в первом столбце, соответствующий заданию 1, соединен дугами с узлами второго столбца, соответствующими заданиям 2, 3, 7, 8 и 9. Узлы, соответствующие заданиям 3 и 8, можно не включать в первый столбец (но следует включать во второй столбец), поскольку эти задания не связаны ни с каким другим заданием (однако другие задания связаны с ним). Верхняя граница потока по «возвратной» дуге, или дуге «циркуляции» (т. е. дуге, соединяющей главный сток с главным источником), равна разности между общим числом заданий и числом заданий, не связанных ни с какими другими заданиями. Верхняя граница потока по всем остальным дугам равна 1. Нижняя граница потока по каждой дуге равна 0.

Цель — найти максимальный поток в построенной сети с ограниченной пропускной способностью. Для этого стоимость единицы потока по возвратной дуге полагается равной некоторому отрицательному числу (например, —10), а стоимость потока по всем остальным дугам — равной 0. Максимальный поток (оптимальное решение) в построенной сети определяет минимальное число рабочих, необходимое для выполнения плана заданий. В частности, с помощью дуг, поток по которым положителен, можно определить, какие задания следует объединить для выполнения их одним человеком. Число таких групп равно минимальному числу рабочих, требуемому для выполнения плана заданий. По следующим «назначающим дугам» поток, соответствующий оптимальному решению, положителен.

Задание 1⟶задание 7
Задание 4⟶задание 2
Задание 5⟶задание 3
Задание 6⟶задание 10
Задание 7⟶задание 9
Задание 9⟶задание 8
Задание 10⟶задание 4

Напомним, что дуга в сети строится только в том случае, если соответствующие узлы связаны. Поэтому найденное решение можно интерпретировать следующим образом: задание 1 связано с заданием 7, которое связано с заданием 9, которое связано с заданием 8.

Задание 1⟶задание 7⟶задание 9⟶задание 8. Эти четыре задания являются «объединенными» (каждое связано с последующим) и могут быть последовательно выполнены одним человеком. Эта «строка» заданий обрывается тогда, когда задание 8 не удается связать ни с одним другим заданием. Кроме того, имеются еще две «строки» заданий:

Задание 6⟶задание 10⟶задание 4⟶задание 2
Задание 5⟶задание 3

Таким образом, каждую из этих трех групп, или строк, заданий может выполнить один человек. При заданных ограничениях на время начала и время конца выполнения заданий, а также на время их подготовки меньше, чем трое рабочих, не смогут выполнить все 10 заданий.

3.22. ЗАДАЧА О ХРАНЕНИИ И СБЫТЕ ТОВАРА

Рассмотрим следующую задачу управления запасами. Владелец магазина приобрел 50 радиоприемников, заплатив по 20 долл. за каждый. Из-за непостоянства спроса на эти радиоприемники продажная цена их изменяется каждый месяц. Кроме того, каждый месяц изменяются затраты на хранение радиоприемников, поскольку изменяется их количество и условия хранения. Соответствующие данные приведены в табл. 3.12.

Таблица 3.12. Исходные данные в задаче о хранении и сбыте товара

Число радиоприемников		*0–10*	*11–25*	*26–45*	*46* и более
Затраты на хранение одного радиоприемника (долл.)					
Месяц	1	1,80	1,60	1,40	1,20
Месяц	2	2,60	2,40	2,20	2,00
Месяц	3	2,40	2,20	2,00	1,80
Продажная цена одного радиоприемника (долл.)					
Месяц	1	45,00	40,00	35,00	30,00
Месяц	2	40,00	35,00	30,00	25,00
Месяц	3	55,00	50,00	45,00	40,00
Месяц	4	50,00	45,00	40,00	35,00

Задача формулируется следующим образом: сколько радиоприемников надо продать в каждый из месяцев 1, 2, 3 и 4, чтобы максимизировать прибыль. На рис. 3.21 изображена сеть для данной задачи. Дуга, ведущая из узла 1 в узел 2, соответствует периоду, в котором радиоприемники были приобретены. Дуги (2, 3), (3, 4) и (4, 5) соответствуют хранению радиоприемников на складе в месяцы 1, 2 и 3 соответственно. Дуги (2, 6), (3, 6), (4, 6) и (5, 6) соответствуют продаже радиоприемников в месяцы 1, 2, 3 и 4 соответственно.

Как видно из исходных данных, рассматриваемая задача не является линейной и поэтому нельзя ожидать, что ее решение окажется простым. Однако эта задача, так же как и аналогичные ей задачи, имеет отличительную особенность, заключающуюся в том, что оптимальный поток протекает по одной единственной цепи, т. е. все радиоприемники следует продать в один и тот же период, а до этого времени они должны храниться на складе. Таким образом, задача свелась к определению месяца, подходящего для продажи радиоприемников. Зная эту особен-

ность задачи, можно использовать алгоритм дефекта для поиска оптимальной («максимальной») цепи в сети. Оптимальное решение задачи следующее: все радиоприемники следует хранить до 3-го месяца и затем продать их. Прибыль (в долл.) при этом составит 840 = (40—20—1,2—2,0) 50.

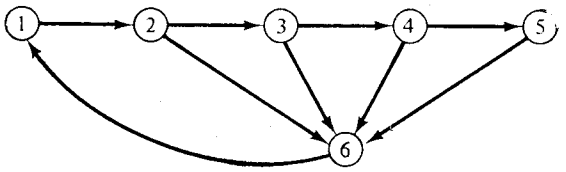


Рис. 3.21. Сеть в задаче о продаже радиоприемников.

3.23. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ АЛГОРИТМ ДЕФЕКТА

Назначение: нахождение оптимальных решений в различных потоковых задачах с сетями, имеющими ограниченную пропускную способность. Могут быть решены следующие задачи:

1) транспортная задача; 2) задача о назначениях; 3) задача о максимальном потоке; 4) задача о дереве кратчайших цепей; 5) задача о перевозках; 6) задача о максимальном потоке минимальной стоимости.

Локализация: подпрограмма OKALG в пакете сетевой оптимизации.

Ограничения: программа обрабатывает сети, содержащие до 500 узлов и 500 дуг. Размеры сети можно увеличить, изменив границы массивов в операторах размерности, записанных в подпрограмме OKALG и основной программе.

Входные данные:

Набор 1. Одна карта с именем алгоритма OKAL в формате (A4).

Набор 2. Одна карта с числом узлов и числом дуг в сети в формате (2I10).

Набор 3. Общее число карт в данном наборе равно числу дуг в сети. С каждой карты считываются следующие величины:
1) номер начального узла дуги; 2) номер конечного узла дуги; 3) верхняя граница потока по дуге; 4) нижняя граница потока по дуге; 5) стоимость прохождения единицы потока из начального узла дуги в конечный узел.

Формат (2I5, 3F10.2).

Набор 4. Данный набор состоит из одной карты, содержащей слово 'PEND' в формате (A4), которая указывает конец задачи.

Набор 5. Данный набор состоит из одной карты, содержащей

слово 'EXIT' в формате (A4), которая указывает конец входных данных.

Составные части программы. Программа состоит из следующих подпрограмм: OKALG (ввод, вывод и вычисления); NETFLOW (процедура расстановки пометок).

Используемые переменные:

I — начальный узел дуги,
J — конечный узел дуги,
HI — верхняя пропускная способность этой дуги,
LO — нижняя пропускная способность этой дуги,
FLOW — поток по этой дуге,
PI — значения двойственных переменных,
COST — стоимость прохождения единицы потока по этой дуге.

Используемый метод: данный алгоритм основан на методе, описанном в разд. 3.3—3.5.

Литература: [6].

3.24. РЕКТИФИКАЦИЯ И РАСПРЕДЕЛЕНИЕ НЕФТИ

Три нефтеочистительных завода могут получать неочищенную нефть из двух месторождений, первое из которых расположено в заливе Аляска, а второе — в Персидском заливе. Стоимость одного барреля нефти в этих месторождениях составляет 16,80 и 12,60 долл. соответственно (стоимость галлона — 0,40 и 0,30 долл.). Из обоих месторождений нефть может поставляться в неограниченном количестве, но не менее 300 000 галлонов в день. Производственная мощность и затраты на ректификацию одного галлона нефти для каждого из заводов указаны в табл. 3.13.

Таблица 3.13. Стоимость ректификации и производительность заводов

Нефтеочистительный завод	Затраты на ректификацию (долл./галлон)	Максимальная производительность (галлоны)	Минимальная производительность (галлоны)
1	0,03	1 250 000	200 000
2	0,04	1 500 000	300 000
3	0,05	1 350 000	300 000

Эти заводы должны снабжать нефтью четыре района. В табл. 3.14 указаны затраты (в долл.) на транспортировку одного галлона нефти. В табл. 3.15 даны суточная потребность в нефти каждого из районов и продажная цена одного галлона.

Для максимизации своей прибыли нефтяная компания долж-

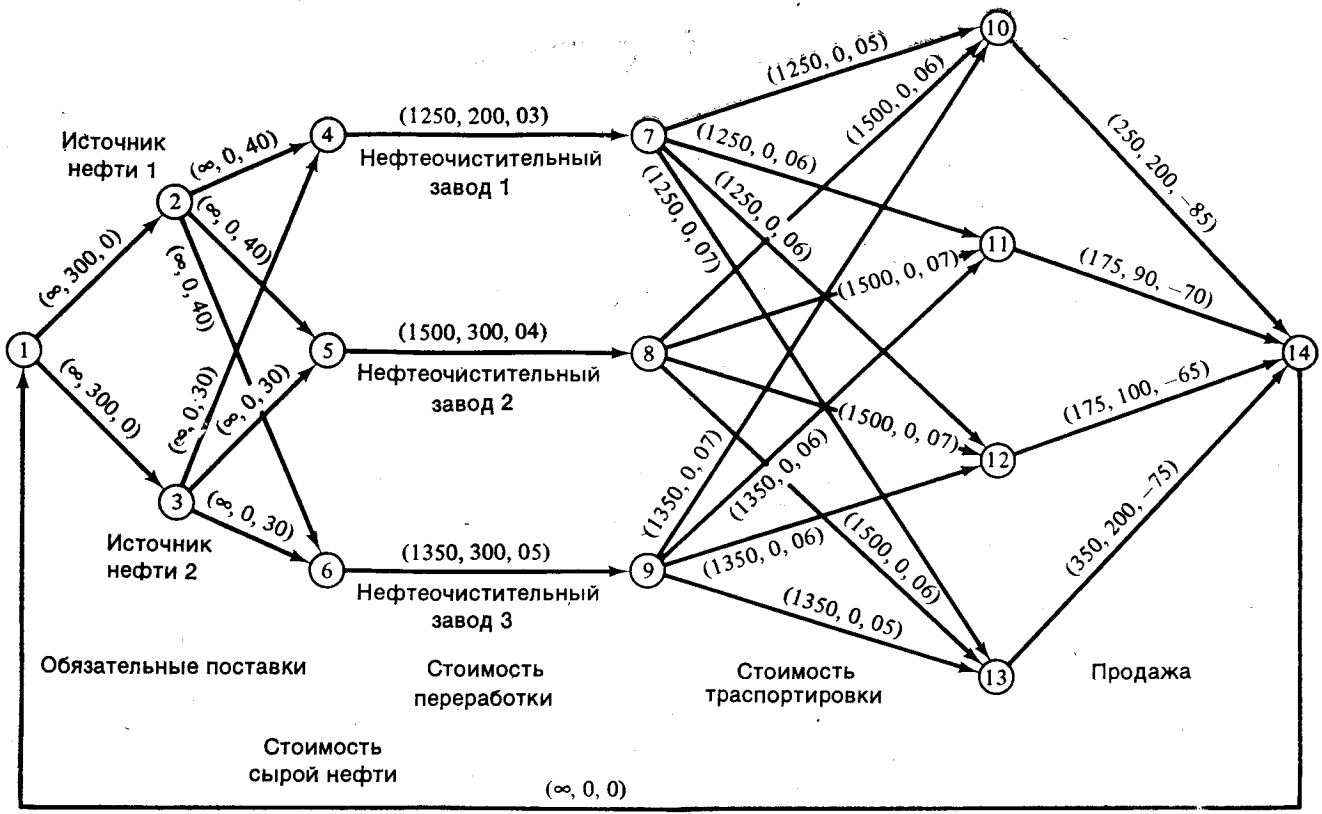


Рис. 3.22. Переработка и распределение нефти.

Таблица 3.14. Затраты на транспортировку

Нефтеочистительный завод	Нью-Йорк	Атланта	Даллас	Лос-Анджелес
1	0,05	0,06	0,06	0,07
2	0,06	0,07	0,07	0,06
3	0,07	0,06	0,06	0,05

на поставлять в указанные районы наиболее дешевую нефть при минимальных затратах на транспортировку. Сетевая диаграмма изображена на рис. 3.22. Решение задачи, найденное с помощью программы, описанной в разд. 3.23, дается ниже.

Решение, полученное с помощью алгоритма дефекта

Краткий отчет

Число узлов = 14

Число дуг = 28

M	I	J	HI	LO	Поток	Стоимость
1	1	2	9999999	300	300	0
2	1	3	9999999	300	650	0
3	2	4	9999999	0	0	40
4	2	5	9999999	0	0	40
5	2	6	9999999	0	300	40
6	3	4	9999999	0	350	30
7	3	5	9999999	0	300	30
8	3	6	9999999	0	0	30
9	4	7	1250	200	350	3
10	5	8	1500	300	300	4
11	6	9	1350	300	300	5
12	7	10	1250	0	200	5
13	7	11	1250	0	0	6
14	7	12	1250	0	150	6
15	7	13	1250	0	0	7
16	8	10	1500	0	50	6
17	8	11	1500	0	175	7
18	8	12	1500	0	25	7
19	8	13	1500	0	50	6
20	9	10	1350	0	0	7
21	9	11	1350	0	0	6
22	9	12	1350	0	0	6
23	9	13	1350	0	300	5
24	10	14	250	200	250	−84
25	11	14	175	90	175	−69
26	12	14	175	100	175	−64
27	13	14	350	200	350	−74
28	14	1	9999999	0	950	0

Суммарные затраты на осуществление проекта = −29525,00

Дуга	Поток по дуге
1	300
2	650
3	0
4	0
5	300
6	350
7	300
8	0
9	350
10	300
11	300
12	200
13	0
14	150
15	0
16	50
17	175
18	25
19	50
20	0
21	0
22	0
23	300
24	250
25	175
26	175
27	350
28	950

Узел	Узловое число PI
1	47
2	37
3	47
4	77
5	77
6	77
7	80
8	79
9	80
10	85
11	86
12	86
13	85
14	47

Проверка точности

M	I	J	HI	LO	Поток	Стоимость
1	1	2	9999999	300	300	0
2	1	3	9999999	300	650	0
5	2	6	9999999	0	300	40
6	3	4	9999999	0	350	30
7	3	5	9999999	0	300	30
9	4	7	1250	200	350	3
10	5	8	1500	300	300	4
11	6	9	1350	300	300	5
12	7	10	1250	0	200	5
14	7	12	1250	0	150	6
16	8	10	1500	0	50	6
17	8	11	1500	0	175	7
18	8	12	1500	0	25	7
19	8	13	1500	0	50	6
23	9	13	1350	0	300	5
24	10	14	250	200	250	−84
25	11	14	175	90	175	−69
26	12	14	175	100	175	−64
27	13	14	350	200	350	−74
28	14	1	9999999	0	950	0

Отметим, что оптимальное решение имеет *отрицательную* стоимость. Это означает, что при оптимальной организации процесса производства и распределения капитал будет *увеличиваться* (получение прибыли), поскольку затраты обозначались положительными числами, а продажные цены — отрицательными. Соответствующая оптимальная структура производства и распределения легко определяется из найденных потоков по дугам.

Таблица 3.15. Суточная потребность

Город	Максимальная потребность	Минимальная потребность	Продажная цена (долл./галлон)
Нью-Йорк	250 000	200 000	0,85
Атланта	175 000	90 000	0,70
Даллас	175 000	100 000	0,65
Лос-Анджелес	350 000	200 000	0,75

УПРАЖНЕНИЯ

1. Сформулировать перечисленные ниже задачи и построить для каждой из них циркуляционную модель:

а) транспортная задача (пять источников и четыре стока);

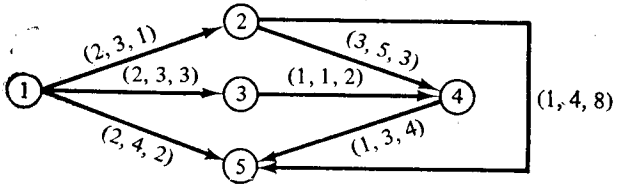
б) задача о назначениях (три источника и три стока);
в) задача о максимальном потоке (в общем виде);
г) задача о кратчайшей цепи (в общем виде);
д) задача о дереве кратчайших цепей (сеть состоит из четырех узлов, каждый из которых соединен со всеми другими);
е) задача о перевозках (четыре источника и четыре стока).

2. Модель производственного планирования. Фирма должна удовлетворять спрос на продукцию в течение трех периодов. В каждом периоде продукция может выпускаться в рабочее и в сверхурочное время. Заданы следующие исходные данные:

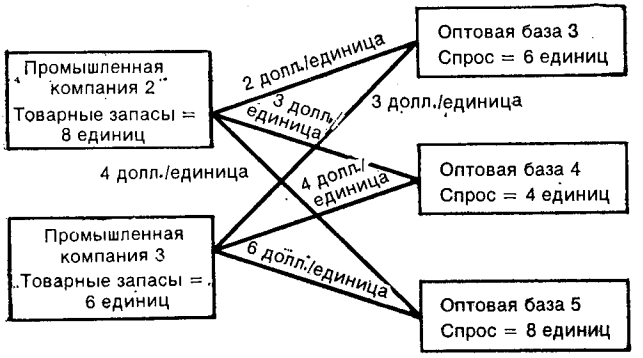
Период	Производительность (в единицах)		Стоимость производства единицы продукции		Ожидаемый спрос (в единицах)
	Рабочее время	Сверхурочное время	Рабочее время	Сверхурочное время	
1	100	20	14	18	60
2	100	10	17	22	80
3	60	20	17	22	140

Стоимость хранения единицы продукции на складе в течение одного периода равна 1 долл. Уровень запасов в начале первого периода составляет 15 единиц. Сформулировать данную задачу как циркуляционную и решить ее, используя алгоритм дефекта. Сформулировать данную задачу как транспортную и решить ее, используя алгоритм дефекта.

3. В следующей циркуляционной задаче найти максимальный поток из узла 1 в узел 5. В качестве начального решения взять нулевой поток и нулевые значения для всех двойственных переменных π. Каждой дуге приписана тройка (L_{ij}, U_{ij}, c_{ij}).



4. Следующую транспортную задачу сформулировать и решить как циркуляционную.



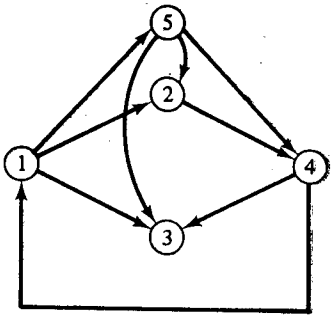
5. Для изображенной ниже сети сформулировать прямую и двойственную задачи и записать условия дополняющей нежесткости.

Исходные данные *Переменные*

Стоимость $= c_{ij}$ $f_{ij} \geq 0$

Нижняя граница $= L_{ij}$

Верхняя граница $= U_{ij}$



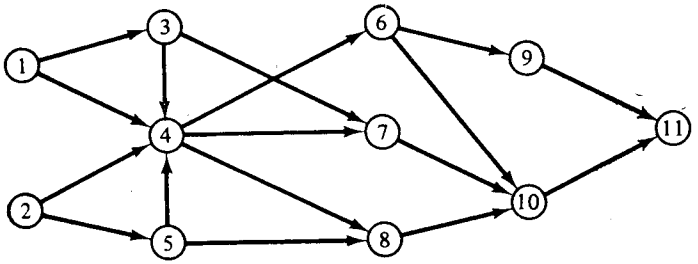
6. Рассмотреть заданную ниже сеть. Используя программу, реализующую алгоритм дефекта, решить следующие задачи:

Дуга	Верхняя граница	Нижняя граница	Стоимость единицы потока, долл.
(1, 3)	20	0	12
(1, 4)	15	0	8
(2, 4)	17	0	13
(2, 5)	32	0	9
(3, 4)	41	0	10
(3, 7)	27	0	15
(4, 6)	∞	0	0
(4, 7)	16	0	7
(4, 8)	19	0	9
(5, 4)	23	0	11
(5, 8)	29	0	5
(6, 9)	31	0	7
(6, 10)	14	0	10
(7, 10)	19	0	8
(8, 10)	28	0	14
(9, 11)	34	0	12
(10, 11)	29	0	11

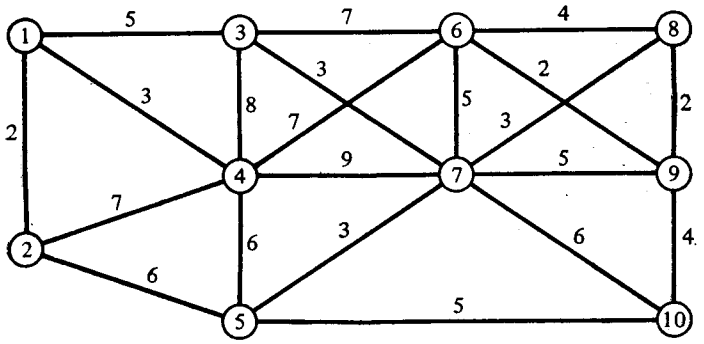
а. Найти цепь из узла 1 в узел 11, имеющую минимальную стоимость.

б. Найти потоки по дугам, обеспечивающие доставку 50 единиц потока в узел 11.

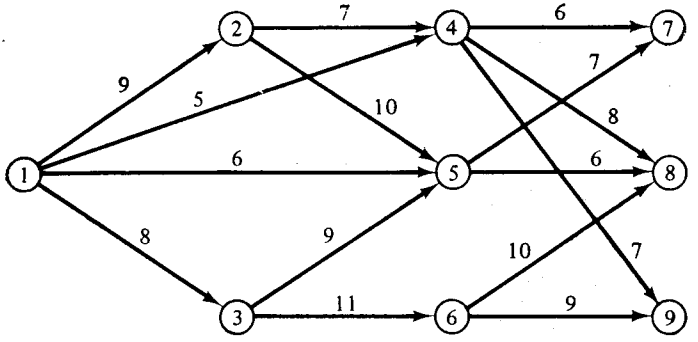
в. Найти максимальный поток, который может протекать из узлов 1 и 2 в узел 11.



7. Используя алгоритм дефекта, найти кратчайший остов изображенной ниже неориентированной сети. Параметр каждой дуги равен стоимости единицы потока.



8. Для изображенной ниже сети найти кратчайший древовидный остов с корнем в узле 1 (см. определения, данные в гл. 1).

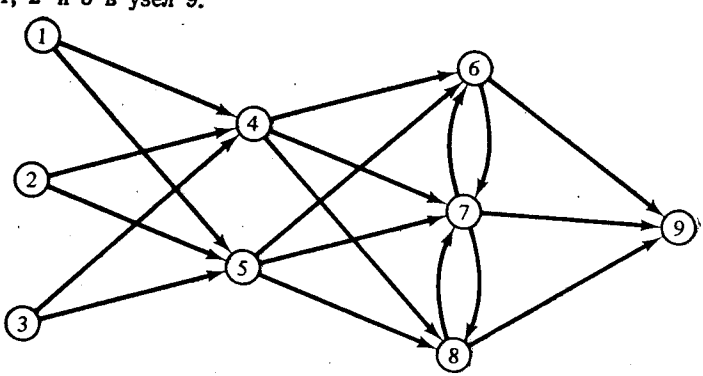


9. Рассмотреть заданную ниже сеть. Используя программу, реализующую алгоритм дефекта, решить следующие задачи:

Дуга	Верхняя граница	Нижняя граница	Стоимость ед. потока, долл.
(1, 4)	95	0	46
(1, 5)	86	0	29
(2, 4)	91	0	31
(2, 5)	78	0	47
(3, 4)	98	0	45
(3, 5)	77	0	53
(4, 5)	81	0	29
(5, 4)	86	0	37
(4, 6)	90	0	41
(4, 7)	79	0	36
(4, 8)	96	0	30

Дуга	Верхняя граница	Нижняя граница	Стоимость единицы потока
(5, 6)	77	0	29
(5, 7)	95	0	43
(5, 8)	84	0	27
(6, 7)	82	0	31
(7, 6)	79	0	30
(7, 8)	91	0	29
(8, 7)	85	0	43
(6, 9)	84	29	69
(7, 9)	80	37	68
(8, 9)	93	31	64

а. Найти циркуляцию минимальной стоимости из узлов 1 и 3 в узел 9.
б. Найти величину максимального потока минимальной стоимости из узлов 1, 2 и 3 в узел 9.

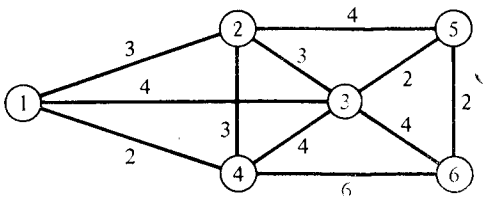


10. Какой результат будет получен при использовании алгоритма дефекта, если стоимости дуг отрицательны?

11. При работе алгоритма дефекта в качестве начального можно выбирать любое (допустимое или недопустимое) решение, удовлетворяющее условию сохранения потока. Какой выигрыш будет получен, если в качестве начального использовать допустимое решение? Сформулировать все условия дополняющей нежесткости.

12. В чем различие между кратчайшим остовом и деревом кратчайших цепей? Между деревом кратчайших цепей и кратчайшим древовидным остовом?

13. В изображенной ниже сети найти дерево кратчайших цепей и сравнить его с кратчашим остовом.



14. Сформулировать задачу о поставщике (разд. 2.16) как (а) обычную потоковую задачу и (б) транспортную задачу, для решения которой может быть использован алгоритм дефекта.

15. Решить упражнение 51 гл. 2, используя алгоритм дефекта.

16. Решить упражнение 52 гл. 2, используя алгоритм дефекта.

17. Фирма Alamo получила заказ на кирпич, производимый на ее заводе в Сан-Антонио, шт. Техас. Заказ должен быть выполнен в ближайшие 24 ч, а время на выполнение одного рейса до заказчика и обратно составляет 8 ч. Фирма располагает четырьмя грузовиками, имеющими следующую грузоподъемность (в т).:

Грузовики, принадлежащие фирме Alamo	1	2	3	4
Грузоподъемность	15	12	13	16

Для выполнения заказа фирма Alamo может арендовать у фирмы Glider Rent-A-Truck еще три грузовика, имеющие следующую грузоподъемность (в т):

Грузовики, принадлежащие фирме Glider Rent—A—Track	1	2	3
Грузоподъемность	7	8	6

Эксплуатационные расходы для каждого грузовика следующие:

	1	2	3	4
Alamo	25	31	22	30
Glider	44	48	41	—

Всего было заказано 200 т кирпича.

а. Определить допустимую схему перевозки, минимизирующую суммарные расходы.

б. Чему равно максимальное количество кирпича, которое можно перевезти за 3 дня?

в. Заказчик потребовал доставить ему не 200, а 300 т кирпича. Фирма Alamo может приобрести кирпич стоимостью 52 долл. за 10 т у завода Rio Grande Brick Factory. Определить для данного случая схему перевозки, имеющую минимальную стоимость.

18. Tangled Web Production Company владеет тремя компаниями, каждая из которых может поставлять товар в четыре различных пункта сбыта. Издержки производства нелинейные, но могут быть вычислены. Максимальная производительность каждого из трех пунктов производства равна 25 единицам. В табл. 1 даны предельные и общие переменные издержки производства. В табл. 2 даны затраты на транспортировку единицы товара из каждо-

Таблица 1. ПРЕДЕЛЬНЫЕ ИЗДЕРЖКИ ПРОИЗВОДСТВА

	Пункт производства I		Пункт производства II		Пункт производства III	
Уровень производства	Предельные издержки	Общие перемен. издержки	Предельные издержки	Общие перемен. издержки	Предельные издержки	Общие перемен. издержки
1	50	50	60	60	54	54
2	51	101	60	120	54	108
3	51	152	60	180	55	163
4	52	204	60	240	55	218
5	52	256	61	301	56	274
6	53	304	61	362	56	330
7	54	363	62	424	57	387
8	55	418	62	486	57	444
9	57	475	62	548	58	502
10	58	533	63	611	58	560
11	60	593	63	674	58	618
12	61	654	64	738	59	677
13	62	716	64	802	60	737
14	64	780	64	866	61	798
15	65	845	64	930	62	860
16	66	911	65	995	63	923
17	67	978	65	1060	63	986
18	68	1046	66	1126	64	1050
19	69	1115	66	1192	64	1114
20	70	1185	67	1259	65	1179
21	72	1257	67	1326	65	1244
22	75	1352	67	1393	66	1310
23	80	1412	69	1462	66	1376
24	100	1512	70	1532	67	1443
25	120	1632	80	1612	70	1513

Таблица 2. МАТРИЦА ЗАТРАТ НА ТРАНСПОРТИРОВКУ

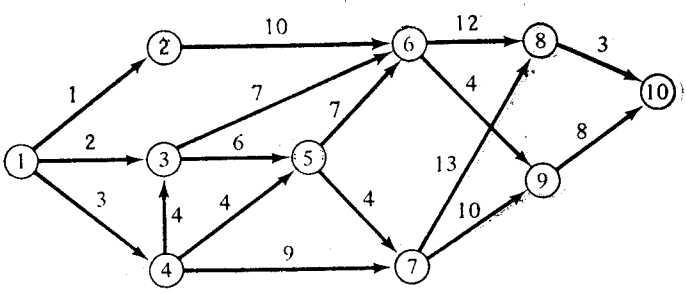
Из пункта производства	В пункт сбыта			
	A	B	C	D
I	20 долл.	20 долл.	12 долл.	30 долл.
II	45	35	10	20
III	22	10	30	55

Таблица 3. СПРОС ПУНКТОВ СБЫТА

Пункт сбыта	A	B	C	D	Общий спрос
Спрос	10	10	6	9	35

го пункта производства в каждый пункт сбыта. В табл. 3 даны величины спроса каждого из четырех секторов потребления. Найти оптимальный (минимизирующий общие затраты) план производства и перевозок.

19. ABC Computer Company располагает 10 программами, с помощью которых может быть решена задача. Требуется получить выход программы 10, на вход которой может поступать выход программы 8 или программы 9. В первом случае время работы программы 10 равно 3 с, в во втором — 8 с. Точно так же на вход программ 8 и 9 и всех остальных программ поступает выход некоторой другой программы. Задача заключается в определении набора программ, в результате работы которых будут получены требуемые результаты за минимальное время. На изображенной ниже сетевой диаграмме указаны отношения предшествования и время работы каждой программы, на вход которой должен поступать выход любой из предшествующих ей программ. Найти последовательность выполнения программ, минимизирующую общее время выполнения.



20. Решить упражнение 51 гл. 2, предполагая, что возможны следующие перевозки. Элементы таблицы равны затратам в сотнях долл.

Из \ В	Шахта					
		1	2	3	4	5
Шахта	1	—	7	4	3	—
	2	6	—	5	7	—
	3	5	5	—	9	—
	4	3	8	10	—	—
Завод	1	—	4	7	2	6
	2	4	—	8	9	2
	3	7	8	—	3	1
	4	3	10	3	—	6
	5	5	2	1	5	—

21. Решить упражнение 35 гл. 2, используя программу, реализующую алгоритм дефекта, и сравнить время ее работы со временем работы программы, реализующей алгоритм Дейкстры.

22. Решить упражнения 28а, б гл. 2, используя алгоритм дефекта.

23. Решить упражнение 29а гл. 2, используя алгоритм дефекта.

24. Решить упражнение 21а гл. 2, используя алгоритм дефекта, и сравнить полученный результат с результатом работы алгоритма, данного в гл. 2. Смогли бы вы еще решить задачи б и в?

ЛИТЕРАТУРА

1. Durbin E. P., The Out-of-Kilter Algorithm: A Primer, Rand Corporation, Santa Monica, California, December 1967.
2. Ford L. R., Fulkerson D. R., Maximal Flow through a Network, *Canadian Journal of Mathematics* (August 1956).
3. Ford L. R., Fulkerson D. R., Flows in Networks, Princeton University Press, Princeton, N. J., 1962. [Имеется перевод: Форд Л. Р., Фалкерсон Д. Потоки в сетях. — М.: Мир, 1966.]
4. Fulkerson D. R., The Out-of-Kilter Method for Minimal Cost Flow Problems, *Journal of Applied Mathematics*, 9 (1) (March 1961).
5. Jensen P. A., Barnes W., Network Flow Programming, Wiley, Inc., New York, 1979.
6. Phillips D. T., Jensen P. A., Network Flow Optimization with the Out-of-Kilter Algorithm, Part I — Theory, Research Memorandum 71-2, February 1971.
7. Phillips D. T., Jensen P. A., Network Flow Optimization with the Out-of-Kilter Algorithm, Part II — Applications, Research Memorandum 71-3, Purdue University, February 1971.
8. Phillips D. T., Jensen P. A., Neetwork Flow Analysis: The Qut-of-Kilter Algorithm, Industrial Engineering (February 1974). Portions reproduced by permission of the authors and the American Institute of Industrial Engineers.
9. Phillips D. T., Ravindran A., Solberg J. J., Operations Research: Principles and Practice, Wiley, Inc., New York, 1977.
10. Saha J. L., An Algorithm for Bus Scheduling Problems, *Operational Research Quarterly*, 21 (4) (December 1970).
11. Swanson H. S., Woolsey R. E. D., Hillis H., Using the Out-of-Kilter Algorithm, *Industrial Engineering* (March 1974). Portions reproduced by permission of the authors and the American Institute of Industrial Engineers.
12. Swanson H. S., Woolsey R. E. E., An Out-of-Kilter Network Tutorial, *SIGMAP Newsletter* (January 1973).
13. Vajda, Mathematical Programming, Addison-Wesley Publ. Co., Inc., Reading, Mass., 1961.

Глава 4

МЕТОДЫ УПРАВЛЕНИЯ ПРОЕКТАМИ

Я был бы рад стать лучше, — сказал
он, — но не знаю, как этого добиться.
Что мне нужно делать?

К. Джексон, Б. Джексон. Толстый неуклюжий слоненок

Нам, находясь в положении «толстого неуклюжего слоненка», стоит задуматься, как существенно повысить эффективность вычислений, используя специфику определенного класса задач о потоках в сетях. Такое повышение эффективности может быть достигнуто при использовании сетевых моделей для изображения крупных строительных или конструкторских проектов и управления ими. При управлении проектами с помощью МКП (метод критического пути) или ПЕРТ (метод оценки и пересмотра планов) и при календарном планировании работ применяются специальные способы изображения сетей. В частности, при этом используются ориентированные дуги и бесконтурные сети. Благодаря такой специальной структуре были разработаны очень эффективные и простые процедуры вычислений для получения важной информации о состоянии проекта. Эти специальные методы и связанные с ними задачи управления затратами и ресурсами и составляют основное содержание этой главы, которая посвящена не только тому, «как этого добиться», но и, когда определен класс задач, тому, «что нужно делать». Глава разбита на три части. Часть I посвящена схематическому изображению, построению сетевых моделей и процедурам вычислений, используемым при анализе с помощью МКП и ПЕРТ. В части II обсуждаются вопросы управления ресурсами и затратами с помощью сетевых методов. В части III описано математическое обеспечение ЭВМ, имеющееся для решения задач этого класса.

ЧАСТЬ I. УПРАВЛЕНИЕ ПРОЕКТАМИ С ПОМОЩЬЮ МКП И ПЕРТ

Результаты получены при участии
Уоррена Томаса,
Университет в Нью-Йорке

Управление крупными проектами, состоящими из большого числа взаимосвязанных работ, сопряжено с решением сложных проблем планирования, установления сроков и контроля, осо-

бенно когда работы должны выполняться в заданной технологической последовательности. С помощью ПЕРТ и МКП руководитель проекта может:

1. Заблаговременно планировать работу над проектом и предвидеть возможные источники затруднений и задержки выполнения его в срок.

2. Планировать завершение работ в нужные сроки в соответствии с требуемой последовательностью выполнения заданий с целью быстрейшего осуществления проекта.

3. Координировать и контролировать выполнение работ для соблюдения календарного графика и завершения проекта в срок.

С принципами, лежащими в основе сетевых методов, тесно связаны задачи распределения и использования ресурсов, сокращения сроков выполнения отдельных работ с целью уменьшения общей продолжительности проекта и анализа допустимых задержек, называемых резервом времени, для дальнейшей координации осуществления проекта. Это позволяет руководителю проекта выполнять следующие задачи:

1. Устанавливать последовательность и сроки использования ограниченных ресурсов в течение всего периода осуществления проекта.

2. Проводить динамическое регулирование сроков начала каждой работы.

3. Осуществлять оптимальное распределение средств, выделенных на проект, с целью сократить продолжительность всего проекта.

4. Выполнять анализ компромиссных соотношений между затратами и сроками выполнения различных работ с учетом имеющегося резерва времени.

В данной главе рассматриваются методы, необходимые для достижения этих целей, приводятся примеры алгоритмических процедур и обсуждаются способы получения решений с помощью ЭВМ.

4.1. ПОЯВЛЕНИЕ И ПРИМЕНЕНИЕ ПЕРТ

ПЕРТ был создан в конце 50-х годов в военно-морских силах США для ускорения разработки лодочной баллистической ракеты «Поларис». При разработке этой системы оружия требовалось координировать работу нескольких тысяч частных подрядчиков и правительственных организаций. Координация работ с помощью ПЕРТ оказалась настолько успешной, что весь проект был завершен на два года раньше планового срока. Это привело к дальнейшему применению ПЕРТ в других програм-

мах разработки оружия в ВМС, ВВС и сухопутных войсках США. В настоящее время он широко применяется в промышленности, а также в обслуживающих организациях.

Обычно при осуществлении научных исследований и разработок заранее неизвестно время, необходимое для выполнения различных работ. Поэтому при использовании ПЕРТ учитывается неопределенность в задании продолжительности работ. Метод позволяет определить вероятность завершения различных этапов проекта в заданный срок, а также вычислить ожидаемую продолжительность проекта. Важным и исключительно полезным результатом применения ПЕРТ является определение узких мест проекта. Иначе говоря, выявляются те работы, которые с большей вероятностью способны вызвать задержку сроков завершения проекта. Таким образом, еще до начала работ руководитель проекта знает, где могут ожидаться задержки. Он имеет возможность заранее принять необходимые меры с целью устранить возможные задержки и обеспечить осуществление проекта в срок.

ПЕРТ широко используется в научно-исследовательских и опытно-конструкторских проектах, так как позволяет учитывать неопределенность сроков выполнения работ.

4.2. ПОЯВЛЕНИЕ И ПРИМЕНЕНИЕ МКП

МКП во многих отношениях напоминает ПЕРТ, но был разработан независимо от него фирмой «Дюпон де Немур». Фактически оба этих метода — ПЕРТ и МКП — разработаны почти одновременно. Основное различие между ними состоит в том, что МКП не учитывает случайные колебания продолжительности работ. Вместо этого предполагается, что продолжительность работы пропорциональна количеству выделяемых ресурсов и что, изменяя количество ресурсов, можно изменять продолжительность работы и сроки завершения проекта. Таким образом, при использовании МКП на основе имеющегося опыта осуществления аналогичных проектов устанавливаются соотношения между имеющимися ресурсами и продолжительностями работ. Затем оцениваются компромиссные соотношения между затратами и продолжительностью проекта.

МКП применяется главным образом в строительных проектах, когда имеется опыт выполнения аналогичных работ. Хотя ПЕРТ и МКП разработаны и появились независимо друг от друга, между ними много общего. На практике ПЕРТ иногда называют методом критического пути, и наоборот. Рассматриваемые ниже определения и процедуры относятся как к ПЕРТ, так и к МКП.

4.3. ПОСТАНОВКА ЗАДАЧИ

МКП/ПЕРТ применимы к проектам, когда для достижения определенной цели должна выполняться упорядоченная последовательность заданий. Примерами таких проектов могут быть строительство здания или другого сооружения, выполнение крупных ремонтных работ, разработка и создание сложной системы оружия, изготовление крупной единицы оборудования и выполнение научно-исследовательской или опытно-конструкторской работы.

Эти проекты имеют ряд общих характеристик.

1. Они состоят из хорошо определенной совокупности заданий, выполнение которых означает завершение проекта.

2. Задания упорядочены таким образом, что они должны выполняться в определенной последовательности.

3. Продолжительность выполнения каждого задания известна заранее либо может быть оценена достаточно точно. При решении задачи с помощью МКП предполагается, что продолжительность выполнения задания имеет единственное значение, которое известно заранее (или может быть оценено достаточно точно). В случае ПЕРТ вносится большая неопределенность, так как плановому органу разрешается устанавливать верхний и нижний пределы продолжительности выполнения каждого задания. Единственным различием между методом критического пути и методом ПЕРТ является способ задания срока завершения работы.

4. Предполагается, что начатая работа продолжается без перерыва до завершения.

5. Выполнение последующей работы не обязательно должно начинаться сразу же после завершения непосредственно предшествующей ей, однако оно не может начинаться, пока не будет завершена предыдущая работа. Это требование вызывает некоторые трудности применения МКП и ПЕРТ в отраслях с непрерывным производственным циклом, где не допускается перерыв между этапами производства.

В принципе при использовании сетевых методов продолжительность проекта может быть определена способами, описанными ранее в этой книге. Действительно, минимальная продолжительность проекта определяется последовательностью работ, составляющих самый длинный путь через сеть. Он называется *критическим путем*, а составляющие его работы — *критическими работами*. Это объясняется тем, что любое увеличение их продолжительности или любая задержка в их выполнении увеличивают время осуществления всего проекта. Критические работы играют важную роль в методологии сетевых методов, и здесь детально не рассматриваются. Однако важно заметить,

что в силу специфики структуры каждой сетевой модели существуют более простые способы описания продолжительности проекта, чем обычные алгоритмы максимального потока. Заметим, что сеть, анализируемая с помощью МКП/ПЕРТ, имеет следующую структуру:

1. Каждая дуга имеет определенную ориентацию.
2. Сеть проекта является бесконтурной.
3. Не допускается вероятностное ветвление.
4. Какая-либо работа не может начинаться раньше, чем будут завершены все предыдущие работы.
5. Всегда можно так пометить каждую работу, что ее ориентированная дуга начинается в узле, имеющем *меньший* номер, чем узел, в котором она заканчивается. Вследствие такой специальной структуры сети для анализа сетей проекта и определения критического пути может быть использован очень эффективный алгоритм. Далее будет рассмотрена такая процедура.

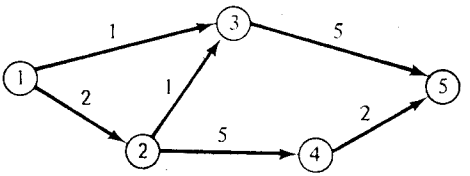


Рис. 4.1. Сетевая модель работ.

Для пояснения последующего изложения рассмотрим изображенный на рис. 4.1 отрезок сети, содержащий последовательность из шести работ — по одной для каждой дуги. Цифра на каждой дуге обозначает время, необходимое для выполнения каждой работы. Заметим, что узел 5 не может быть достигнут, пока не будут завершены работы (3, 5) и (4, 5). Существуют три различных пути, которые должны быть пройдены для завершения проекта.

I. ① → ③ → ⑤
II. ① → ② → ④ → ⑤
III. ① → ② → ③ → ⑤

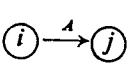
Продолжительность пути I равна 6 единицам, пути II—9 единицам, а пути III — 8 единицам. Как уже указывалось ранее, путь II называется *критическим*, так как необходимо своевременное выполнение операций, лежащих на этом пути, чтобы завершить проект без задержки. Поскольку работы (1, 2), (2, 4) и (4, 5) находятся на критическом пути, нельзя допускать задержки их начала или окончания. Это *критические работы.* Иными словами, здесь отсутствует какой-либо резерв времени как для наиболее раннего возможного начала, так и для наибо-

лее позднего допустимого начала этих работ. Каждая работа *должна начинаться в установленный срок.* Следовательно, наиболее ранний возможный и наиболее поздний допустимый сроки начала работы совпадают. Аналогичным образом любая работа может быть охарактеризована в соответствии с ее наиболее ранним возможным и наиболее поздним допустимым сроками начала. Например, для работ (1, 3) и (3, 5) наиболее ранний возможный срок начала равен соответственно 0 и 3. Поскольку не требуется, чтобы работа (3, 5) завершилась до момента $T=9$, наиболее поздний допустимый срок начала работы (3, 5) равен 4. Заметим, что соответственно наиболее поздний допустимый срок начала работы (1, 3) равен 3 единицам времени. Следовательно, *резерв времени* для работ (1, 3) и (3, 5), определяемый как разность между наиболее поздним допустимым и наиболее ранним возможным сроками начала работы, равен соответственно 3 единицам и 1 единице. Следуя этой же логике, читатель может проверить, что резерв времени для работы (2, 3) равен 1 единице. Для более крупных сетей проще вычислять резерв времени в табличном виде. Теперь изложим методику и математический аппарат такой процедуры.

4.4. ПОСТРОЕНИЕ СЕТИ

Сетевая модель проекта представляет собой графическое описание плана, показывающее взаимосвязь между всеми заданиями, выполнение которых необходимо для завершения проекта. Сеть состоит из ориентированных дуг, соединяющих пару узлов. Элементы сети, характеризуемые затратами времени (дуги), называются *работами.* Узлы (обозначаемые кружками) являются *событиями.* События являются точно заданными моментами времени. Например, событие может представлять собой момент времени, когда в наличии имеются все детали, что позволяет начать сборку изделия. Сам процесс сборки, требующий определенного времени, представляет собой работу.

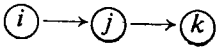
Направление дуги определяет соотношения предшествования. На отрезке сети



i-е событие должно произойти до начала работы A. Аналогично j-е событие не может произойти до завершения работы A.

Отношение предшествования является *транзитивным* между узлами. Если i-е событие предшествует j-му событию, а j-е со-

бытие предшествует k-му событию, то i-е событие предшествует k-му событию:



Иногда отношение предшествования между работами нельзя представить точно с помощью обычной структуры работ и событий. Допустим, например, что сетевая модель, показанная

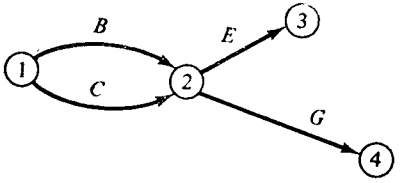


Рис. 4.2. Неправильное представление отношения предшествования.

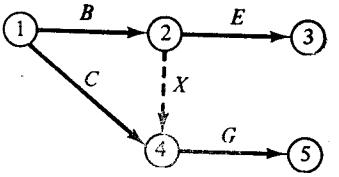


Рис. 4.3. Сетевая модель с фиктивной работой.

на рис. 4.2, предназначена для изображения такой последовательности: [работа G следует за работами B и C] и [работа E следует за работой B (но не за работой C)]. Такая схема является неправильной, поскольку она показывает, что как работа G, так и работа E следуют за работами C и B. Чтобы получить правильное представление, необходимо ввести фиктивную работу, продолжительность которой равна нулю. Обычно фиктивная работа обозначается штрихпунктиром. Правильное представление дано на рис. 4.3, где фиктивная работа обозначается через X. При необходимости фиктивные работы могут использоваться для изображения соотношений, которые невозможно представить другим способом. Это просто прием, позволяющий показать требуемое соотношение без изменения фактической продолжительности проекта. В литературе [26, 29] можно найти дальнейшие указания о построении сетей. Для иллюстрации построения сетевой модели рассмотрим следующий пример.

4.4.1. ПРОИЗВОДСТВЕННАЯ ЗАДАЧА

Рассмотрим построение сетевой модели, изображающей соотношение между работами, выполняемыми при сборке крупного станка, в котором узлы 1 и 2 объединяются в узел 4, а соединение узлов 3 и 4 дает готовое изделие. Вследствие необходимости согласовать некоторые детали узла 3 с соответствующими деталями узла 2 нельзя собрать узел 3 раньше, чем будут иметься в наличии детали для узла 2. Основные работы, кото-

Таблица 4.1. **Пример: изготовление крупного станка**

Работа		Продолжительность, сут	Непосредственно предшествующая работа
Обозначение	Описание		
A	Закупка деталей для узла 1	5	Нет
B	Закупка деталей для узла 2	3	Нет
C	Закупка деталей для узла 3	10	Нет
D	Изготовление узла 1	7	*A*
E	Изготовление узла 2	10	*B*
F	Изготовление узла 4	5	*D* и *E*
G	Изготовление узла 3	9	*B* и *C*
H	Окончательная сборка	4	*F* и *G*
I	Окончательная проверка и испытания	2	*H*

рые необходимо выполнить для изготовления станка, показаны в табл. 4.1, а сетевая модель — на рис. 4.4. Заметим, что имеется одно *начальное* событие и одно *завершающее* событие. За исключением этих событий, все остальные события имеют хотя бы

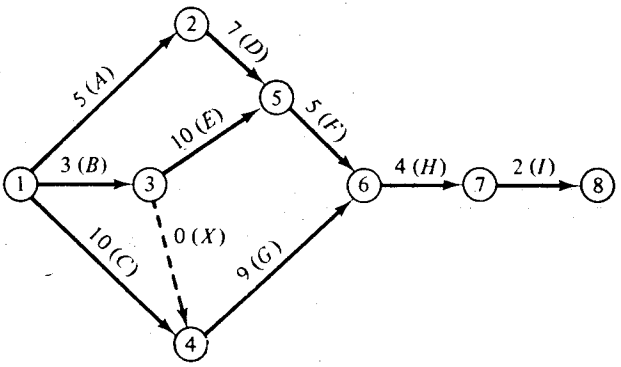


Рис. 4.4. Пример сети в виде модели узел-событие.

одну работу, ведущую к нему, и хотя бы одну, выходящую из него. Каждая работа обеспечивает однозначную связь между двумя узлами, что дает возможность определить работу через соединяемые ею события. Узловые события имеют последовательную нумерацию, поэтому все работы идут от узлов с меньшими номерами к узлам с бóльшими номерами.

Конечной целью сетевой модели является получение информации о плановых сроках выполнения отдельных работ. Для этого необходимо вначале определить наиболее ранний возможный и наиболее поздний допустимый сроки появления каждого события.

4.5. НАИБОЛЕЕ РАННИЙ ВОЗМОЖНЫЙ СРОК ПОЯВЛЕНИЯ СОБЫТИЯ

Напомним, что *событие* соответствует некоторому *узлу* и представляет собой момент начала одной или большего числа работ. Пусть $T_j(E)$ — наиболее ранний возможный срок наступления j-го события, $j=1, 2, ..., n$, где n — число событий (узлов) в сети. Заметим, что $T_j(E)$ — наиболее ранний возможный срок завершения *всех* работ, подходящих к j-му узлу. Пусть d_{ij} — продолжительность работы, соединяющей два события (узлы) — i-е и j-е.

Поскольку j-е событие не может произойти, пока не будут завершены все работы, ведущие к j-му узлу, и поскольку работа не может начаться, пока не произойдет предшествующее ей событие, наиболее ранний возможный срок наступления каждого события вычисляется как продолжительность *самого длинного* пути от начального до данного события.

Допустим, что от начального события (события 1) к j-му событию ведут r путей. Обозначим эти пути $\Pi_1, \Pi_2, ..., \Pi_r$. Каждому пути соответствует некоторая мера, равная сумме продолжительностей всех работ на данном пути. Следовательно,

$$T_j(\Pi_k)=\sum_m \sum_p d_{mp}, \quad m, p\in\Pi_k, \quad k=1, 2,..., r, \quad j=1, 2,..., n.$$

Таким образом, самый длинный путь от начального узла (узел 1) до j-го узла определяется как

$$T_j(E)=\max_k [T_j(\Pi_k)], \quad j=1, 2,..., n,$$

где максимум берется по всем путям, соединяющим узлы 1 и j.

Удобно принять $T_1(E)=0$ (самый длинный путь к первому узлу равен нулю). Рассмотрим теперь *все* работы, ведущие к последующему событию. Вычислим для каждой такой работы время, равное наиболее раннему возможному сроку наступления предыдущего события плюс продолжительность работы. Поскольку последующее событие не может появиться до завершения всех предшествующих работ, наиболее ранний возможный срок наступления рассматриваемого события равен максимуму этих двух различных промежутков времени. Иными словами, самый ранний возможный срок наступления j-го события определяется как

$$T_j(E)=\begin{cases} 0, & \text{если } j=1 \text{ (начальное событие)},\\ \max\limits_{i<j} [T_i(E)+d_{ij}], & 2\leqslant j\leqslant n, \end{cases}$$

где максимум берется по всем работам, завершающимся в j-м узле и выходящим из любого предшествующего i-го узла. Для

сети с n событиями, $j=1, 2, \ldots, n$, вычисления продолжаются до тех пор, пока не будет определен наиболее ранний возможный срок наступления завершающего n-го события.

В данном примере

$$T_1(E)=0,$$
$$T_2(E)=T_1(E)+d_{12}=0+5=5,$$
$$T_3(E)=T_1(E)+d_{13}=0+3=3,$$
$$T_4(E)=\max\begin{Bmatrix}T_3(E)+d_{34}=3+0=3\\T_1(E)+d_{14}=0+10=10\end{Bmatrix}=10,$$
$$T_5(E)=13,$$
$$T_6(E)=19,$$
$$T_7(E)=23,$$
$$T_8(E)=25.$$

4.6. НАИБОЛЕЕ ПОЗДНИЙ ДОПУСТИМЫЙ СРОК НАСТУПЛЕНИЯ КАЖДОГО СОБЫТИЯ

Пусть $T_i(L)$ — наиболее поздний срок наступления i-го события, не влияющий на время завершения всего проекта (наиболее поздний срок завершения всех работ, идущих к i-му узлу). Начиная с n-го события, движемся в *обратном направлении* через каждое предшествующее событие. Чтобы гарантировать, что продолжительность критического (самого длинного) пути не будет превышена, необходимо начинать процедуру с приравнивания наиболее позднего допустимого срока наступления завершающего события наиболее раннему возможному сроку завершения проекта. Следовательно, $T_n(L)=T_n(E)$.

Наиболее поздний допустимый срок наступления любого i-го события, непосредственно предшествующего n-му событию, определяется как

$$T_i(L)=T_n(L)-\min[d_{in}], \qquad i\leqslant n-1.$$

Для вычисления наиболее позднего срока наступления любого i-го события ($i<n$) рассмотрим все работы, идущие от этого события. Вычислим для каждой такой работы наиболее поздний срок наступления всех последующих событий и вычтем продолжительность работы. Наименьшее значение является наиболее поздним сроком наступления i-го события

$$T_i(L)=\begin{cases}T_n(E), & i=n,\\ \min\limits_{j>i}[T_j(L)-d_{ij}], & 1\leqslant i\leqslant n-1.\end{cases}$$

Минимум берется по всем j-м событиям, соединенным с i-м событием работой (i, j). Вычисления ведутся до тех пор, пока не будет определен наиболее поздний срок наступления начального события (событие 1).

Возвращаясь к предыдущему примеру, выполняем следующие вычисления:

$$T_8(L) = T_8(E) = 25,$$
$$T_7(L) = T_8(L) - d_{78} = 25 - 2 = 23,$$
$$T_6(L) = T_7(L) - d_{67} = 23 - 4 = 19,$$
$$T_5(L) = T_6(L) - d_{56} = 19 - 5 = 14,$$
$$T_4(L) = T_6(L) - d_{46} = 19 - 9 = 10,$$
$$T_3(L) = \min \begin{Bmatrix} T_5(L) - d_{35} = 14 - 10 = 4 \\ T_4(L) - d_{34} = 10 - 0 = 10 \end{Bmatrix} = 4,$$
$$T_2(L) = 7,$$
$$T_1(L) = 0.$$

4.7. РЕЗЕРВ ВРЕМЕНИ И КРИТИЧЕСКИЙ ПУТЬ

Максимальное время, на которое можно задержать наступление некоторого события без соответствующей задержки срока завершения всего проекта, называется резервом времени для данного события. Пусть S_i — резерв времени для i-го события. Тогда $S_i = T_i(L) - T_i(E)$. Если наиболее поздний допустимый и наиболее ранний возможный сроки наступления события одинаковы ($S_i = 0$), то задержка наступления события не допускается. События с нулевым резервом времени находятся на *критическом пути*. Работы, соединяющие события, находящиеся на критическом пути, называются *критическими*. Обозначим множество критических работ через SC. В рассматриваемом примере события 1, 4, 6, 7 и 8 находятся на критическом пути. Следовательно, $T_n(L) = T_n(E) = \sum_i \sum_j d_{ij} = 25$, $(i, j) \in SC$.

Критический путь представляет собой взаимосвязанную последовательность работ и событий — от начального до завершающего. Задержка наступления любого события приведет к задержке срока завершения всего проекта. Напротив, работы и события, не находящиеся на критическом пути, могут быть задержаны на ненулевой промежуток времени без влияния на срок завершения всего проекта. Например, из рис. 4.4 видно, что задержка события 2 на 7—5=2 единицы времени относительно наиболее раннего возможного срока его наступления не повлияет на срок завершения всего проекта.

С точки зрения руководителя, при управлении проектом основное внимание должно уделяться работам, находящимся на критическом пути. Если требуется сократить срок завершения всего проекта, то необходимо приложить усилия к сокращению продолжительности работ, находящихся на критическом пути. Процедуры оптимального сокращения продолжительности работ рассматриваются в разд. 4.12.

Следует заметить, что при сокращении продолжительности работы критический путь может измениться. Например, читатель может убедиться, что если продолжительность работ C или G сокращается более чем на 1 единицу времени, то критический путь пройдет через события 1, 3, 5, 6, 7, 8.

В дальнейшем необходимо проводить различие между *управлением сроками* и *сокращением* продолжительности проекта. Сокращение продолжительности проекта достигается за счет выделения большего количества ресурсов для отдельных работ, находящихся на критическом пути, что приведет к уменьшению продолжительности работ. Управление сроками представляет собой регулирование времени начала и завершения работ с целью сбалансирования имеющихся ресурсов. Процедура сокращения продолжительности работ будет изложена в разд. 4.12. На данном этапе рассматривается управление сроками. Как указывалось ранее, управление сроками осуществляется путем регулирования моментов начала и окончания работ, а не их продолжительности, так как только после выполнения всех работ (достижения всех целей) будет завершен весь проект. Поэтому желательно иметь информацию о планировании сроков конкретных работ в виде наиболее ранних и наиболее поздних сроков начала и окончания каждой работы. Как отмечалось выше, критический путь проходит через те работы, для которых наиболее ранний возможный и наиболее поздний допустимый сроки начала (или наиболее поздний допустимый срок окончания) одинаковы, т. е. задержки не допускаются.

4.8. СОСТАВЛЕНИЕ ТАБЛИЦ НАИБОЛЕЕ РАННИХ ВОЗМОЖНЫХ И НАИБОЛЕЕ ПОЗДНИХ ДОПУСТИМЫХ СРОКОВ ВЫПОЛНЕНИЯ РАБОТ

Наиболее ранний возможный срок начала работы представляет собой самое раннее время начала работы при допущении, что все предшествующие работы будут завершены как можно раньше. Пусть ES_{ij} — наиболее ранний возможный срок начала работы (i, j). Поскольку *работа* не может начинаться раньше наступления *предшествующего события*, имеем $ES_{ij}=T_i(E)$, где индекс i обозначает событие, предшествующее работе (i, j).

Отсюда следует, что

$$EF_{ij} = T_i(E) + d_{ij} = ES_{ij} + d_{ij},$$

где EF_{ij} — наиболее ранний возможный срок окончания работы (i, j).

Наиболее поздний допустимый срок окончания работы представляет собой самое позднее время завершения работы без задержки срока окончания всего проекта. Пусть LF_{ij} — наиболее поздний допустимый срок окончания работы (i, j). Поскольку работа может быть закончена не позднее наибольшего допустимого срока наступления последующего события j, имеем $LF_{ij} = T_j(L)$.

Наиболее поздний допустимый срок начала работы (i, j) можно вычислить следующим образом: $LS_{ij} = LF_{ij} - d_{ij}$.

В табл. 4.2 приводятся результаты вычислений для каждой работы в рассматриваемом примере. Критический путь состоит из работ, обозначаемых звездочкой (*). Заметим, что эти работы имеют нулевой резерв времени.

Таблица 4.2. **Наиболее ранние и наиболее поздние сроки начала и окончания работ: метод узел-событие**

Работа	Предшествующее событие, i	Последующее событие, j	Продолжительность, d_{ij}	Наиб. ранний возм. срок Начала, ES_{ij}	Наиб. ранний возм. срок Окончания, EF_{ij}	Наиб. поздний допуст. срок Начала, LS_{ij}	Наиб. поздний допуст. срок Окончания, LF_{ij}	Резерв времени	Критический путь
A	1	2	5	0	5	2	7	2	
B	1	3	3	0	3	1	4	1	
C	1	4	10	0	10	0	10	0	*
D	2	5	7	5	12	7	14	2	
X	3	4	0	3	3	10	10	7	
E	3	5	10	3	13	4	14	1	
G	4	6	9	10	19	10	19	0	*
F	5	6	5	13	18	14	19	1	
H	6	7	4	19	23	19	23	0	*
I	7	8	2	23	25	23	25	0	*

Для целей планирования сроков выполнения различных работ важно иметь некоторые другие показатели степени свободы задания сроков при разнообразных условиях. Эти показатели характеризуют имеющийся *резерв* времени.

4.9. ЧЕТЫРЕ ПОКАЗАТЕЛЯ РЕЗЕРВА ВРЕМЕНИ ПРИ ПЛАНИРОВАНИИ МЕТОДОМ КРИТИЧЕСКОГО ПУТИ [45]

Резерв времени является показателем гибкости планирования сроков в сетевой модели. Он существенно повышает полезность наиболее ранних возможных и наиболее поздних допусти-

мых сроков начала и окончания отдельных работ. Однако во многих случаях показатели резерва времени изучаются недостаточно хорошо и используются неудовлетворительно.

Можно определить четыре показателя: *суммарный, свободный, независимый* и *гарантированный резервы времени.* Показатели резерва времени находят ряд важных применений. Первым и наиболее распространенным из них является определение критического пути. Второй, более важной ролью показателей резерва времени является планирование фактических сроков выполнения работ, не относящихся к критическому пути. При планировании сроков работ с помощью МКП, предполагается отсутствие ограничений на ресурсы, необходимые для осуществления проекта. Однако в действительности работы выполняются персоналом, имеющим определенную квалификацию и использующим специальное оборудование. Необходимо распределять работы по определенным периодам времени, чтобы сбалансировать спрос на различные ресурсы. В связи с этим возникает задача распределения ресурсов, рассматриваемая в части II данной главы.

Возможно также, что организационное подразделение, которому поручен определенный вид работ (отдел, сектор, торговая точка и т. д.), несет также ответственность за выполнение работ, составляющих часть других проектов. Показатели резерва времени содержат информацию, необходимую для планирования сроков работ.

4.9.1. ПРОЦЕДУРА ВЫЧИСЛЕНИЙ

Вычисление каждого показателя резерва времени и его смысл лучше всего проиллюстрировать на численном примере. Рассматриваемый случай взят из статьи Томаса [45] и относится к работе установки на химическом заводе. Реактор и накопительный резервуар соединены герметичным трубопроводом диаметром 76 мм. Вследствие эрозии материала требуется периодическая замена трубопровода.

Клапаны, расположенные в трубопроводе и на его концах, также должны заменяться. Задача состоит в составлении основного календарного плана периодического технического обслуживания. Кроме того, необходимо заказать трубопровод и клапаны. Точные чертежи отсутствуют, и их нужно изготовить. Линия расположена в верхней части установки, поэтому для операции замены нужно возвести подмости; необходимая рабочая сила имеется. Предполагается, что для минимизации времени простоя установки работы должны вестись круглосуточно. Линию нельзя остановить без предварительного уведомления за

Таблица 4.3. Задача о ремонте трубопровода

	Работа	Непосредственно предшествующая работа	Продолжительность, сут
Q	Подготовительные операции		10
R	Подготовка линии к отключению		30
A	Измерение и изготовление чертежей	*Q*	2
B	Составление перечня материалов	*A*	1
C	Закупка трубопровода	*B*	30
D	Закупка клапанов	*B*	45
E	Изготовление секций	*C*	5
F	Отключение линии	*R, B*	1
G	Сооружение подмостей	*B*	2
H	Снятие старого трубопровода и клапанов	*F, G*	6
I	Установка нового трубопровода	*H, E*	6
J	Сварка трубопровода	*I*	2
K	Установка клапанов	*D, F, G*	1
L	Подгонка трубопровода и клапанов	*K, J*	1
M	Проверка на герметичность	*L*	1
N	Нанесение изоляции	*K, J*	4
O	Разборка подмостей	*L, N*	1
P	Уборка	*M, O*	1

30 сут. Кроме того, подготовка к инженерным работам займет 10 сут.

Инженер, управляющий работой установки, потребовал, чтобы старший инспектор по техническому обслуживанию и строительству совместно с эксплуатирующими подразделениями составил план и график осмотра установки. Сектор технологиче-

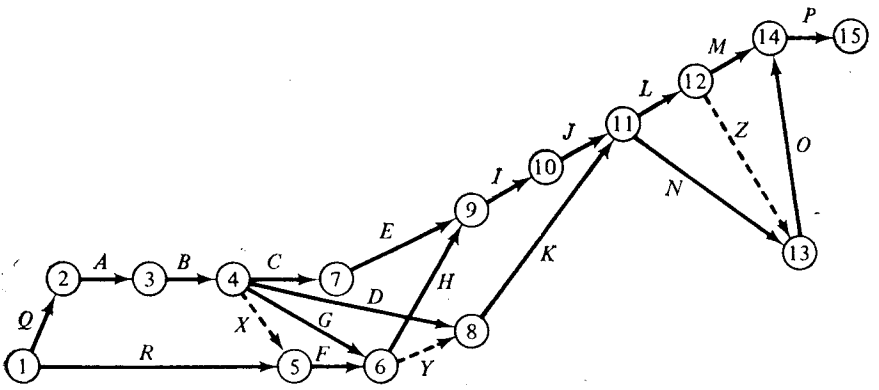


Рис. 4.5. Сетевая модель проекта.

Таблица 4.4. Множество непосредственно предшествующих и непосредственно следующих работ

Работа, i	Множество предшествующих работ	Множество последующих работ
Начало	Нет	Q, R
Q	Начало	A
R	Начало	F
A	Q	B
B	A	C, D, F, G
C	B	E
D	B	K
E	C	I
F	B, R	H, K
G	B	H, K
H	F, G	I
I	E, H	J
J	I	L, N
K	D, F, G	L, N
L	J, K	M, O
M	L	P
N	J, K	O
O	L, N	P
P	M, O	Окончание
Окончание	P	Нет

ских процессов и стандартов представил данные, показанные в табл. 4.3, для различных видов работ, составляющих проект. В табл. 4.4 дается сводка предшествующих и последующих событий для каждой операции. С помощью данных из табл. 4.3 и 4.4 и результатов, полученных в предыдущем разделе, были получены рис. 4.5, табл. 4.5 и 4.6.

4.9.2. ВЫЧИСЛЕНИЕ РЕЗЕРВА ВРЕМЕНИ

Суммарный резерв времени TF_{ij} для работы (i, j) представляет собой максимальную продолжительность задержки работы (i, j), не вызывающую задержки в осуществлении всего проекта. Он вычисляется как

$$TF_{ij} = LS_{ij} - ES_{ij}, \quad \text{или} \quad TF_{ij} = LF_{ij} - EF_{ij}.$$

Таблица 4.5. Сроки наступления событий

Событие, i	Наиб. ранний возм. срок $T_i(E)$	Наиб поздний допуст срок $T_i(L)$
1	0	0
2	10	10
3	12	12
4	13	13
5	30	44
6	31	45
7	43	46
8	58	58
9	48	51
10	54	57
11	59	59
12	60	63
13	63	63
14	64	64
15	65	65

Работа, имеющая нулевой суммарный резерв времени, находится на критическом пути. Следовательно, определение *суммарного резерва* времени идентично введенному ранее определению резерва. Суммарный резерв времени для каждой работы, рассматриваемой в данном примере, показан в табл. 4.7. Кратко описаны также и другие показатели резерва времени, приведенные в табл. 4.7. Работы *Q, A, B, D, K, N, O* и *P* находятся на критическом пути.

Очень часто не учитывается тот факт, что при вычислении суммарного резерва времени принимается неявное допущение, согласно которому все предшествующие работы (во всяком случае, те, которые имеют какое-либо отношение к рассматриваемой работе) должны выполняться как можно раньше, чтобы обеспечить суммарный резерв времени для одной работы. Следовательно, в общем случае практически невозможно для каждой работы реализовать собственный суммарный резерв времени, так как суммарный резерв времени для одной работы тесно связан с суммарным резервом других работ.

В рассматриваемом примере работа *C* может иметь суммарный резерв времени, равный 3 единицам, только за счет резерва времени для работ *E, I* и *J*.

Таблица 4.6. Сроки начала и окончания работ, модель узел-событие

Работа	Предшествующее событие i	Последующее событие j	Продолжительность d_{ij}	Наиб. ранний возм. срок Начала $ES_{ij}=T_i(E)$	Наиб. ранний возм. срок Окончания EF_{ij}	Наиб. поздний допуст. срок Начала LS_{ij}	Наиб. поздний допуст. срок Окончания $LF_{ij}=T_j(L)$	Резерв времени
Q	1	2	10	0	10	0	10	0
R	1	5	30	0	30	14	44	14
A	2	3	2	10	12	10	12	0
B	3	4	1	12	13	12	13	0
X	4	5	0	13	13	44	44	31
C	4	7	30	13	43	16	46	3
D	4	8	45	13	58	13	58	0
E	7	9	5	43	48	46	51	3
F	5	6	1	30	31	44	45	14
G	4	6	2	13	15	43	45	30
Y	6	8	0	31	31	58	58	27
H	6	9	6	31	37	45	51	14
I	9	10	6	48	54	51	57	3
J	10	11	2	54	56	57	59	3
K	8	11	1	58	59	58	59	0
L	11	12	1	59	60	62	63	3
M	12	14	1	60	61	63	64	3
Z	12	13	0	60	60	63	63	3
N	11	13	4	59	63	59	63	0
O	13	14	1	63	64	63	64	0
P	14	15	1	64	65	64	65	0

Следовательно, хотя суммарный резерв времени имеет важное значение для определения критического пути и используется при планировании срока завершения всего проекта, его ценность сомнительна при планировании сроков отдельных работ.

4.9.3. СВОБОДНЫЙ РЕЗЕРВ ВРЕМЕНИ

Свободный резерв времени является показателем максимальной задержки работы (i, j), не влияющей на начало последующих работ. Как и в предыдущем случае, предполагается, что все предшествующие работы завершаются как можно раньше. Свободный резерв времени FF_{ij} отличается от суммарного в том отношении, что он измеряет имеющееся время, не влияющее на задержку последующих работ. Он определяется как $FF_{ij}=T_j(E)-EF_{ij}$. Свободный резерв времени для определенной работы не может превышать суммарный резерв.

Таблица 4.7. Резервы времени

Работа, i	Общий резерв времени, TF_i	Свободный резерв времени, FF_i	Независимый резерв времени, IF_i	Гарантированный резерв времени, SF_i
Q	0	0	0	0
R	14	0	0	14
A	0	0	0	0
B	0	0	0	0
C	3	0	0	3
D	0	0	0	0
E	3	0	0	0
F	14	0	0	0
G	30	1	16	30
H	14	11	0	0
I	3	0	0	0
J	3	3	0	0
K	0	0	0	0
L	3	0	0	3
M	3	3	0	0
N	0	0	0	0
O	0	0	0	0
P	0	0	0	0

Для рассматриваемого примера свободный резерв времени показан в табл. 4.7. Хотя работа *E* имеет суммарный резерв времени, равный 3 единицам, она не имеет свободного резерва, так как ее задержка приведет к задержке работы *I*. Работа *H* имеет суммарный резерв времени, равный 14 единицам, но ее свободный резерв составляет всего 11 единиц. Любая задержка, превышающая 11 единиц, вызовет задержку срока наиболее раннего возможного начала работы *I*.

4.9.4. НЕЗАВИСИМЫЙ РЕЗЕРВ ВРЕМЕНИ

Плановые сроки выполнения определенной работы могут серьезно нарушаться, если предшествующие работы не будут закончены как можно раньше, что было показано при рассмотрении суммарного и свободного резервов времени. *Независимый резерв времени* является удобным показателем свободы планирования сроков. Независимый резерв IF_{ij} работы (i, j) представляет собой максимальную продолжительность задержки работы (i, j) без задержки последующих работ, если все предшествующие работы заканчиваются как можно позже. Это показатель имеющегося времени, если при выполнении предшествующих работ возникнут наихудшие из возможных условий. Он является также показателем возможной степени нарушения

связи между работами проекта, отсюда и название — независимый резерв.

Независимый резерв времени определяется по формуле

$$IF_{ij} = \max \begin{cases} 0, \\ T_j(E) - [T_i(L) + d_{ij}]. \end{cases}$$

Нулевой член вводится для того, чтобы приравнять нулю отрицательные результаты.

В рассматриваемом примере только работа G имеет ненулевой независимый резерв времени. Все остальные работы могут полностью потерять свой резерв времени вследствие запаздывания предшествующих работ.

4.9.5. ГАРАНТИРОВАННЫЙ РЕЗЕРВ ВРЕМЕНИ

SF_{ij} — максимальная возможная задержка работы, не влияющая на окончательный срок завершения проекта, если предшествующие работы выполняются с запаздыванием. Этот показатель резерва времени оказывается одним из наиболее удобных при планировании сроков выполнения определенной работы. Заметим, что он допускает задержку только последующих работ, а не всего проекта.

$$SF_{ij} = LF_{ij} - [T_i(L) + d_{ij}], \text{ или } SF_{ij} = T_j(L) - [T_i(L) + d_{ij}].$$

В данном примере только работы R, C, G и L имеют ненулевой гарантированный резерв. Из сетевого графика видно, что возможна задержка работы C вследствие задержки работ E и

Таблица 4.8. Систематизация четырех показателей резерва времени

Сроки завершения предшествующих работ	Сроки завершения последующих работ: Наиболее ранние	Сроки завершения последующих работ: Наиболее поздние
Наиболее ранние	Свободный резерв времени	Общий резерв времени
Наиболее поздние	Независимый резерв времени	Гарантированный резерв времени

Для работы (i, j)

Общий резерв: $LS_{ij} - ES_{ij}$ или $LF_{ij} - EF_{ij}$

Свободный резерв: $T_j(E) - EF_{ij}$

Независимый резерв: $\max \begin{cases} 0 \\ T_j(E) - [T_i(L) + d_{ij}] \end{cases}$

Гарантированный резерв: $LF_{ij} - [T_i(L) + d_{ij}]$

или $T_j(L) - [T_i(L) + d_{ij}]$

I, а задержка работы *G* возможна вследствие задержки работы *H*. Работы *H*, *E* и *I* лишаются гарантированного резерва вследствие возможной задержки предшествующих операций.

В табл. 4.8 представлены эти четыре показателя резерва времени в соответствии с наиболее ранним возможным или наиболее поздним допустимым сроками выполнения предшествующих или последующих работ.

Можно задать вопрос: «Какова ценность различных показателей резерва времени?» Первый очевидный ответ состоит в том, что нулевой суммарный резерв времени определяет работу, находящуюся на критическом пути, и необходимо тщательно следить за сроком ее выполнения. Все эти показатели резерва времени используются для точного определения сроков выполнения работ. Во многих случаях рабочая сила или оборудование, необходимые для выполнения определенной работы, должны распределяться с учетом потребности других работ в этих ресурсах. Различные показатели резерва времени помогают распределять имеющиеся ресурсы для каждой работы. При наличии резерва времени имеется некоторая свобода распределения ресурсов.

4.10. ФОРМУЛИРОВКА ЗАДАЧИ В ВИДЕ МОДЕЛИ УЗЕЛ-РАБОТА

Формулировка задачи в виде модели узел-работа обладает значительными преимуществами по сравнению с методом, при котором узел обозначал событие (разд. 2.1.2), вследствие чего такой подход может получить более широкое применение при планировании проектов с использованием ЭВМ.

4.10.1. ПОСТРОЕНИЕ СЕТИ

Основной особенностью рассматриваемого метода является то, что работы обозначаются *узлами*, а дуги только показывают отношения предшествования. Иначе говоря, время расходуется в узлах, а не на дугах (рис. 4.6). Работы *A* и *B* предшествуют работе *C*, которая в свою очередь предшествует работам *D* и *E*. Заметим, что реальный процесс перехода от узла *A* к узлу *E* требует определенного времени. Понятие события в данном случае не нужно, и оно не вводится. Все вычисления непосредственно связаны со сроками начала и окончания работ. Поскольку нет необходимости вводить события, задача построения сети значительно упрощается. Кроме того, как будет показано, нет

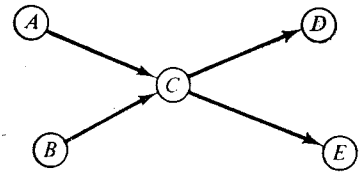


Рис. 4.6. Сеть в виде модели узел-работа.

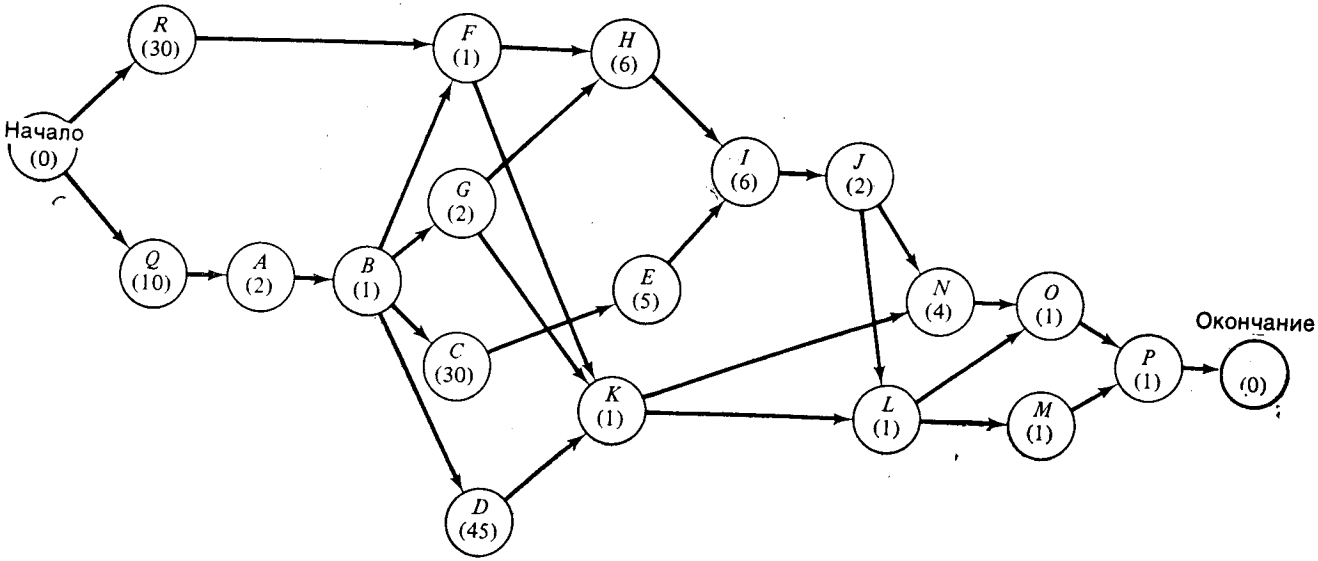


Рис. 4.7. Пример сети в виде модели узел-работа.

необходимости вводить в различные моменты времени фиктивные работы. Любое отношение предшествования можно точно представить и без них.

Построение сети облегчается за счет добавления лишь двух условных работ. Первая из них, обозначаемая «Начало», предшествует всем остальным работам, а вторая, называемая «Окончание», следует после всех работ. Каждая из них имеет нулевую продолжительность. На рис. 4.7 рассмотренный ранее пример представлен для случая, когда узел изображает работу. Заметим, что не только сеть имеет другой вид, но и правильное соотношение предшествования изображается без введения фиктивных работ. В каждом узле число в круглых скобках показывает продолжительность работы.

4.10.2. ПРОЦЕДУРЫ ВЫЧИСЛЕНИЙ

Проект завершается, когда заканчивается выполнение работы P. На рис. 4.7 показана сетевая модель проекта, в который включены условные работы «Начало» и «Окончание» (каждая нулевой продолжительности), чтобы получить единый начальный и конечный узлы. В этой задаче работу «Окончание» можно было бы опустить, так как уже имеется одна последняя работа, но она вводится в рамках общей процедуры. Нет необходимости употреблять обозначения с двумя индексами, как в случае обозначения работы в виде дуги. Поскольку все работы однозначно связаны с каким-либо одним узлом, достаточно одного индекса. Далее при ссылке на i-ю работу подразумевается ссылка на i-й узел. Продолжительность i-й работы обозначается теперь d_i, $i=1, 2, \ldots, n$. Обычно продолжительность работы d_i указывается возле каждого узла.

Далее целесообразно определить *предшествующее* множество работ $\mathbf{P}_i$ как множество работ, непосредственно предшествующих i-й работе. Непосредственно *следующее* множество работ $\mathbf{S}_i$ определяется как множество работ, непосредственно следующих за i-й работой. Для большей точности обозначим символом $\ll$ отношение непосредственного предшествования. Запись вида $i \ll j$ означает, что i-я работа непосредственно предшествует j-й работе, или что то же самое, j-я работа следует непосредственно за i-й. Отсюда вытекает, что $\mathbf{P}_i=\{a \mid a \ll i\}$ и $\mathbf{S}_i=\{a \mid i \ll a\}$. Введем следующие обозначения: ES_i — наиболее ранний возможный срок начала i-й работы; EF_i — наиболее ранний возможный срок окончания i-й работы; LS_i — наиболее поздний допустимый срок начала i-й работы; LF_i — наиболее поздний допустимый срок окончания i-й работы. Полагаем, что для условной работы «Начало» $ES_{\text{начало}}=EF_{\text{начало}}=0$. Отсюда следует, что для всех последующих работ $ES_i=\max\limits_{X \in \mathbf{P}_i}[EF_X]$ и

$EF_i = ES_i + d_i$. Эти вычисления ведутся от более ранних к более поздним работам таким образом, чтобы каждая работа была рассмотрена до ее появления в непосредственно предшествующем множестве другой работы.

Для вычисления наиболее позднего допустимого срока окончания работ обозначим через T плановый срок окончания проекта. Отсюда следует, что для выполнения проекта в заданный срок необходимо, чтобы $T \geqslant EF_{\text{окончание}}$. Обычно принимается $T = EF_{\text{окончание}}$. Затем получаем $LF_{\text{окончание}} = LS_{\text{окончание}} = T$. Следовательно, для всех других работ

$$LF_i = \min_{X \in S_i} [LS_X] \quad \text{и} \quad LS_i = LF_i - d_i.$$

Как и в предыдущем методе, движемся в обратном направлении. Каждая работа рассматривается до ее появления в непосредственно следующем множестве другой работы. Результаты вычислений для рассмотренного примера показаны в табл. 4.9.

Читатель может проверить, что четыре показателя резерва времени для каждой работы можно вычислить с помощью следующих соотношений:

Таблица 4.9. Наиболее ранние и наиболее поздние сроки начала и окончания работ

Работа, i	Продолжительность, y_i	Наиб. ранний возм. срок		Наиб. поздний допуст. срок	
		Начала, ES_i	Окончания, EF_i	Начала, LS_i	Окончания, LF_i
Начало	0	0	0	0	0
Q	10	0	10	0	10
R	30	0	30	14	44
A	2	10	12	10	12
B	1	12	13	12	13
C	30	13	43	16	46
D	45	13	58	13	58
E	5	43	48	46	51
F	1	30	31	44	45
G	2	13	15	43	45
H	6	31	37	45	57
I	6	48	54	51	57
J	2	54	56	57	59
K	1	58	59	58	59
L	1	59	60	62	63
M	1	60	61	63	64
N	4	59	63	59	63
O	1	63	64	63	64
P	1	64	65	64	65
Окончание	0	65	65	65	65

Суммарный резерв времени: $TF_i=LS_i-ES_i$ или $TF_i=LF_i-EF_i$. Работы с нулевым резервом времени находятся на критическом пути.

Свободный резерв времени: $FF_i=\min\limits_{X\in S_i}[ES_X-EF_i]$:

Независимый резерв времени: $IF_i=$

$$=\max_{X\in S_i}\begin{cases}0,\\ \min ES_X\end{cases}-[\max_{Z\in P_i} LF_Z+d_i].$$

Гарантированный резерв времени: $SF_i=LF_i-\max\limits_{Z\in P_i}[LF_Z+d_i]$.

Применяя эти результаты к предыдущему примеру, с помощью табл. 4.7 можно получить все показатели резерва времени. Разумеется, что результаты тождественны полученным ранее и приведенным в табл. 4.6.

4.11. МЕТОД ОЦЕНКИ И ПЕРЕСМОТРА ПЛАНОВ (ПЕРТ)

В предыдущих разделах при управлении проектом не учитывались вероятностные соображения. В рассмотренной методике предполагалось, что продолжительность работы точно известна. При использовании ПЕРТ вводится неопределенность в продолжительность работы.

В случае ПЕРТ для каждой работы сетевой модели проекта принимаются три оценки продолжительности выполнения: 1) *наиболее вероятное время выполнения m*; 2) *оптимистическая оценка времени a* и 3) *пессимистическая оценка времени b*.

Наиболее вероятное время определяется как время выполнения работы при нормальных условиях. Оптимистическая и пессимистическая оценки задают размах колебаний продолжительности работы под влиянием неопределенности. Оптимистическая оценка показывает минимально необходимое время, когда все идет по плану, а пессимистическая оценка отражает максимальное время выполнения работы, необходимое при неблагоприятных условиях, например механических поломках, нехватке рабочей силы или материалов, перебоях в снабжении. Следует заметить, что пессимистическая оценка не учитывает необычные продолжительные задержки либо катастрофы. Поскольку обе эти оценки являются лишь приемлемыми предположениями, фактическая продолжительность работы может лежать за пределами этого интервала. (С вероятностной точки зрения можно сказать, что вероятность выхода продолжительности работы за пределы заданного интервала очень мала.)

В большинстве случаев в системе ПЕРТ принимается бета-распределение продолжительности работ, показанное на рис. 4.8, где μ обозначает среднюю продолжительность. Значе-

ние μ зависит от того, насколько близко находятся значения a и b к m.

Ожидаемая продолжительность работы приближенно определяется как $\mu=(a+4m+b)/6$. Поскольку фактическая продолжительность может отличаться от среднего значения, необходимо знать дисперсию продолжительности работы. У большинства унимодальных распределений (т. е. распределений с одним максимумом) крайние значения отстоят на три среднеквадратических отклонения от среднего значения. Таким образом, размах распределения равен шести среднеквадратическим отклонениям (σ).

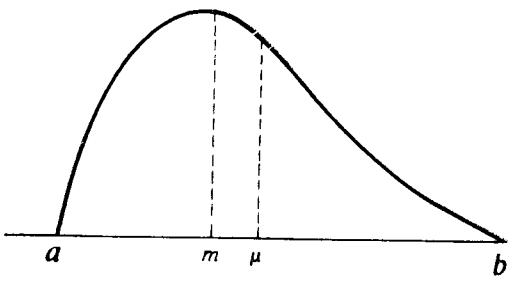


Рис. 4.8. Бета-распределение продолжительности работ.

Итак, $6\sigma=b-a$, или $\sigma=(b-a)/6$. Дисперсия продолжительности работы равна $\sigma^2=[(b-a)/6]^2$.

В системе ПЕРТ с помощью трех оценок продолжительности всех работ по этим формулам для μ и σ^2 вычисляется средняя продолжительность и ее дисперсия для каждой работы. Рассматривая среднее значение как фактическую продолжительность работы, можно найти критический путь. Продолжительность проекта T определяется как сумма продолжительностей всех работ, находящихся на критическом пути. Однако продолжительности работ являются случайными величинами. Следовательно, продолжительность проекта T также является случайной величиной, и можно говорить о средней продолжительности проекта и ее дисперсии.

Ожидаемая продолжительность проекта равна сумме всех средних продолжительностей работ, находящихся на критическом пути. Аналогично дисперсия продолжительности проекта равна сумме всех дисперсий продолжительностей работ, находящихся на критическом пути, при допущении, что продолжительности всех работ независимы.

4.11.1. ПРИМЕР СИСТЕМЫ ПЕРТ

Рассмотрим проект, состоящий из девяти работ (A, B, ..., I), с отношениями предшествования и оценками продолжительно-

Таблица 4.10. Отношения предшествования и оценки продолжительности

Работа	Предшествующие работы	Оценка продолжительности: Оптимистическая, a	Наиболее вероятная, m	Пессимистическая, b
A	—	2	5	8
B	A	6	9	12
C	A	6	7	8
D	B, C	1	4	7
E	A	8	8	8
F	D, E	5	14	17
G	C	3	12	21
H	F, G	3	6	9
I	H	5	8	11

Таблица 4.11. Средние продолжительности работ и среднеквадратические отклонения

Работа	Средняя продолжительность	Среднеквадратическое отклонение	Дисперсия
A	5	1	1
B	9	1	1
C	7	$\frac{1}{3}$	$\frac{1}{9}$
D	4	1	1
E	8	0	0
F	13	2	4
G	12	3	9
H	6	1	1
I	8	1	1

сти, показанными в табл. 4.10. Вначале вычислим среднюю продолжительность и дисперсию для каждой работы. Полученные данные приведены в табл. 4.11.

На рис. 4.9 показана сетевая модель проекта с обозначением работ в виде дуг, где числа над дугами показывают среднюю продолжительность работ. С помощью средних продолжительностей работ вычисляются наиболее ранний возможный и наиболее поздний допустимый сроки наступления каждого события. Критический путь определяется как

$$① \rightarrow ② \rightarrow ④ \rightarrow ⑤ \rightarrow ⑥ \rightarrow ⑦ \rightarrow ⑧$$

Соответственно критическими работами являются A, B, D, F, H и I.

Обозначим через T продолжительность проекта. Тогда ожидаемая продолжительность проекта равна сумме ожидаемых

продолжительностей работ A, B, D, F, H и I: $E(T)=5+9+4+13+6+8=45$ сут. Дисперсия продолжительности проекта равна сумме дисперсий продолжительности работ A, B, D, F, H и I: $V(T)=1+1+1+4+1+1=9$. Среднеквадратическое отклонение продолжительности проекта равно $\sigma(T)=\sqrt{V(T)}=3$.

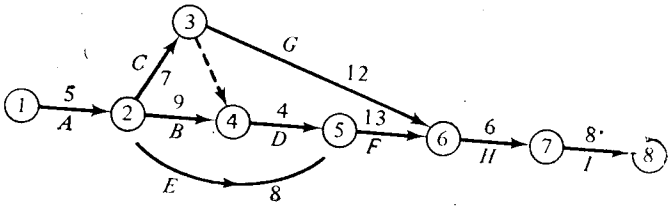


Рис. 4.9. Пример сетевой модели проекта.

4.11.2. ВЕРОЯТНОСТИ ЗАВЕРШЕНИЯ ПРОЕКТА

Продолжительность проекта T равна сумме продолжительностей всех работ, находящихся на критическом пути. В системе ПЕРТ предполагается, что продолжительности всех работ

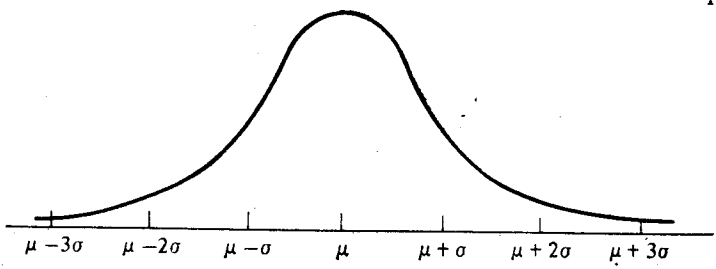


Рис. 4.10. Нормальное распределение с математическим ожиданием μ и среднеквадратическим отклонением σ.

независимы и распределены по одному закону. Следовательно, в силу центральной предельной теоремы, случайная величина T имеет нормальное распределение с математическим ожиданием $E(T)$ и дисперсией $V(T)$. На рис. 4.10 показано номальное распределение с математическим ожиданием μ и дисперсией σ^2.

В нашем примере продолжительность T имеет нормальное распределение с математическим ожиданием, равным 45, и среднеквадратическим отклонением, равным 3. В случае нормального распределения вероятность того, что значение случайной величины отличается от математического ожидания не более чем на одно среднеквадратическое отклонение, равна 0,68. Следовательно, с вероятностью 0,68 продолжительность проекта составит от 42 до 48 сут. Аналогично с вероятностью 0,997 продолжительность проекта T будет отличаться от среднего значе-

ния не более чем на три среднеквадратических отклонения (от 36 до 54 сут).

Можно также вычислить вероятность завершения проекта к определенному сроку. Например, руководителям нужно знать вероятность осуществления проекта за 50 дней. Иными словами, требуется вычислить $P(T \leqslant 50)$, где T — случайная величина, распределенная по нормальному закону с математическим ожиданием, равным 45, и дисперсией, равной 9. Эту вероятность можно найти с помощью таблицы для нормированного нормального распределения с нулевым математическим ожиданием и среднеквадратическим отклонением, равным 1.

Согласно теории вероятностей, случайная величина $Z=[T-E(T)]/\sigma(T)$ имеет нормальное распределение с нулевым математическим ожиданием и среднеквадратическим отклонением, равным 1. Следовательно,

$$P(T \leqslant 50) = P\left(Z \leqslant \frac{50-45}{3}\right) = P(Z \leqslant 1{,}67) = 0{,}95.$$

Таким образом, вероятность того, что проект будет закончен за 50 сут, составляет 0,95.

Допустим, что необходимо знать вероятность завершения проекта на 4 дня раньше, чем ожидается. Это означает, что требуется вычислить

$$P(T \leqslant 41) = P\left(Z \leqslant \frac{41-45}{3}\right) = P(Z \leqslant -1{,}33) = 0{,}09.$$

Следовательно, вероятность того, что проект будет закончен через 41 сут, составляет всего 0,09.

ЧАСТЬ II. РАСПРЕДЕЛЕНИЕ РЕСУРСОВ В СЕТЕВЫХ ГРАФИКАХ ПРОЕКТОВ

Результаты получены при участии
Эдварда Дэвиса,
Школа бизнеса «Колгейт Дарден» при Университете
шт. Вирджиния

Широкое распространение ПЕРТ и МКП в начале 60-х годов вызвало большой интерес к сетевым методам управления проектами. Последующий рост популярности этих методов показал, что сетевые модели являются полезным средством для формулирования широкого круга задач планирования и определения сроков выполнения работ. Однако наряду с этим отмечено, что основные методы — ПЕРТ и МКП — являются несколько упрощенными моделями большинства реальных ситуаций, так как

в них основной упор делается на *сроки*, но не учитывается потребность в *ресурсах* и их наличие.

Вследствие этого в последние годы возрастает внимание к задачам распределения ресурсов, связанным с планированием и заданием сроков осуществления проектов. Многие новые результаты, полученные при разработке сетевых методов, относятся именно к этой области. Развитие сетевых методов достигло такого уровня, что люди, знакомые с ними поверхностно, часто не имеют четкого представления об их возможностях при решении задач распределения ресурсов. В данном разделе делается попытка внести некоторую ясность путем систематизации разработанных методов, а также рассмотрения возможностей и недостатков каждого из них.

4.12. СООТНОШЕНИЕ МЕЖДУ ВРЕМЕНЕМ И ЗАТРАТАМИ: РАСПРЕДЕЛЕНИЕ ДЕНЕЖНЫХ СРЕДСТВ

Нередко выполнение некоторых или даже всех работ проекта можно ускорить путем выделения большего количества ресурсов за счет увеличения прямых затрат на выполнение работы. В этих случаях существует много различных комбинаций продолжительностей работ, при которых может быть получена некоторая требуемая плановая продолжительность. Однако каждая комбинация может давать различные значения общей стоимости проекта. Процедуры выбора компромиссного соотношения между сроками и затратами имеют целью составление календарного плана, обеспечивающего минимальные затраты при данной продолжительности проекта.

Рассмотрим, к примеру, простой проект, состоящий из восьми работ, представленных в табл. 4.12. Каждая работа может

Таблица 4.12. Пример сети с данными о продолжительностях работ и затратах

	Нормальные сроки		Сжатые сроки		
Работа	Продолжительность, сут	Затраты, долл.	Продолжительность, сут	Затраты, долл.	Увеличение затрат, долл.
(0, 1)	4	210	3	280	70
(0, 2)	8	400	6	560	80
(1, 2)	6	500	4	600	50
(1, 4)	9	540	7	600	30
(2, 3)	4	500	1	1100	200
(2, 4)	5	150	4	240	90
(3, 5)	3	150	3	150	—[a]
(4, 5)	7	600	6	750	150
		3050		4280	

[a] Эту работу нельзя ускорить

выполняться за разное время — от верхнего «нормального» срока при некоторых соответствующих «нормальных» затратах до меньшего «сокращенного» срока при соответствующих более высоких затратах. Заметим, что если предполагается, что компромиссное соотношение между временем и затратами для каждой работы является линейным, то затраты при промежуточных продолжительностях работы, лежащих между нормальным и сокращенным сроками, легко определить с помощью единствен-

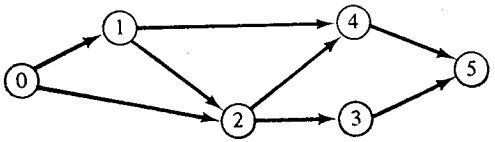


Рис. 4.11. Сетевая модель работ для задачи сокращения сроков за счет увеличения затрат.

ного значения приращения затрат для каждой работы [например, затраты на выполнение работы за 7 сут вместо 8 равны 400 долл.+80 долл.=480 долл.].

Если заданы «нормальные» продолжительности всех работ, то продолжительность проекта составляет 22 сут, что определяется с помощью критического пути, состоящего из работ (0, 1), (1, 2), (2, 4), (4, 5) на рис. 4.11. Результаты вычислений приводятся в табл. 4.13. Как показано на рис. 4.12, соответствующая стоимость выполнения проекта составляет 3050 долл. Заметим, что принятие неправильного решения, согласно которому ускоряется выполнение всех работ, не лежащих на критическом пути, не приводит к сокращению продолжительности проекта. Между этими верхним и нижним значениями затрат при продолжительности проекта 22 сут возможны несколько других значений в зависимости от числа некритических работ, срок выполнения которых сокращается.

Таблица 4.13. Анализ критического пути

Работа	Продолжительность	Наиб. ранний возм. срок		Наиб. поздний допуст. срок		Резерв времени	Критич. путь
		Начала	Окончания	Начала	Окончания		
(0, 1)	4	0	4	0	4	0	*
(0, 2)	8	0	8	2	10	2	
(1, 2)	6	4	10	4	10	0	*
(1, 4)	9	4	13	6	15	2	
(2, 3)	4	10	14	15	19	5	
(2, 4)	5	10	15	10	15	0	*
(3, 5)	3	14	17	19	22	5	
(4, 5)	7	15	22	15	22	0	*
					Общая продолжительность:		22

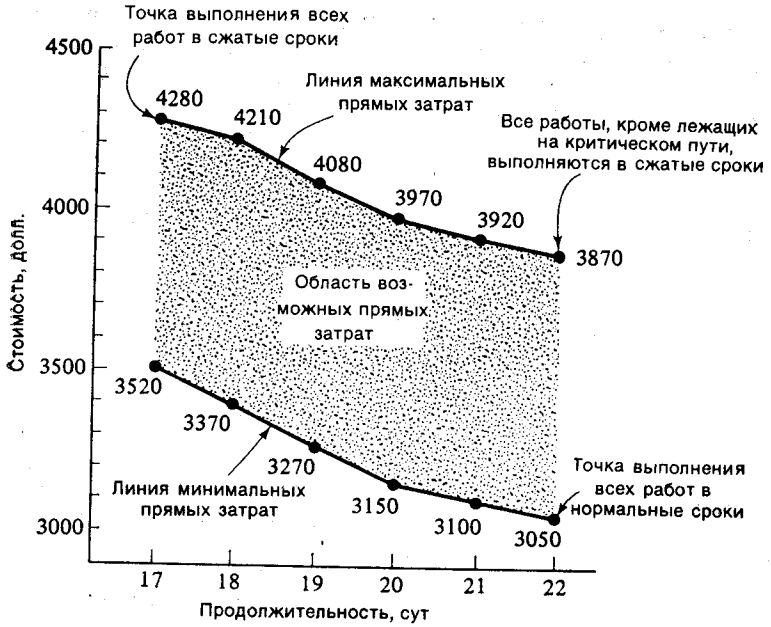


Рис. 4.12. Зависимость между продолжительностью проекта и прямыми затратами.

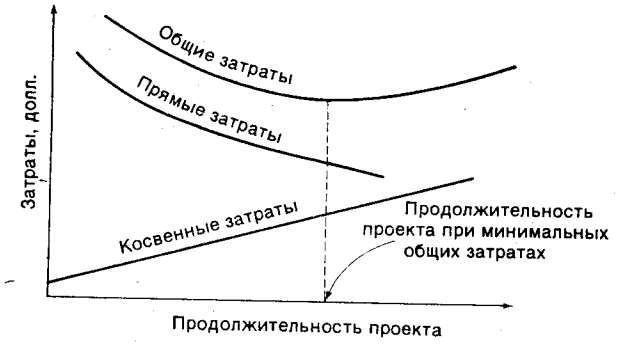


Рис. 4.13. Определение продолжительности проекта при минимальных общих затратах.

Если устанавливаются сокращенные сроки выполнения всех работ, то продолжительность осуществления проекта можно сократить до 17 сут, но, как следует из рис. 4.12 (верхняя левая точка), общие затраты при этом возрастают до 4280 долл. Однако продолжительность осуществления проекта, равную 17 сут, можно достигнуть при меньших затратах без ненужного уско-

рения отдельных работ. Так, работа (0, 2) может продолжаться не 6, а 7 единиц времени, работа (1, 4) — не 7, а 8 единиц времени, а работа (2, 3) — 4, а не 1 единицу времени. Если все остальные работы выполняются в ускоренном темпе, то стоимость выполнения проекта за 17 сут снижается до 3520 долл.

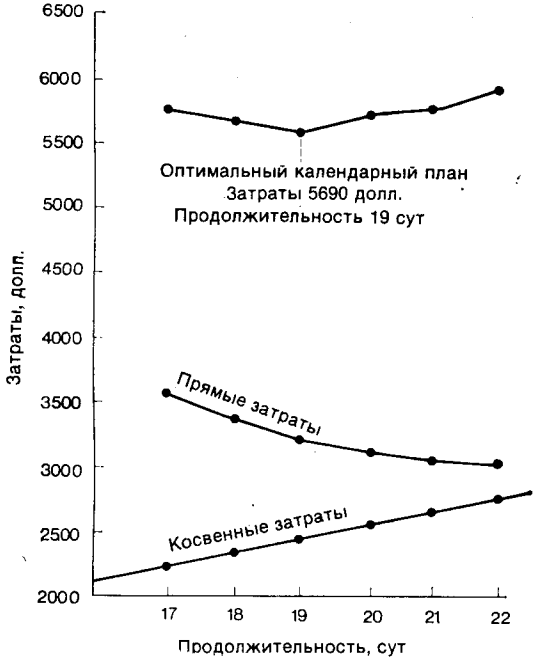


Рис. 4.14. Оптимальное сокращение сроков за счет прямых и косвенных затрат.

Как легко установить путем эксперимента, это самая низкая стоимость осуществления проекта за 17 сут.

Как показано на рис. 4.12, продолжительность проекта может иметь некоторое промежуточное значение, лежащее между 17 и 22 сут. Для каждой продолжительности проекта существует некоторый интервал возможных значений стоимости, зависящих от продолжительности отдельных работ и оттого, производится ли ненужное ускорение сроков их осуществления. На рис. 4.12 показаны кривые максимальных и минимальных затрат и область возможных затрат при каждой продолжительности проекта, лежащая между этими кривыми.

В этом простом примере кривую минимальных прямых затрат легко определить методом проб и ошибок. Однако в более реальных ситуациях, когда имеются десятки или даже сотни

Таблица 4.14. Изменение затрат при сокращении сроков работ

Работы, выполняемые в сжатые сроки	Продолжительность, сут	Прямые затраты, долл.	Косвенные затраты, долл.	Общие затраты, долл.
Критический путь	22	3050	2860	5910
(1, 2)	21	3100	2730	5830
(1, 2)	20	3150	2600	5750
(2, 4) и (1, 4)	19	3220	2470	5690 Оптимальный вариант
(4, 5)	18	3370	2340	5710
(0, 1) и (0, 2)	17	3520	2210	5730

работ, построение кривой методом проб и ошибок становится исключительно утомительным занятием, либо оно вообще невозможно. Поэтому были разработаны различные систематические методы вычислений, в том числе методы математического программирования, позволяющие быстро определить кривую минимальных затрат при любом возможном значении продолжительности проекта. Некоторые из этих стандартных методов предназначены для использования в тех случаях, когда компромиссные соотношения между временем и затратами являются нелинейными; многие из них позволяют получить кривую минимальных общих затрат (равных сумме прямых и косвенных затрат), что показано на рис. 4.13.

Чтобы проиллюстрировать влияние ускорения работ на общие затраты, рассмотрим снова предыдущий пример с постоянными косвенными затратами, равными 130 долл./сут. В данном случае кривая, изображенная на рис. 4.13, представлена на рис. 4.14. Результаты вычислений приводятся в табл. 4.14.

4.12.1. ПОТОКОВЫЙ АЛГОРИТМ, ИСПОЛЬЗУЮЩИЙ МЕТОД КРИТИЧЕСКОГО ПУТИ, В СЕТИ С ЗАВИСИМОСТЬЮ МЕЖДУ ВРЕМЕНЕМ И ЗАТРАТАМИ

Продолжительность любой работы проекта можно регулировать количеством ресурсов, выделяемых для выполнения работы. В общем случае можно предположить, что руководители проекта могут оценивать продолжительность работ как функцию суммы денежных средств, затраченных на каждую из них. Поэтому при таком допущении можно построить математическую модель, предназначенную для минимизации общей стоимости проекта. Модель позволяет найти оптимальные значения сроков наступления событий и продолжительностей работ при заданных продолжительности проекта, отношениях предшество-

вания, верхних и нижних пределах продолжительности каждой работы.

Для формулировки задачи линейного программирования и вывода процедуры решения принимаем следующие обозначения: y_{ij} — продолжительность работы (i, j); t_i — момент наступления i-го события; c_{ij} — стоимость работы (i, j) как функция продолжительности работы, т. е. $c_{ij} = f(y_{ij})$; L_{ij} — нижний предел продолжительности работы (i, j); U_{ij} — верхний предел продолжительности работы (i, j); T — заданная продолжительность проекта.

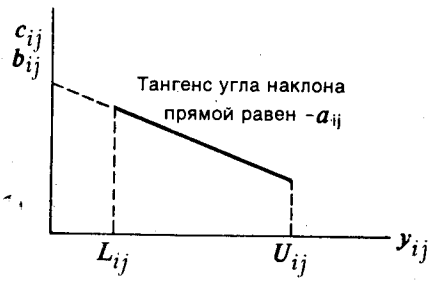


Рис. 4.15. Линейная зависимость между затратами и продолжительностью.

В рассматриваемом случае предполагается существование линейной зависимости продолжительности от затрат, т. е. $c_{ij} = f(y_{ij}) = b_{ij} - a_{ij}y_{ij}$, где $L_{ij} \leqslant y_{ij} \leqslant U_{ij}$. Это линейное соотношение показано на рис. 4.15. Нижний предел обычно соответствует более быстрому выполнению работы, а в качестве верхнего предела принимается «нормальная» продолжительность. При заданной продолжительности проекта и линейном соотношении между затратами и продолжительностью для проекта с представлением сети в виде $G = (\mathbf{N}, \mathbf{A})$ требуется знать, какие работы множества $\mathbf{A}$ необходимо ускорить, а для каких сохранить нормальную продолжительность.

Как показано ниже, рассматриваемую задачу можно сформулировать как задачу линейного программирования. Предполагается, что множество узлов $\mathbf{N}$ можно определить как $\mathbf{N} = \{1, 2, \ldots, n\}$, где узел 1 обозначает начало проекта, а узел n — окончание проекта.

$$\text{Минимизировать} \sum_{(i,\, j) \in \mathbf{A}} (b_{ij} - a_{ij}y_{ij}).$$

при условии, что

$$t_i - t_j + y_{ij} \leqslant 0 \quad \text{для всех } (i, j) \text{ в } \mathbf{A},$$

$$-t_1 + t_n = T$$

$$y_{ij} \leqslant U_{ij} \quad \text{для всех } (i, j) \text{ в } \mathbf{A},$$

$$y_{ij} \geqslant L_{ij} \quad \text{для всех } (i, j) \text{ в } \mathbf{A}.$$

Эта модель эквивалентна рассматриваемой ниже задаче линейного программирования с максимизацией функции цели.

Ограничения на неотрицательность t_i и y_{ij} не заданы в модели в явном виде, а это означает, что все ограничения соответствующей двойственной задачи линейного программирования должны иметь вид равенств.

$$\text{Максимизировать} \sum_{(i,\,j)\,\in A} a_{ij}y_{ij}$$

при условии, что

$$\begin{aligned} t_i - t_j + y_{ij} &\leqslant 0 \quad \text{для всех } (i,j) \text{ в } \mathbf{A}, \\ -t_1 + t_n &= T \\ y_{ij} &\leqslant U_{ij} \quad \text{для всех } (i,j) \text{ в } \mathbf{A}, \\ -y_{ij} &\leqslant -L_{ij} \quad \text{для всех } (i,j) \text{ в } \mathbf{A}. \end{aligned}$$

Пусть f_{ij}, v, γ_{ij} и δ_{ij} — двойственные переменные, соответствующие предыдущим линейным ограничениям. Двойственные переменные перечисляются в таком же порядке, в котором вводились ограничения в данную модель. Двойственную задачу можно сформулировать следующим образом.

$$\text{Минимизировать} \left\{ Tv + \sum_{i,j} U_{ij}\gamma_{ij} - \sum_{i,j} L_{ij}\delta_{ij} \right\}$$

при условии, что

$$\begin{aligned} f_{ij} + \gamma_{ij} - \delta_{ij} &= a_{ij} \quad \text{для всех } (i,j) \text{ в } \mathbf{A}, \\ \sum_j f_{1j} - v &= 0 \\ \sum_j [f_{ij} - f_{ji}] &= 0 \quad \text{для всех } i = 2, \ldots, n-1, \\ -\sum_j f_{jn} + v &= 0 \\ f_{ij},\ \gamma_{ij},\ \delta_{ij} &\geqslant 0 \quad \text{для всех } (i,j) \text{ в } \mathbf{A}. \end{aligned}$$

На основании математической структуры двойственной задачи двойственные переменные f_{ij} можно рассматривать как потоки в сети с ограниченной пропускной способностью. Вторая, третья и четвертая группы ограничений соответствуют ограничениям потока для источника, промежуточных и конечного узлов соответственно. В частности, третья группа ограничений соответствует известным ограничениям на сохранение потока.

Используя условия дополняющей нежесткости для задач линейного программирования, можно вывести следующие результаты, которые должны выполняться для оптимального решения:

$$t_i - t_j + y_{ij} < 0, \qquad f_{ij} = 0,$$

$$
\begin{aligned}
&y_{ij} < U_{ij}, \quad && \gamma_{ij} = 0,\\
&y_{ij} > L_{ij}, \quad && \delta_{ij} = 0,\\
&f_{ij} > 0, \quad && t_i - t_j + y_{ij} = 0,\\
&\gamma_{ij} > 0, \quad && y_{ij} = U_{ij},\\
&\delta_{ij} > 0, \quad && y_{ij} = L_{ij}.
\end{aligned}
$$

Последние два условия означают, что γ_{ij} и δ_{ij} не могут быть одновременно положительными при допущении, что $L_{ij} \neq U_{ij}$.

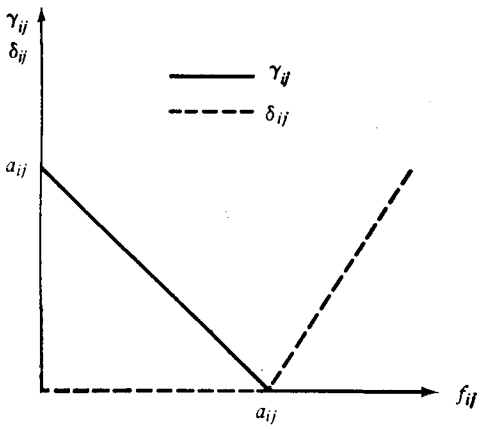


Рис. 4.16. Зависимость γ_{ij} и δ_{ij} от f_{ij}.

Поскольку $f_{ij} + \gamma_{ij} - \delta_{ij} = a_{ij}$, легко сделать вывод, что неотрицательные значения γ_{ij} и δ_{ij} определяются по формулам:

1. $\gamma_{ij} = a_{ij} - f_{ij}$ при $\delta_{ij} = 0$.
2. $\delta_{ij} = f_{ij} - a_{ij}$ при $\gamma_{ij} = 0$.

Поэтому можно записать:

1. $\gamma_{ij} = \max[0;\ a_{ij} - f_{ij}]$ при $\delta_{ij} = 0$.
2. $\delta_{ij} = \max[0;\ f_{ij} - a_{ij}]$ при $\gamma_{ij} = 0$.

Графическое представление γ_{ij} и δ_{ij} как линейных функций от f_{ij} показано на рис. 4.16.

При исследовании всех возможных значений γ_{ij}, δ_{ij} и f_{ij} можно выделить три случая.

Случай 1: $\gamma_{ij} > 0$ и $\delta_{ij} = 0$; $0 \leqslant f_{ij} < a_{ij}$ и $y_{ij} = U_{ij}$.

Случай 2: $\gamma_{ij} = 0$ и $\delta_{ij} = 0$; $f_{ij} = a_{ij}$ и $L_{ij} \leqslant y_{ij} \leqslant U_{ij}$.

Случай 3: $\gamma_{ij} = 0$ и $\delta_{ij} > 0$; $f_{ij} > a_{ij}$ и $y_{ij} = L_{ij}$.

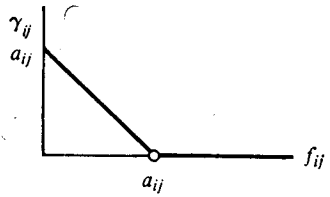
На основе условий дополняющей нежесткости для каждого случая находятся следующие условия оптимальности:

Случай 1

$$0 < f_{ij} < a_{ij} \quad \text{и} \quad t_i - t_j + U_{ij} = 0.$$

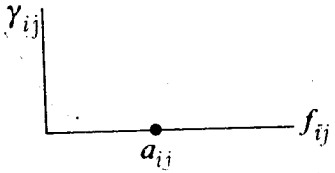
или

$$f_{ij}=0 \quad \text{и} \quad t_i-t_j+U_{ij}<0.$$



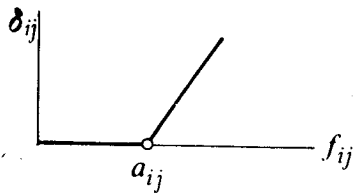
Случай 2

$$f_{ij}=a_{ij} \quad \text{и} \quad t_i-t_j+y_{ij}=0, \quad L_{ij}\leqslant y_{ij}\leqslant U_{ij}.$$



Случай 3

$$a_{ij}<f_{ij}<\infty \quad \text{и} \quad t_i-t_j+L_{ij}=0.$$



Введем следующие дополнительные символы:

$$\overline{U}_{ij}=t_i-t_j+U_{ij},$$
$$\overline{L}_{ij}=t_i-t_j+L_{ij},$$
$$\overline{y}_{ij}=t_i-t_j+y_{ij}.$$

Таким образом, условия оптимальности для каждого случая можно записать в более компактном виде:

Случай 1: $0<f_{ij}<a_{ij}$ и $\overline{U}_{ij}=0$ или $b_{ij}=0$ и $\overline{U}_{ij}<0$.

Случай 2: $f_{ij}=a_{ij}$ и $\overline{y}_{ij}=0$.

Случай 3: $a_{ij}<f_{ij}<\infty$ и $\overline{L}_{ij}=0$.

Эти условия играют важную роль при выводе потокового алгоритма в сетях, который будет рассмотрен далее.

Алгоритм. Процедура, рассматриваемая в данном разделе, была разработана Фалкерсоном [15], и ее можно рассматривать как двойственный метод, так как ее вывод основан на структуре двойственной задачи и условиях дополняющей нежесткости задач линейного программирования. Прежде чем приступать к описанию алгоритма, изложим некоторые простые, но полезные соображения.

Наиболее типичной интерпретацией дуги в сети с потоками является представление ее в виде канала или трубы, по которой

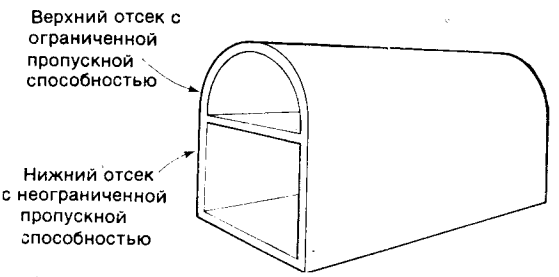


Рис. 4.17. Интерпретация дуги как трубопровода с двумя отсеками.

транспортируется определенный вид товара. Несколько более сложной, но удобной интерпретацией дуги в рамках рассматриваемого алгоритма является представление ее в виде трубопровода с двумя отсеками, как показано на рис. 4.17.

Верхний отсек трубопровода, изображенного на рис. 4.17, имеет ограниченную пропускную способность, а нижний отсек — неограниченную пропускную способность. При такой интерпретации предполагается, что поток проходит через нижний отсек только в том случае, когда превышается пропускная способность верхнего отсека. В противном случае нижний отсек не используется.

С помощью графического представления функций (рис. 4.16) можно убедиться, что случай 1 наблюдается, когда в верхнем отсеке еще отсутствует насыщение; случай 2 появляется, когда верхний отсек полностью заполнен; случай 3 имеет место, когда поток поступает в нижний отсек, пропускная способность которого, как уже говорилось выше, считается бесконечной. Таким образом, абсолютное значение тангенса угла наклона линии зависимости между затратами и продолжительностью данной работы можно интерпретировать как пропускную способность верхнего отсека дуги, изображающей эту работу на сетевом графике.

Рассматриваемый алгоритм может использоваться для построения кривой зависимости между затратами и продолжи-

тельностью проекта. В этом случае алгоритм начинается с максимальной продолжительности проекта, и на каждом шаге оцениваются дополнительные затраты, с помощью которых достигается некоторое сокращение продолжительности проекта. Пример такой кривой дан на рис. 4.18.

Алгоритм состоит из трех основных шагов. Первым шагом является проверка возможности сокращения заданной продолжительности проекта. На втором шаге осуществляется процедура расстановки пометок для модификации потоков в сети, соот-

Рис. 4.18. Кривая зависимости продолжительности проекта от затрат.

ветствующих двойственной задаче. На третьем шаге выполняется сокращение продолжительности проекта, если на втором шаге алгоритма достигается непрорыв. Блок-схема алгоритма изображена на рис. 4.19.

Начало вычислений. Если продолжительность проекта необходимо сократить относительно максимального значения, полагаем $\overline{T}=t_n$, где t_n можно получить рекуррентным способом с помощью следующего прямого соотношения:

$$t_j=\max[t_i+U_{ij}], \quad j=2, 3,\ldots, n \quad \text{и} \quad t_1=0.$$

Кроме того, все потоки могут быть равны нулю или какому-либо другому значению, удовлетворяющему условиям сохранения потока.

Шаг 1. Проверка возможности сокращения продолжительности проекта. Полагаем $y_{ij}=U_{ij}$. Если от узла 1 до узла n существует такой путь P, что $\overline{L}_{ij}=0$ для каждой дуги (i, j) пути P, то это означает, что $y_{ij}=L_{ij}$ для каждой дуги (i, j) данного пути. Однако в нетривиальных случаях $U_{ij}\neq L_{ij}$ и поэтому условие $T<\overline{T}$ невозможно. Если путь P не существует, то продолжительность проекта $T<\overline{T}$ возможна. Чтобы идентифицировать узлы пути P, их можно пометить в соответствии со следующим

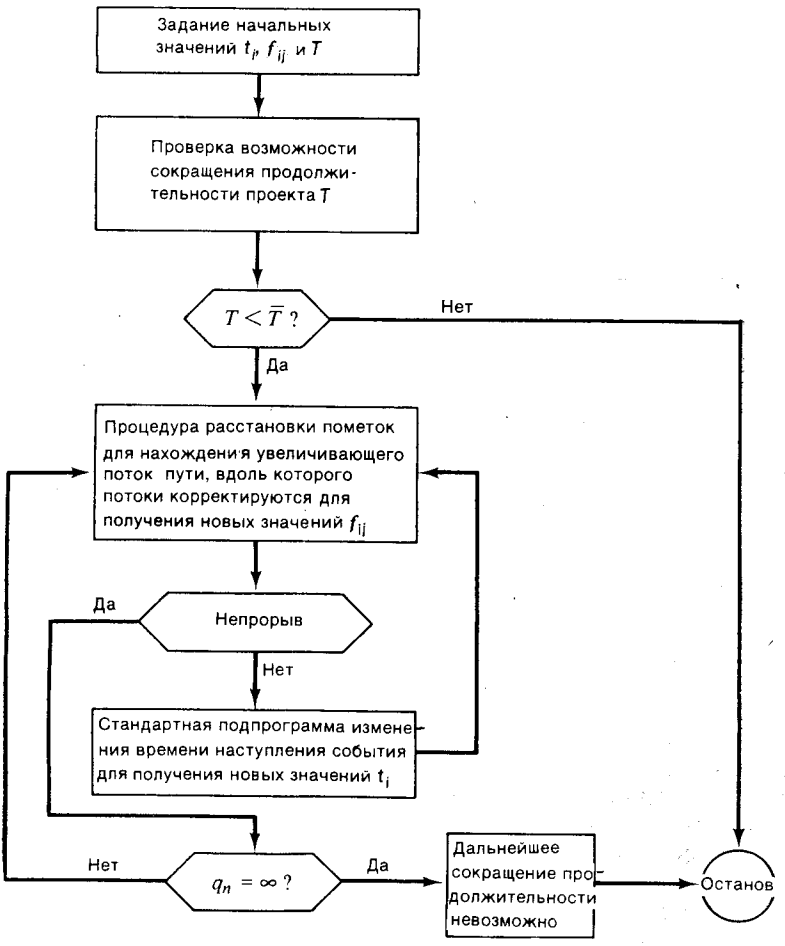


Рис. 4.19. Блок-схема алгоритма расчета потока сети.

правилом: помечаем j-й узел из i-го узла через $[\infty, i]$, если дуга (i, j) находится на пути P. В данной процедуре узел 1 имеет постоянную пометку $[\infty, 0]$.

Шаг 2. Процедура расстановки пометок. Разделим обсуждение процедуры расстановки пометок на две части: а) увеличение потока вдоль прямых дуг и б) уменьшение потока вдоль обратных дуг. а) Прямые дуги. Если дуга (i, j) относится к случаю 1, то для достижения оптимальности необходимо выполнение условия $\bar{U}_{ij}=0$. Таким образом, поток f_{ij} можно увеличить на минималь-

ное значение, лежащее между величиной потока в i-м узле и величиной, необходимой для достижения значения a_{ij}, которое отражает условия на поток для случая 2. Таким образом, j-му узлу можно дать пометку $[q_j, i]$, где $q_j=\min[q_i, a_{ij}-f_{ij}]$.

Если дуга (i, j) относится к случаю 3, то поток f_{ij} можно увеличить на любую величину, так как в этом случае пропускная способность дуги становится неограниченной. Поэтому весь поток, имеющийся в i-м узле, можно переместить в j-й узел. Это означает, что j-й узел может получить пометку $[q_j, i]$, где $q_j=q_i$.

б) Обратные дуги. Если дуга (i, j) относится к случаю 1, то после любого сокращения потока f_{ij} необходимым условием оптимальности остается $\bar{U}_{ij}=0$. Это означает, что из i-го узла в j-й узел можно направить встречный поток, величина которого ограничивается потоком, имеющимся в j-м узле. Назначение встречного потока состоит в том, чтобы уменьшить величину потока f_{ij}. Поэтому для i-го узла из узла j можно задать пометку $[q_i, j]$, где $q_i=\min[q_j, f_{ij}]$.

Если дуга (i, j) относится к случаю 3, то необходимым условием оптимальности остается $\bar{L}_{ij}=0$, если поток f_{ij} сокращается таким образом, что дуга (i, j) не переходит в случай 1. Следовательно, для дуги должны сохраняться условия случая 3 или обеспечиваться переход к случаю 2. Тогда для i-го узла можно задать пометку $[q_i, j]$ из j-го узла, где $q_i=\min[q_j, f_{ij}-a_{ij}]$.

Шаг 3. Процедура изменения момента наступления события. Этот этап выполняется только в том случае, когда происходит непрорыв, т. е. когда невозможно задать пометку для n-го узла. Пусть $\mathbf{E}$ — множество помеченных узлов, а $\bar{\mathbf{E}}$ — множество непомеченных узлов. Назначение этого шага алгоритма состоит в том, чтобы ускорить время наступления каждого события множества $\bar{\mathbf{E}}$.

Пусть $i\in\mathbf{E}$ и $j\in\bar{\mathbf{E}}$. Если дуга (i, j) удовлетворяет хотя бы одному из условий $\bar{L}_{ij}<0$ и $\bar{U}_{ij}<0$, то время наступления j-го события может быть сокращено, чтобы выполнялось это условие для y_{ij}, которое является необходимым для достижения оптимальности. Чтобы показать это, введем следующие определения:

$$\mathbf{B}=\{(i, j)\mid i\in\mathbf{E},\ j\in\bar{\mathbf{E}},\quad \bar{L}_{ij}<0 \text{ или } \bar{U}_{ij}<0\},$$

$$\zeta_1=\min_{\mathbf{B}}[\bar{y}_{ij}=y_{ij}+t_i-t_j<0].$$

Условие $\bar{y}_{ij}=y_{ij}+t_i-t_j$ можно переписать в виде $0=y_{ij}+t_i-(t_j)+\bar{y}_{ij})$, откуда следует, что t_j можно уменьшить на $-\bar{y}_{ij}$ (заметим, что $\bar{y}_{ij}<0$), чтобы удовлетворялось условие, согласно

которому новое значение $\bar{y}_{ij}$, вычисленное при новом значении t_j, равно нулю, что требуется для обеспечения оптимальности.

Аналогично если дуга (j, i) удовлетворяет хотя бы одному из условий $\bar{L}_{ji}>0$ и $\bar{U}_{ji}>0$, то время наступления j-го события может быть сокращено, чтобы выполнялось условие для y_{ji}, ко-

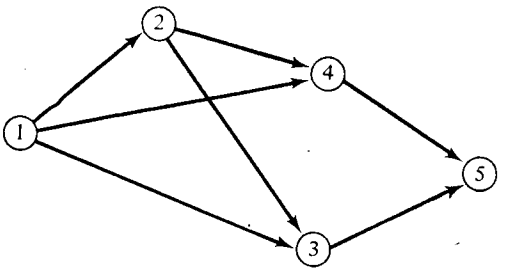


Рис. 4.20. Сетевая модель для примера.

торое необходимо для достижения оптимальности. Как и ранее, вводим следующие определения:

$$\bar{\mathbf{B}}=\{(j, i)\,|\,i\in\bar{\mathbf{E}},\ j\in\bar{\mathbf{E}},\ \bar{L}_{ji}>0 \text{ или } \bar{U}_{ji}>0\},$$

$$\zeta_2=\min_{\bar{\mathbf{B}}}[\bar{y}_{ji}=y_{ji}+t_j-t_i>0].$$

Условие $\bar{y}_{ji}=y_{ji}+t_j-t_i$ можно переписать в виде $0=y_{ji}+(t_j-\bar{y}_{ji})-t_i$, откуда следует, что t_j можно сократить на y_{ji} (заметим, что $y_{ij}>0$), чтобы выполнялось условие, согласно которому новое значение $\bar{y}_{ji}$, вычисленное при новом значении t_j, было равно нулю, что требуется для обеспечения оптимальности.

Если производится такое же сокращение времени наступления каждого j-го события множества $\bar{\mathbf{E}}$, то может использоваться следующая процедура.

1. Находим ζ_1.
2. Находим ζ_2.
3. Находим $\zeta=\min[\zeta_1, \zeta_2]$.
4. Уменьшаем t_j на ζ для каждого узла $j\in\bar{\mathbf{E}}$.

После изменения всех значений t_j начинаем процедуру расстановки пометок (шаг 2), беря во всех случаях в качестве пометки $[\infty, i]$ для того, чтобы сократить объем вычислений.

Пример оптимального соотношения между временем и затратами. Построим кривую для соотношения между затратами и продолжительностью проекта, представленного на рис. 4.20. Дополнительные данные, необходимые для решения этой задачи, приводятся в табл. 4.15.

Таблица 4.15. Данные о сроках и затратах

Дуга (i, j)	L_{ij}	U_{ij}	a_{ij}
(1, 2)	2	5	3
(1, 3)	1	3	2
(1, 4)	4	6	2
(2, 3)	2	4	5
(2, 4)	3	7	1
(3, 5)	0	3	4
(4, 5)	3	3	—

Таблица 4.16. Сроки наступления начального события

Узел	Инцидентные дуги	t_i
1	—	0
2	(1, 2)	max [0 + 5] = 5
3	(1, 3), (2, 3)	max [0 + 3, 5 + 4] = 9
4	(1, 4), (2, 4)	max [0 + 6, 5 + 7] = 12
5	(3, 5), (4, 5)	max [9 + 3, 12 + 3] = 15

Каждая дуга помечается четырьмя числами $[L_{ij}, U_{ij}, a_{ij}, f_{ij}]$. Момент наступления каждого события обозначается $[t_i]$ и записывается возле i-го узла. Все начальные значения потока при-

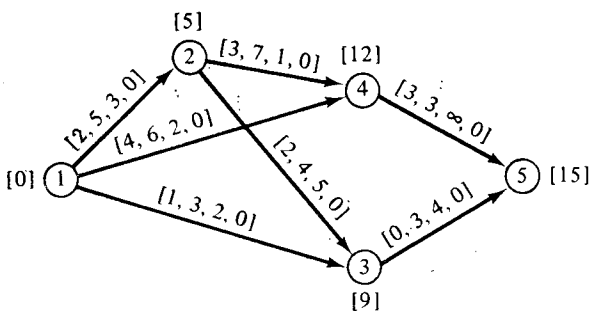


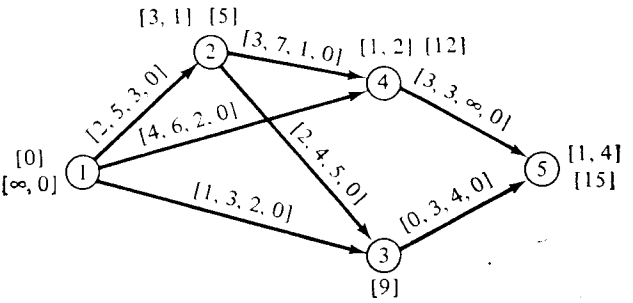
Рис. 4.21. Исходные решения задачи.

нимаются равными нулю. Начальные значения t_i можно вычислить, как показано в табл. 4.16. Начальные решения показаны на рис. 4.21.

Проверка возможности сокращения продолжительности проекта. После задания $y_{ij}=U_{ij}$ для всех дуг сети невозможно существование пути со всеми $\bar{L}_{ij}=0$. Поэтому условие $T<\bar{T}=15$ является возможным.

Итерация 1

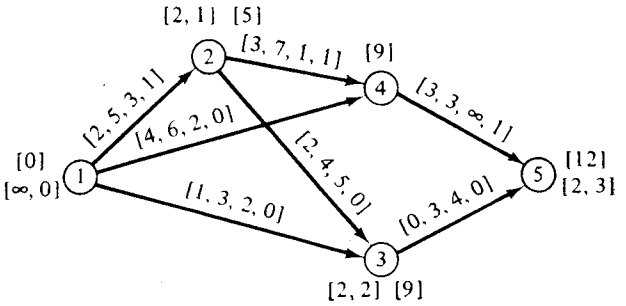
Дуга	$\bar{U}_{ij}$	$\bar{L}_{ij}$
(1, 2)	0	−3
(2, 4)	0	−4
(4, 5)	0	0



Результатом этой итерации является прорыв при $q_5=1$. На каждой дуге аугментального пути потока (1, 2)—(2, 4)—(4, 5) происходит увеличение потока на 1.

Итерация 2

Дуга	$\bar{U}_{ij}$	$\bar{L}_{ij}$
(1, 2)	0	−3
(2, 3)	0	−2
(3, 5)	0	−3



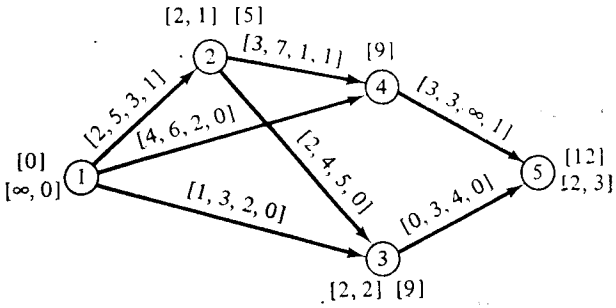
Поскольку пометить узел 5 не оказалось возможным, получаем непрорыв:

$$\mathbf{B}=\{(3,5),\ (2,4),\ (1,4)\},\qquad \zeta_1=3,$$
$$\bar{\mathbf{B}}=\varnothing,\qquad \zeta_2=\infty.$$

Следовательно, $\zeta=3$. Как t_4, так и t_5 должны уменьшаться на величину $\zeta=3$, т. е. $t_4=9$, $t_5=12$.

Итерация 3

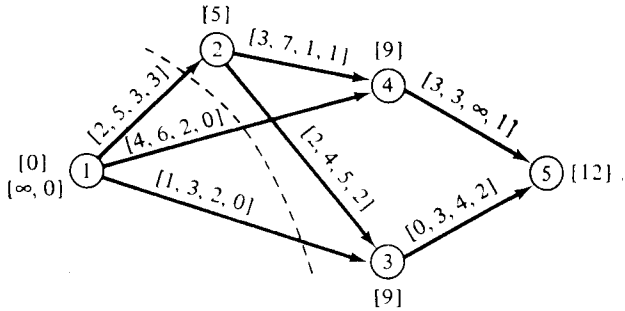
Дуга	$\overline{U}_{ij}$	$\overline{L}_{ij}$
(1, 2)	0	−3
(2, 3)	0	−2
(3, 5)	0	−3



Получен прорыв при $q_5=2$. На каждой дуге аугментального пути потока (1, 2)—(2, 3)—(3, 5) происходит увеличение потока на 2.

Итерация 4

Дуга	$\overline{U}_{ij}$	$\overline{L}_{ij}$
(1, 2)	0	−3
(1, 3)	−6	−8
(1, 4)	−3	−5



Поскольку пометить узел 5 невозможно, результатом является непрорыв:

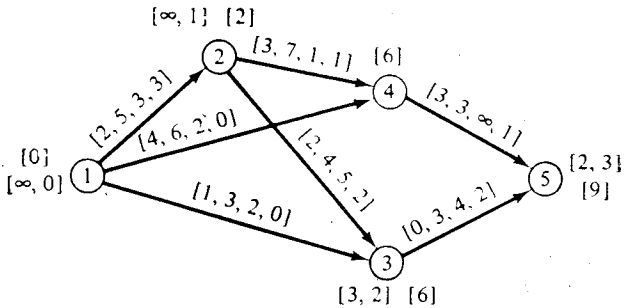
$$\mathbf{B}=\{(1,2),\ (1,3),\ (1,4)\},\quad \zeta_1=3,$$

$$\overline{\mathbf{B}}=\varnothing,\qquad \zeta_2=\infty.$$

Следовательно, $\zeta=3$ и $t_2=2$, $t_3=6$, $t_4=6$ и $t_5=9$.

Итерация 5

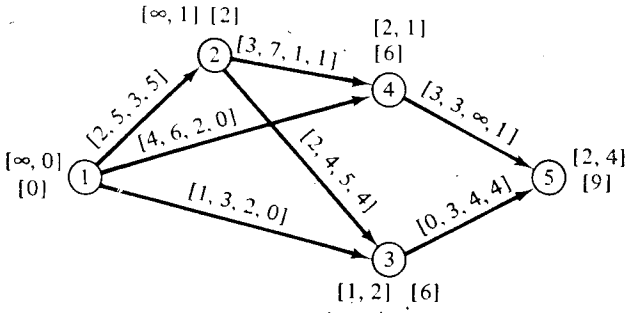
Дуга	$\overline{U}_{ij}$	$\overline{L}_{ij}$
(1, 2)	3	0
(2, 3)	0	−2
(3, 5)	0	−3



Результатом этой итерации является прорыв при $q_5=2$. На каждой дуге аугментального пути потока (1, 2)—(2, 3)—(3, 5) происходит увеличение потока на 2.

Итерация 6

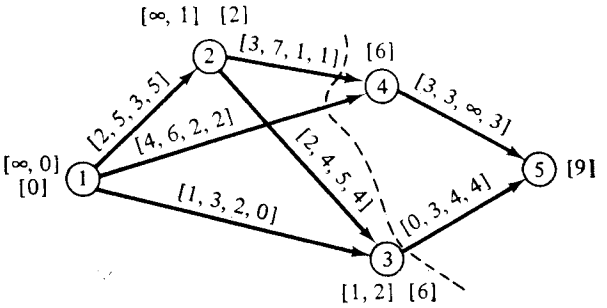
Дуга	$\overline{U}_{ij}$	$\overline{L}_{ij}$
(1, 2)	3	0
(2, 3)	0	−2
(3, 5)	0	−3
(1, 4)	0	−2
(4, 5)	0	0



Получен прорыв при $q_5=2$. На каждой дуге аугментального пути потока (1, 4)—(4, 5) происходит увеличение потока на 2.

Итерация 7

Дуга	$\overline{U}_{ij}$	$\overline{L}_{ij}$
(1, 2)	3	0
(2, 3)	0	−2
(3, 5)	0	−3
(2, 4)	3	−1
(1, 4)	0	−2



Поскольку пометить узел 5 невозможно, получаем непрорыв:

$$\mathbf{B}=\{(1,4),\ (2,4),\ (3,5)\},\qquad \zeta_1=1,$$

$$\overline{\mathbf{B}}=\underline{\varnothing},\qquad \zeta_2=\infty.$$

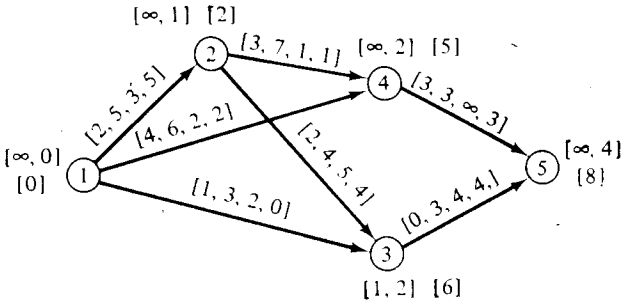
Таблица 4.17. Оптимальное решение

Итерация	t_5	$F=\sum f_{i5}$	Сокращение продолжительности, R	R F	Суммарное приращение
2	15	1	3	3	3
4	12	3	3	9	12
7	9	7	1	7	19
8	8	—	Прекращение вычислений	—	—

Следовательно, $\zeta=1$ и $t_4=5$, $t_5=8$.

Итерация 8

Дуга	$\bar{U}_{ij}$	$\bar{L}_{ij}$
(1, 2)	3	0
(2, 3)	0	−2
(3, 5)	1	−2
(2, 4)	4	0
(4, 5)	0	0



Получен прорыв при $q_5=\infty$. Следовательно, это оптимальное решение. Результаты приводятся в табл. 4.17. Соответствующая кривая зависимости между затратами и продолжительностью проекта представлена на рис. 4.22.

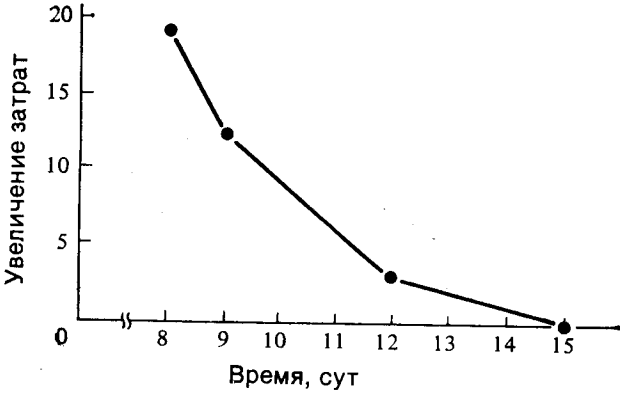


Рис. 4.22. Кривая зависимости продолжительности от затрат.

Продолжительности работ, соответствующие каждой точке излома кривой зависимости между затратами и продолжительностью проекта, изображенной на рис. 4.22, можно получить с помощью условий дополняющей нежесткости. Поскольку значения потока f_{ij} и параметра a_{ij} известны на каждой итерации алгоритма, можно вычислить значения γ_{ij} и δ_{ij} с помощью соотношений

$$\gamma_{ij}=\max[0,\ a_{ij}-f_{ij}],$$
$$\delta_{ij}=\max[0,\ f_{ij}-a_{ij}].$$

Таблица 4.18. Сводка результатов

Дуга	$T=8$				$T=9$				$T=12$				$T=15$			
(i,j)	f_{ij}	γ_{ij}	δ_{ij}	y_{ij}	f_{ij}	γ_{ij}	δ_{ij}	y_{ij}	f_{ij}	γ_{ij}	δ_{ij}	y_{ij}	f_{ij}	γ_{ij}	δ_{ij}	y_{ij}
(1, 2)	5	0	2	2	5	0	2	2	3	0	0	5[a]	1	2	0	5
(1, 3)	0	2	0	3	0	2	0	3	0	2	0	3	0	2	0	3
(1, 4)	2	0	0	5[a]	0	2	0	6	0	2	0	6	0	2	0	6
(2, 3)	4	1	0	4	4	1	0	4	2	3	0	4	0	5	0	4
(2, 4)	1	0	0	3[a]	1	0	0	4[a]	1	0	0	4[a]	1	0	0	7[a]
(3, 5)	4	0	0	2[a]	4	0	0	3[a]	2	2	0	3	0	4	0	3
(4, 5)	3	∞	0	3	1	∞	0	3	1	∞	0	3	1	∞	0	3

[a] Поскольку как γ_{ij}, так и δ_{ij} равны нулю, условия дополнительности указывают лишь, что продолжительность работы лежит в некотором интервале. В этой задаче устанавливается максимально допустимая продолжительность работы, так как модель имеет целью минимизировать общую стоимость проекта.

В силу условий дополняющей нежесткости задачи линейного программирования при $\gamma_{ij}>0$ имеем $y_{ij}=U_{ij}$, а при $\delta_{ij}>0$ имеем $y_{ij}=L_{ij}$. В противном случае $L_{ij}\leqslant y_{ij}\leqslant U_{ij}$. Результаты численного решения рассматриваемой задачи приводятся в табл. 4.18.

4.12.2. ПРИМЕНЕНИЕ ПРОЦЕДУР УСТАНОВЛЕНИЯ КОМПРОМИССНОГО СООТНОШЕНИЯ МЕЖДУ ЗАТРАТАМИ И ПРОДОЛЖИТЕЛЬНОСТЬЮ ПРОЕКТА

Примеры основных процедур установления компромиссного соотношения между затратами и продолжительностью проекта даются в основополагающих статьях Келли и Уолкера [21—23] и Фалкерсона [15]. Процедура, описанная в этих статьях, основана на линейном программировании и осуществляется с помощью потокового алгоритма в сети, для которого уже давно имеются машинные программы, хранящиеся в библиотеке фирмы IBM [11]. Такая программа автоматически выдает кривую значений продолжительности проекта при минимальных затратах и подробные данные о сроках начала и окончания работ, связанные с каждой точкой кривой. Одним из преимуществ этой стандартной программы вычисления потока в сети является то, что при использовании в качестве исходных данных целочисленных продолжительностей работ точки излома кривой стоимости (возможные продолжительности проекта) также являются целыми числами. Однако необходимы допущения о линейных соотношениях между временем и затратами.

Другой подход к этой проблеме при иных допущениях о соотношениях между временем и затратами изложен в руковод-

стве Национального управления по аэронавтике и исследованию космического пространства (НАСА) и министерства обороны США по применению системы ПЕРТ-СТОИМОСТЬ [32]. Еще один полезный вариант основной процедуры разработал Фондал [14]. Схема вычислений, выполняемых вручную, позволяет сокращать и увеличивать продолжительность проекта при различных допущениях о соотношении между временем и затратами с целью определить минимальные затраты при требуемой продолжительности проекта. Большинство методов анализа соотношений между временем и затратами, разработанных до 1966 г., подробно рассматривается в статье [11].

К числу сравнительно новых процедур вывода соотношений между временем и затратами, разработанных после 1966 г., относится простой метод Сименса [42], который может применяться вручную или с использованием машинной программы; он позволяет получить оптимальные результаты при линейных соотношениях и близкие к оптимальным в случае нелинейных соотношений. Бенсон и Сьюолл [2] описывают имеющуюся в продаже машинную программу для анализа соотношений между затратами и продолжительностью проекта в сетях, содержащих до 40 тыс. работ. Холландер [17] описывает еще одну большую программу, в которой могут вводиться различные допущения о соотношениях между затратами и продолжительностью работы. Модер и Филлипс [29] дают четкое изложение и сравнение процедур анализа соотношений между затратами и временем и приводят список нескольких машинных программ, предназначенных для анализа соотношений между затратами и временем.

Важно заметить, что в процедурах анализа соотношений между затратами и временем ограничения на наличие ресурсов обычно не рассматриваются в явном виде. Ни одна из упоминавшихся здесь процедур не позволяет автоматически получить календарные графики работ при минимальных затратах, удовлетворяющие заданным ограничениям на наличие таких ресурсов, как денежные средства, рабочая сила и оборудование. Например, хотя программа Бенсона и Сьюолла во многих отношениях является исключительно сложной, она не позволяет автоматически планировать сроки выпуска продукции. Программа выдает отчеты, показывающие, что для отдельных работ нарушаются ограничения на затраты или время, но задача внесения подробных изменений, необходимых для соблюдения ограничений, налагаемых на проект, ложится на человека.

Необходимо также заметить, что процедуры анализа компромиссных соотношений между затратами и временем не находят широкого применения на практике. Основной причиной этого являются большой объем и сложность исходных данных, необходимых для анализа, а также типичная неопределенность

этих данных. Обычно сбор подробной информации о соотношении между затратами и продолжительностью проекта, содержащего, например, сотни работ, является непростой задачей. Таким образом, хотя информация, получаемая с помощью процедур анализа соотношений между затратами и временем, потенциально очень полезна в некоторых ситуациях, в целом сегодня такие процедуры занимают сравнительно скромное место в общей иерархии сетевых методов распределения ресурсов.

4.13. РАСПРЕДЕЛЕНИЕ РЕСУРСОВ

Одним из основных преимуществ представления проекта в виде сетевой модели является возможность легко получать информацию о потребностях в каждом промежутке времени в таких ресурсах, необходимых для выполнения проекта, как рабочая сила, оборудование и денежные средства. Эта информация является побочным результатом вычислений, выполняемых обычным МКП. Единственное требование состоит в том, что

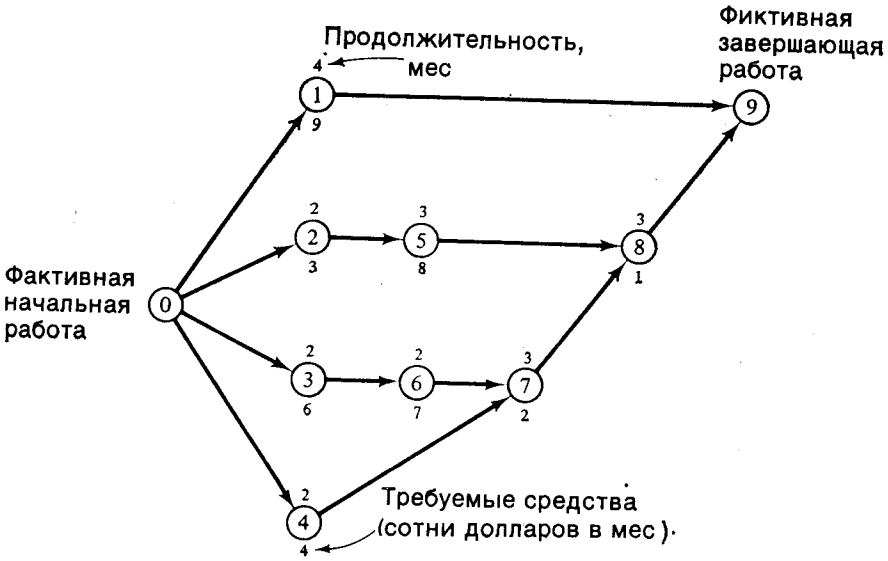
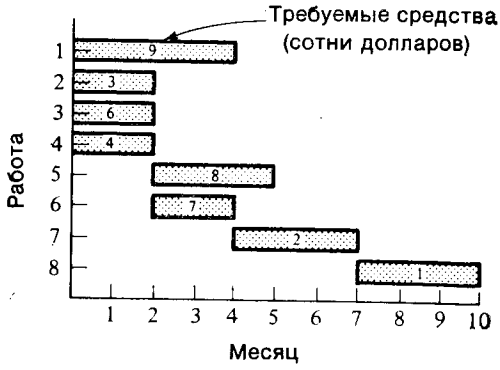


Рис. 4.23. Пример сетевой модели проекта с заданными продолжительностями работ и потребностями в ресурсах.

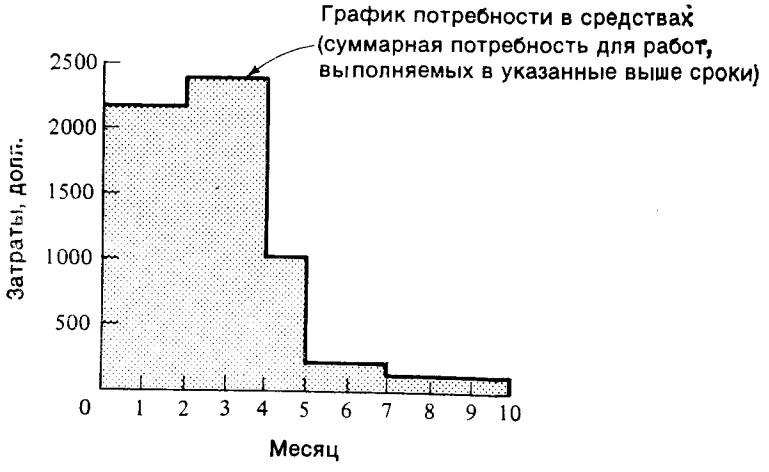
запросы ресурсов для выполнения каждой работы проекта, представленной в сетевой модели, должны определяться отдельно.

Например, на рис. 4.23 показана сеть типа работа-узел для простого гипотетического проекта. Около каждого узла сети,

изображающего определенную работу проекта, показаны ее продолжительность в месяцах и требуемые затраты в сотнях долларов в месяц. С помощью обычных вычислений МКП для каждой работы получены наиболее ранний возможный срок



а



б

Рис. 4.24. Построение графика распределения ресурсов для рассматриваемого примера.

а — столбиковая диаграмма, построенная с помощью сетевой модели (все работы начинаются в наиболее ранние возможные сроки); *б* — график потребности в ресурсах.

начала, наиболее поздний допустимый срок окончания и другие данные. Используя эту информацию, можно построить гистограмму, подобную изображенной на рис. 4.24, *а*, на которой все работы начинаются в соответствующие наиболее ранние воз-

можные сроки. Эта гистограмма в свою очередь используется для построения графика движения расходуемых денежных средств, представленного на рис. 4.24, *б*. Подобный анализ может выполняться вручную или автоматически с помощью ЭВМ, имеющей программу для вычислений МКП.

Для иллюстрации в этих примерах рассматривается потребность в денежных средствах. Очевидно, что могут использоваться данные и о других ресурсах, например рабочей силе. В этих случаях график распределения ресурсов во времени обычно называется *диаграммой потребления ресурсов*. Такие диаграммы исключительно полезны при управлении проектами. Они показывают влияние расхода ресурсов на сроки осуществления определенного проекта и дают основу для более рационального планирования проекта. Например, одна крупная фармацевтическая фирма регулярно получает информацию о распределении ресурсов с помощью графиков критического пути (расписаний) для всех научно-исследовательских и опытно-конструкторских проектов с целью обеспечить проверку доступности ресурсов и наличия высококвалифицированных медицинских и научных работников, необходимых в любой момент времени. Такие процедуры широко используются также в строительстве, а некоторые небольшие фирмы часто вручную строят графики потребности в рабочей силе, используя МКП. В большинстве случаев вследствие неопределенности сроков данные о потребности в ресурсах скорее являются ориентировочными плановыми показателями, чем точно заданными величинами.

В настоящее время большинство хороших программ для вычислений МКП автоматически выдает информацию о потребности в ресурсах (в виде таблиц или графиков), даже если в этих программах не обеспечивается более сложная процедура автоматической корректировки календарных планов для удовлетворения потребности в ресурсах. Эта процедура осуществляется путем регулирования потребления ресурсов или календарного планирования при заданных ограничениях на наличие ресурсов.

4.14. РЕГУЛИРОВАНИЕ ПОТРЕБЛЕНИЯ РЕСУРСОВ

Во многих ситуациях календарного планирования проектов общая потребность в необходимых ресурсах, планируемая к определенному сроку на основе графика расхода ресурсов, может не представлять большого интереса вследствие наличия требуемых ресурсов в достаточном количестве. Однако может оказаться, что график (расписание) потребности в ресурсах обладает нежелательными свойствами, например требуется частое изме-

нение числа работников определенной квалификации. В этих случаях используются методы регулирования потребления ресурсов; они обеспечивают распределение ресурсов во времени с целью свести к минимуму колебания потребности в рабочей силе, оборудовании или денежных средствах. Они могут также использоваться для определения возможности снижения максимальной потребности в ресурсах без увеличения продолжительности проекта.

Принципы регулирования потребления ресурсов легко пояснить на простом примере. Рассмотрим сетевой график, изображенный на рис. 4.23, но будем считать, что цифры возле узлов показывают продолжительность работ в сутках, а не в месяцах, и не требуемые денежные средства в сотнях долларов в месяц, а необходимое число работников определенной квалификации в сутки. Тогда изображенный на рис. 4.25, *а* график потребности в ресурсах окажется неудобным с точки зрения загрузки рабочей силы, так как требуемое число работников сильно колеблется — от 24 человек в 3-й и 4-й дни до одного человека в 8, 9 и 10-й дни. При регулировании потребления ресурсов можно получить более равномерное использование рабочей силы без увеличения продолжительности проекта.

Как видно из гистограммы на рис. 4.25, *а*, работы 1, 2, 4 и 5 имеют резерв времени и их продолжительность можно увеличить в пределах соответствующего резерва без задержки других работ. Например, работу 1, как показано на рис. 4.25, *б*, можно перенести на последние 4 дня осуществления проекта. Это позволит снизить максимальную потребность в рабочей силе с 24 до 14 человек, а полученный в итоге график потребности в рабочей силе станет более равномерным. Однако значительные колебания все же сохраняются, и в 6-й день наблюдается нежелательный резкий спад занятости. При последующем сдвиге сроков выполнения работ 2 и 5 можно получить более равномерную загрузку рабочей силы, что показано на рис. 4.25, *в*. Этот окончательный график все еще содержит нежелательные пики во 2-й и 4-й дни, но их нельзя устранить без увеличения продолжительности проекта или изменения характера некоторых работ с тем, чтобы изменить соответствующие потребности в рабочей силе.

Как показывает этот простой пример, основная идея регулирования потребления ресурсов состоит в маневрировании рабочей силой и сдвиге сроков работ в пределах имеющихся резервов времени для обеспечения лучшего распределения потребляемых ресурсов. Имеющийся резерв времени для каждой работы определяется путем стандартных вычислений с помощью МКП. Таким образом, продолжительность проекта (т. е. длина критического пути) определяется только путем анализа сроков без

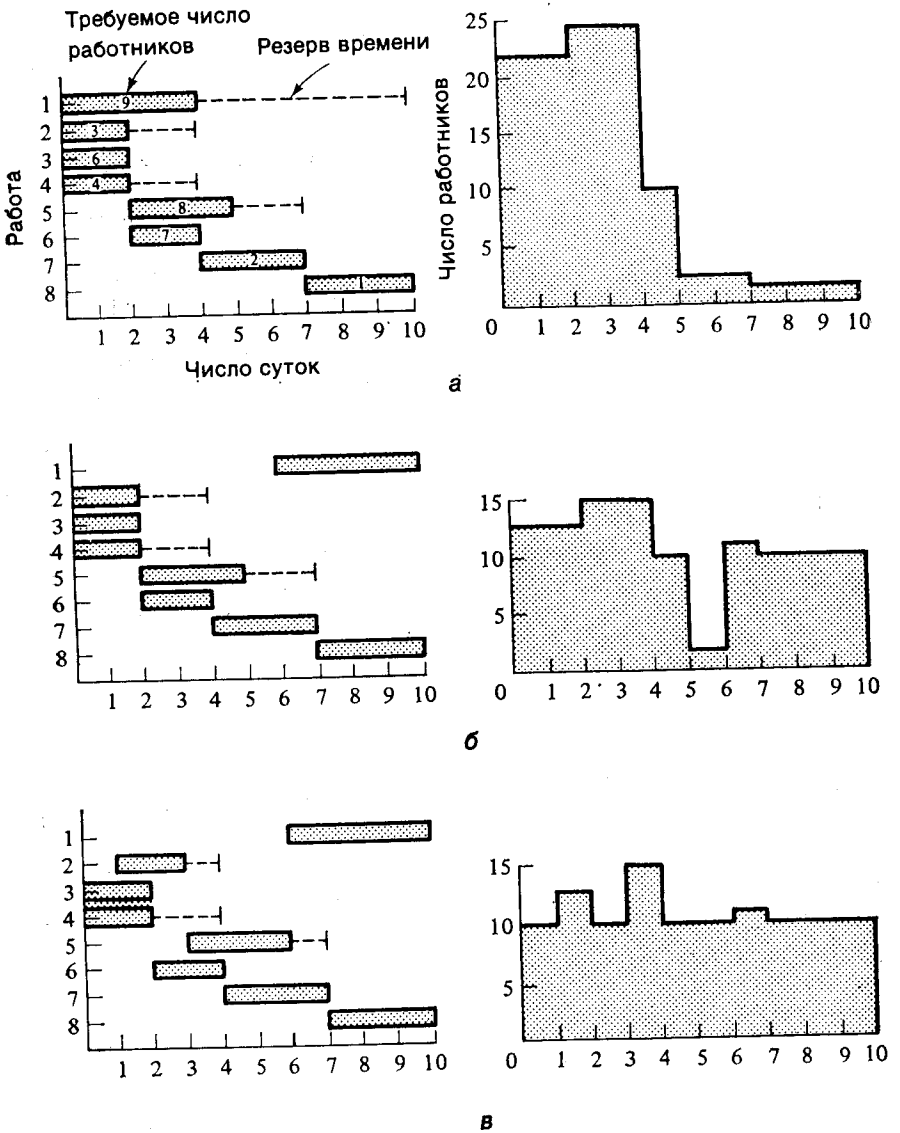


Рис. 4.25. Более равномерное распределение ресурсов.

а — столбиковая диаграмма и график потребности в рабочей силе, когда все работы начинаются в наиболее ранние возможные сроки; *б* — изменены сроки выполнения работы 1; *в* — изменены сроки выполнения работ 1, 2 и 5.

учета потребности в ресурсах, а в процессе регулирования потребления ресурсов продолжительность проекта не может быть увеличена.

В тех случаях, когда рассматривается ресурс только одного вида, регулирование потребления ресурса можно легко выполнить вручную, даже при достаточно больших сетевых графиках, с помощью магнитных наборных панелей для календарного планирования, полосок бумаги или других приспособлений. Однако в случае больших сетей с несколькими ресурсами процесс усложняется, так как регулирование потребления одного ресурса может привести к еще большей неравномерности в потреблении других ресурсов. В этих случаях вычисления лучше всего выполнять на ЭВМ. С начала 60-х годов имеются программы вычислений методом критического пути с регулированием потребления ресурсов [11]. Леви и Уист [26] дают описание применения одной такой программы. Модер и Филлипс [29] приводят список программ для вычисления методом критического пути с регулированием потребления ресурсов. Некоторые из этих программ обеспечивают календарное планирование работ с выравниванием потребления нескольких сотен видов ресурсов в сетях, содержащих тысячи работ. Некоторые из этих программ основаны на принципе чисто механического применения некоторого правила маневрирования ресурсами, или эвристики. Однако в некоторых из них используются довольно сложные стандартные подпрограммы, способные объединять взаимодополняющие ресурсы, устанавливать уровни потребления ресурсов и осуществлять прогнозирование.

4.15. ЗАДАНИЕ ПРЕДЕЛЬНОГО КОЛИЧЕСТВА РЕСУРСОВ

В процессе рассмотренного выше выравнивания потребления ресурсов расход различных ресурсов можно регулировать в зависимости от имеющихся резервов времени для выполнения работ. Однако такой процесс не всегда дает удовлетворительные календарные планы, если количество имеющихся ресурсов строго ограничено. Например, на рис. 4.25, *в* окончательная потребность в рабочей силе составляет 10 человек в сутки, за исключением 2, 4 и 7-го дней, когда потребность больше. Если число работников, выделяемых для этого проекта, не может превышать 10 человек в сутки и если потребность работ в ресурсах нельзя сократить соответствующим образом, то единственным способом срезать пики, чтобы получить календарный план, обеспеченный требуемыми ресурсами, является последующее изменение календарного плана, что приведет к увеличению продолжительности проекта. Например, из рис. 4.25, *в* видно, что выровнять потребность в рабочей силе можно, перенеся начало работы 1 с 7-го на 8-й день. Это позволит задержать работу 5

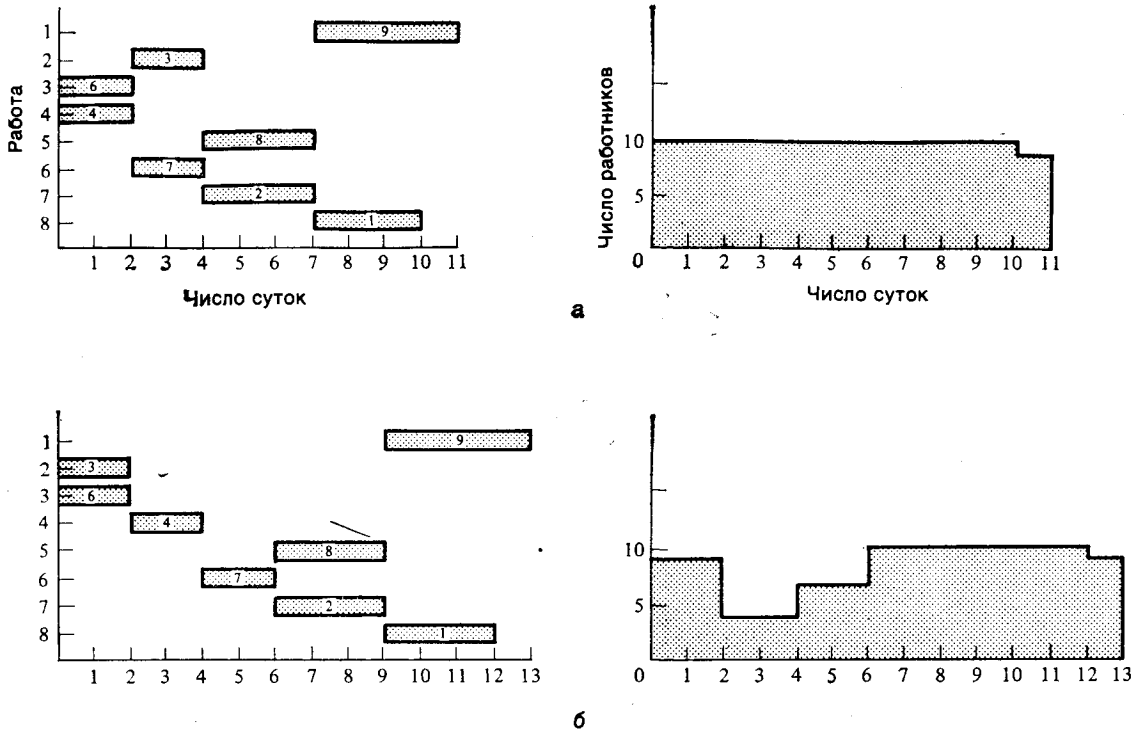


Рис. 4.26. Календарное планирование при ограниченных ресурсах, когда имеется только 10 работников.

а — график продолжительностью 11 сут, полученный за счет перераспределения имеющихся ресурсов; *б* — график продолжительностью 13 сут, полученный с помощью правила «вначале выполняется самая короткая работа».

на один день и выполнять ее одновременно с работой 7, а также задержать работу 2 на один день и выполнять ее одновременно с работой 6. Поскольку в течение всего периода потребность в рабочей силе не превышает 10 человек в сутки, такой календарный план является выполнимым с точки зрения наличия ресурсов. Однако в этом случае, как показано на рис. 4.26,*а*, продолжительность проекта увеличивается на одни сутки.

4.16. ОГРАНИЧЕННЫЕ РЕСУРСЫ

Рассмотренное в данном примере изменение календарного плана относится к другой категории процедур анализа сетей, называемой *календарным планированием при ограниченных ресурсах*. Эти методы предназначены для составления календарных планов, не требующих большего количества ресурсов, чем имеется в любой данный промежуток времени, и продолжительность проекта увеличивается по сравнению с первоначальной, определенной с помощью МКП, как можно меньше.

Процедуры календарного планирования в сетях с ограниченными ресурсами были впервые предложены вскоре после появления общих моделей критического пути и ПЕРТ. Например, Келли и Уолкер [22] описали в одной из своих ранних статей разработку программы для ЭВМ IBM-650, способной регулировать загрузку до четырех различных видов рабочей силы на одну работу и до девяти — на проект. В 1961 г. эта процедура уже использовалась для нескольких проектов и обеспечила снижение максимальной потребности в рабочей силе на 35—50% при увеличении продолжительности проекта в среднем примерно на 5%.

Огромное множество процедур календарного планирования с ограниченными ресурсами можно разбить на две большие группы на основе используемых методов и их полезности для руководителей. К первой самой крупной группе относятся эвристические, или приближенные процедуры, предназначенные для составления хороших календарных планов, выполнимых с точки зрения наличия ресурсов. Напротив, ко второй группе относятся процедуры, предназначенные для составления наилучших (оптимальных) планов и основанные на использовании линейного программирования, метода частичного перебора и других подходов. Эти математические методы оптимизации могут применяться для решения значительно менее сложных задач, чем эвристические.

4.16.1. ЭВРИСТИЧЕСКИЕ МЕТОДЫ

Поскольку эвристические методы играют важную роль в календарном планировании при ограниченных ресурсах, необходимо понять их общие принципы и некоторые особенности. Для

этого лучше всего обратиться к некоторым иллюстративным примерам.

Рассмотрим снова календарный план проекта продолжительностью 11 сут, представленный на рис. 4.26, *а*, который был получен в результате выравнивания потребления ресурсов (рис. 4.25, *в*). В большинстве эвристических процедур решение этой задачи начинается с рассмотрения первоначального календарного плана, определенного с помощью МКП, в котором все работы начинаются в наиболее ранние возможные сроки (рис. 4.25, *а*). Необходимо рассмотреть каждый промежуток времени календарного плана с целью определить, не превышен ли предельный уровень рабочей силы — 10 человек. Если этот уровень превышен, необходимо изучить список работ, планируемых к одновременному выполнению в этом периоде. Следует применять некоторое принятое правило, или эвристику, например, такое, как «вначале выполняется самая короткая работа», чтобы определить, выполнение каких работ следует задержать. В случае наличия «связки», т. е. нескольких работ одинаковой продолжительности, должна применяться другая эвристика, разрывающая «связку», например «вначале выполняется работа, имеющая наименьший порядковый номер». Эти правила последовательно применяются механически в каждом промежутке времени (и для каждого вида ресурсов, если их более одного) всего календарного плана, обеспечивая перенос сроков рассматриваемых и следующих за ними работ до тех пор, пока не будут заданы сроки выполнения всех работ и будет получен календарный план, осуществимый с точки зрения обеспеченности ресурсами. В качестве иллюстрации данного процесса применение описанной здесь эвристики «вначале выполняется самая короткая работа» позволяет получить календарный план продолжительностью 13 сут, изображенный на рис. 4.26, *б*. Этот календарный план действительно осуществим с точки зрения обеспеченности ресурсами, но является менее удовлетворительным, чем предыдущий план продолжительностью 11 сут.

Легко показать, почему применение правила «вначале выполняется самая короткая работа» позволяет получить такие результаты. Например, в промежутке времени 1 механическое применение этой эвристики приводит к задержке работы 4, а работы 2 и 3 выполняются одновременно. Однако очевидно, что лучше было бы перенести срок выполнения работы 2, а работы 3 и 4 выполнять одновременно. Это позволило бы иметь другое распределение работ в последующих периодах и получить менее продолжительный календарный план с более высокими уровнями использования рабочей силы.

Использование других, значительно более сложных эвристик позволяет получить «наилучший из числа возможных», т. е.

рассмотренный ранее календарный план продолжительностью 11 сут. Однако на практике даже при использовании более сложных правил невозможно заранее сказать, какая эвристика или комбинация эвристик обеспечит получение результатов, наилучших для данной задачи. Эвристики, являющиеся неудов-

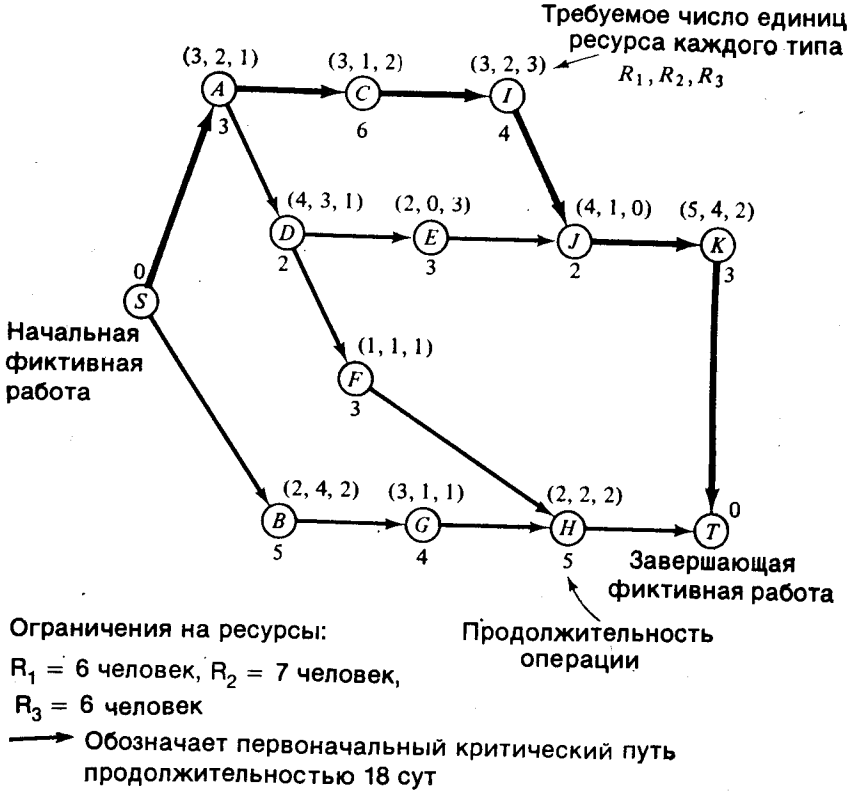
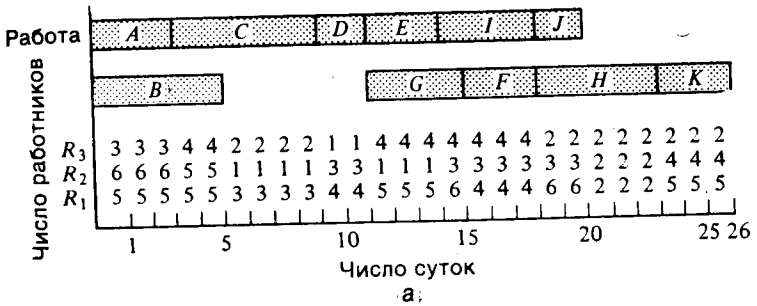
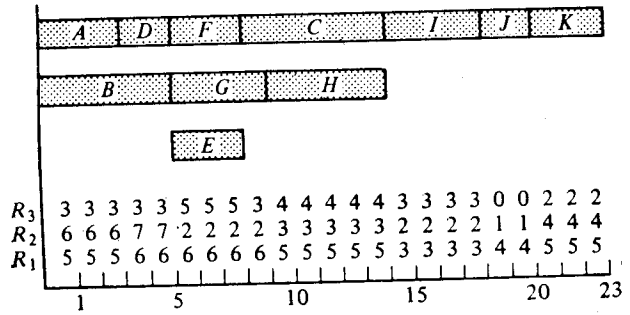


Рис. 4.27. Задача с тремя видами ограниченных ресурсов.

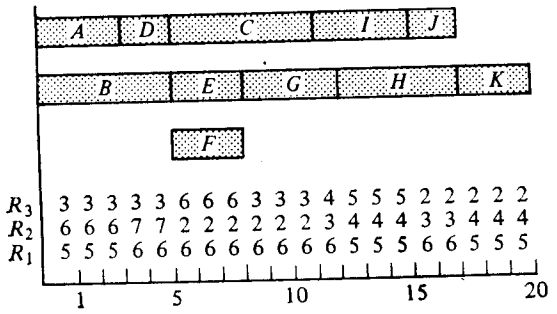
летворительными для одной задачи, могут оказаться хорошими для другой. Например, на рис. 4.27 показана иная, несколько более сложная сеть для трех видов ресурсов. Вследствие указанных ограничений, налагаемых на ресурсы, первоначальная продолжительность критического пути, равная 18 сут, не может быть достигнута. На рис. 4.28 показаны календарные планы, полученные с помощью двух различных эвристик, в сравнении с календарным планом оптимальной продолжительности, определенным с помощью алгоритма, приведенного в статье Дэвиса и Хейдорна [10]. В этом случае правило «вначале выполняет-



а



б



в

Рис. 4.28. Сравнение эвристических и оптимальных календарных графиков для задачи с тремя видами ресурсов.

а — график продолжительностью 26 сут, осуществимый с точки зрения обеспеченности ресурсами и полученный с помощью эвристики (метод календарного планирования ресурсов); *б* — график продолжительностью 23 сут, осуществимый с точки зрения обеспеченности ресурсами и полученный с помощью эвристики «вначале выполняется самая короткая работа»; *в* — график продолжительностью 20 сут (оптимальный срок), осуществимый с точки зрения обеспеченности ресурсами.

ся самая короткая работа» дает лучшие, чем метод планирования ресурсов, результаты [41], которые весьма близки к календарному плану оптимальной продолжительности.

Эвристические процедуры календарного планирования при ограниченных ресурсах находят широкое применение. Одна из причин этого состоит в том, что они являются единственным способом получения календарных планов, осуществимых с точки зрения наличия ресурсов, в случае больших и сложных задач, часто встречающихся на практике. Другой причиной является то, что, хотя получаемые календарные планы не являются наилучшими, часто они оказываются достаточно хорошими для целей планирования, если учесть, что обычно фактическая продолжительность работ и потребность в ресурсах точно не известны. Кроме того, благодаря большой быстроте получения таких календарных планов с помощью ЭВМ подчас оказывается возможным проверить несколько различных эвристик и выбрать из них самую лучшую. Таким путем часто можно получить оптимальные или почти оптимальные результаты (хотя об этом редко бывает точно известно заранее).

Одним из крупных достижений в области сетевых методов было появление огромного числа эвристических стандартных машинных программ для календарного планирования с ограниченными ресурсами. Многие из них разработаны организациями для внутреннего пользования или для внешних пользователей с сохранением права собственности, поэтому в открытой литературе о них приводится очень мало сведений. Однако некоторые из них свободно продаются, и пользователям предоставляется подробная информация. Например, в табл. 4.19 представлен образец одной из 25 имеющихся в США таких программ, доступных на коммерческой основе, и даются ее основные характеристики. Эти программы позволяют составлять самые разнообразные отчеты для руководителей о наличии и потреблении ресурсов в виде таблиц или графиков.

4.16.2. ОПТИМАЛЬНЫЕ РЕШЕНИЯ

Второй широкий класс процедур составления календарных планов с ограничениями на ресурсы предназначен для получения оптимальных решений и находит сравнительно меньшее применение на практике, чем эвристические методы. Разработанные процедуры оптимизации можно разделить на две категории.

1. Процедуры, основанные на линейном программировании.
2. Процедуры, основанные на методе перебора и других математических методах.

Таблица 4.19. Характеристики некоторых имеющихся в продаже машинных программ, обеспечивающих составление сетевых моделей при ограниченных ресурсах

Название программы	Характеристики
CPM-RPSM (метод распределения и календарного планирования ресурсов), Совет по экономическим и промышленным исследованиям (США)	2000—8000 работ на проект, четыре вида ресурсов на проект, 26 переменных или постоянных ограничений на наличие ресурсов, допускается деление работ на отдельные части, допускаются ограничения на сроки начала и окончания работ. Используется постоянное эвристическое правило календарного планирования
MSCS (система календарного планирования и управления), фирма «Макдоннел отомейшн»	Возможность обработки данных по 25 проектам одновременно, 18 000 работ, 12 видов ресурсов на работу, множество гибких допущений об условиях выполнения работ, легкость корректировки данных. Программа обеспечивает определение затрат на проект и составление отчетов. Эвристики календарного планирования основаны на сложном методе функций приоритета и контролируются пользователем
PMS/360 (система управления проектом) фирма «Интернэшенл бизнес мэшинз»	Большая сложная информационная система управления, состоящая из четырех основных модулей (один из которых обеспечивает распределение ресурсов). Используются модели типа «дуга-работа» или графики предшествования работ. Одновременно могут обрабатываться данные для 225 проектов, 32 000 работ и 250 видов ресурсов. Обеспечивается возможность определения затрат, корректировки данных и составления отчетов. Имеется возможность выбора эвристики, устанавливающей последовательность работ
PPS IV (система планирования проектов), фирма «Контрол дэйта»	2000 работ на проект, 20 видов ресурсов на работу и проект, один или несколько проектов; допускается совмещение работ во времени; обеспечивается определение стоимости ресурсов и выдача сообщений о результатах. Выполняется также выравнивание потребления ресурсов при заданной продолжительности проекта. Могут быть заданы приоритеты для ресурсов, и допускается многосменная работа. Используется одна постоянная эвристическая процедура
PROJECT/2, фирма «Проджект софтвэа»	Допускается 50 различных проектов, 32 000 работ, несколько сотен видов ресурсов. Обеспечивается автоматическое построение сетевых графиков для повторяющихся последовательностей, осуществляется простая корректировка данных и имеются широкие возможности анализа затрат. Обеспечивается выбор пользователем эвристической процедуры установления последовательности. Обрабатываются данные для моделей типа «дуга-работа» и типа «узел-работа»

Линейное программирование впервые было разработано в начале 60-х годов как метод решения задач календарного планирования с ограничениями на ресурсы. Одним из авторов метода является Уист, который, однако, отметил нецелесообразность решения подобных задач этим методом. Так, для сети с 55 работами и четырьмя видами ресурсов требуется более 5000 уравнений и 1600 переменных. Впоследствии исследователи усовершенствовали формулировки Уиста и решили ряд задач методами целочисленного линейного программирования с помощью имеющихся машинных программ [16]. Однако решаемые задачи являются небольшими и обычно содержат менее 15 работ и до трех видов ресурсов. В последние годы появился ряд новых и интересных формулировок несколько более сложных задач календарного планирования с ограничениями на ресурсы [4, 13, 38]. Однако процедуры линейного программирования еще не разработаны в такой степени, чтобы их можно было применять для решения задач, легко решаемых эвристическими методами; сегодня они являются главным образом интересной темой исследований для теоретиков.

Ко второй категории процедур оптимизации относятся методы, появившиеся сравнительно недавно. Эти методы основаны на неявном переборе всех комбинаций последовательностей работ и включают такую процедуру оптимизации, как метод ветвей и границ. Впервые эти методы были применены для решения задач сетевого планирования с ограниченными ресурсами в 1967 г. Мюллером-Мербахом и Джонсоном [20]. Эти исследователи показали, что такие процедуры позволяют получить оптимальное решение для одноресурсной сети с 300 работами. В 1968 г. Дэвис [8] предложил несколько иную схему неявного перебора, которая впоследствии была использована в случае сетей, содержащих до 50 работ и до пяти видов ресурсов [10].

С середины 60-х годов сообщается о разработке многих новых методов, применяемых главным образом для календарного планирования при ограниченных ресурсах и стохастического анализа сетей. В основе всех практических систем, разработанных к настоящему времени, лежат методы эвристического календарного планирования, для которых имеются машинные программы, позволяющие получить достаточно хорошие календарные планы. Имеющиеся в настоящее время машинные программы позволяют составлять календарные планы для самых крупных проектов, осуществляемых почти при любых существующих условиях потребления и наличия ресурсов. Они применяются в таких разнообразных областях, как управление повторяющимися производственными процессами: календарное планирование строительных работ, планирование телевизионных передач и радиопрограм.

ЧАСТЬ III. СРАВНЕНИЕ ИМЕЮЩИХСЯ МАШИННЫХ ПРОГРАММ ДЛЯ РЕШЕНИЯ ЗАДАЧ С ПОМОЩЬЮ МКП И ПЕРТ[1)]

Результаты получены при участии
Ларри Смита, Международный университет шт. Флорида
Питера Малера, Университет Конкордиа, Монреаль, Канада

В других частях этой книги говорилось о некоторых машинных программах для решения задач с использованием описанных здесь методов. Мы не пытаемся разработать какую-либо программу для решения задач с помощью МКП и ПЕРТ, так как уже существует разнообразное математическое обеспечение, предоставляемое на коммерческой основе. В этом разделе дается сравнение имеющихся программ.

За последние 15 лет произошли крупные технические изменения в применении вычислительной техники для анализа сетей. В 60-е годы машинные программы разрабатывались в основном изготовителями аппаратного оборудования. Это было обусловлено тем, что изготовители аппаратного оборудования обладали необходимыми знаниями и опытом и что в то время отсутствовали специализированные организации. Число пользователей было невелико, так как большинство программ предназначалось для решения узких задач, и они были дорогостоящими.

В 70-е годы произошли значительные изменения в области программного обеспечения. Использование программ значительно упростилось. За последние несколько лет увеличилось число организаций, разрабатывающих программы для решения сетевых задач. Увеличение числа таких организаций вызвано главным образом ростом числа пользователей в промышленности. Не только крупные корпорации или правительственные организации могут пользоваться этими программами. Средние и мелкие фирмы считают, что экономически выгодно использовать их при планировании своей деятельности и управлении.

В 1968 г. фирма «Модер энд Филлипс» [29] опубликовала обзор машинных программ для решения задач с помощью ПЕРТ «Метод сетевого планирования и организации работ и ПЕРТ». В начале 1976 г. Вулперт из Станфордского университета составил неопубликованный обзор «Математическое обеспечение для планирования проектов и управления» [47]. В этом разделе дается сравнение 20 крупных программ. Были отобраны наиболее известные программы. Для облегчения сравнения составлена таблица характеристик, в которой указаны наиболее важные особенности программ.

[1)] Часть III перепечатывается из журнала *Industrial Engineering* [43] с разрешения авторов и Американского института организации производства.

Таблица 4.20. Обзор машинных программ для решения задач методами ПЕРТ и МКП

(1) *Возможности систем.* Указывается число работ или число подсетей, которые могут быть использованы

(2) *Сетевая модель.* Определяются три типа моделей — «дуга-работа» (AA), «узел-работа» (AN) и «узел-работа» с ограничениями на предшествование (Pre)

(3) *Календарные сроки.* Имеется внутренний календарь для планирования сроков выполнения работ проекта. Существует множество различных вариантов и разновидностей алгоритмов календарного планирования.

(4) *Модель Ганта или столбиковая диаграмма.* При желании может обеспечиваться графическое отображение выходных данных на временной шкале

(5) *Гибкий генератор программы выдачи на печать результатов анализа данных.* В определенных пределах пользователь может задавать формат выходных данных

(6) *Уточнение данных.* Программа принимает исправленные оценки продолжительности и сроки завершения работ и заново вычисляет уточненный календарный график

(7) *Контроль затрат.* В программу вводятся данные о выделенных средствах на каждую работу и фактических затратах, и при каждом прогоне программы суммируются выделенные средства и фактические затраты. Основной целью является оказание помощи руководителям в составлении реального плана, прежде чем начнется осуществление проекта, и в контроле над затратами на проект в ходе выполнения работ

(8) *Определение компромиссного соотношения между затратами и продолжительностью.* Программы с такими возможностями позволяют сокращать продолжительность проекта при минимальном увеличении прямых затрат на проект за счет выполнения работ, лежащих на критическом пути, в сжатые сроки. Указывается размер штрафа, взыскиваемого в случае задержки проекта

(9) *Установленные сроки.* Для целей планирования и контроля устанавливаются сроки завершения каждой работы. При выполнении вычислений эти сроки рассматриваются как ограничения

(10) *Сортировка.* Программа составляет список работ в последовательности, заданной пользователем

(11) *Распределение ресурсов.* С помощью одного из многих эвристических алгоритмов программа обеспечивает оптимальное распределение ресурсов

(12) *Система хранения и корректировки данных.* Программа обеспечивает хранение данных и автоматическую корректировку всех внутренних записей по мере поступления или образования данных

(13) *Требования в ЭВМ.* Это минимальные требования к емкости запоминающего устройства (в килобайтах) для хранения программы

(14) *Стоимость.* Указано, продается или сдается в аренду определенная программа и приводится ее покупная цена или стоимость аренды (если эти данные имеются)

				1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	«Астра», ПЕРТ-стоимость и ПЕРТ-время	От 1 920 до 10 000	AA			X	L			X	X		L	X	X	От 36 до 148	
B	«Си-Пи Эм систем» [1]	500	AA			X	X		X		X		L		X	16	
C	«Дэйталайн»	1100	AA			L		L					L			130	
D	«Фаснет»	2 500	AA AN			L			X	X	X		X				5 долл. за сетевой график, содержащий до 100 работ
E	Система управления и контроля «Марк II»	От 10 000 до 30 000	AA			X	X	X	X	X		X	X	X		512	Покупная цена 1–75 000 долл., Аренда 180–1 125 долл.
F	«Мини-ПЕРТ»	200 / на подмножество	Pre			X	X		X	X		X	X			От 192 до 384	153 долл. за модуль
G	«Мистер»		AA			X	X		X	X	X	X	X	X			
H	MSCS система календар. планиров. и управления	Более 30 000	Pre AA AN			X	X	L	X	X		X	L	X		98	Аренда
I	«Оптима 1100»	1 500	AN AA			X	X	L	X	X		X	X	X	X		Аренда
J	II		AN			X	X	L	X	X	X	X	X	X	X	160	
K	ПЕРТ-время	8 000	AA			L			X				L			65	
L	PMS IV (система управления проектами)	Более 32 000	Pre AA AN			X	X	X	X	X		X	X	X	X	От 44 до 75	От 55 до 200 долл. за модуль
M	[2]	4 000	AA			L	X	X	X	X		X	X	X		16	
N	PPS IV (система планирования проекта)	4 000	AA			L	X	X	X	X				X		200	15 000 долл. или 600 долл. за модуль
O	«Прокон 3»		Pre			X	X			X		X	L			От 36 до 180	15 500 долл.
P	«Проджакс»	15 000	Pre AN AA			X	X	X	X	X		X	L	X	X	96	
Q	«Проджект-2»	32 767	Pre AA AN			X	X	L	X	X		X	X	X	X	190	
R	«Промис»		AA			X	X	X	X				X		X	22	
S	«Проуз»	10 000	AA			X	X	X	X	X	X	X	L	X	X	300	35% доп. плата
T	«Спред»	2 000	Pre			X	X			L		X	X				

[1] Система критического пути

[2] Анализ сетевого графика проекта

Пояснения к обзору программ

A Программа распределения ресурсов «Астра II», ПЕРТ-время, ПЕРТ-стоимость, фирма «Ханиуэлл информейшн системз», Монреаль, провинция Квебек, Канада

B Система «Метод критического пути», информационная служба фирмы «Дженерал электрик», Монреаль, провинция Квебек, Канада

C «Дэталайн», фирма «Дэйталайн», Торонто, провинция Онтарио, Канада

D «Фаснет», фирма «Юниверсити компьютинг», Даллас, шт. Техас, США

E Система управления и контроля «Марк II», Система управления строительством, Хадфонфилд, шт. Нью-Джерси, США

F Мини-ПЕРТ, фирма «Интернэшенл бизнес мэшинз», Уайт-Плейнз, шт. Нью-Йорк, США

G «Мистер», фирма «Компьютер сайэнс», Эль-Сегундо, шт. Калифорния, США

H Система календарного планирования и управления, фирма «Макдоннел Дуглас отомейшн», Хьюстон, шт. Техас, США

I «Оптима 1100», фирма «Сперри Юнивак», Блу-Белл, шт. Пенсильвания, США

J PAC II, фирма «Интернэшенл системз», Кинг-оф-Пруссиа, шт. Пенсильвания, США

K ПЕРТ-время, фирма «Контрол дэйта», Миннеаполис, шт. Миннесота, США

L Система управления проектами PMS IV, фирма «Интернэшенл бизнес мэшинз», Уайт-Плейнз, шт. Нью-Йорк, США

M Система анализа сетевых графиков проектов PNA, фирма «Нэшенл кэш регистр», Дейтон, шт. Огайо, США

N Система планирования проектов PPS IV, фирма «Компьютинг энд информейшн сайэнсиз», Талса, шт. Оклахома, США

O «Прокон 3», фирма «Никол энд компани», Лос-Анджелес, шт. Калифорния, США

P Система анализа и контроля проектов PROJACS, фирма «Интернэшенл бизнес мэшинз», французское отделение, центр разработки программ, Париж, Франция

Q «Проджект-2», фирма «Проджект софтваре энд девелопмент», Кеймбридж, шт. Массачусетс, США

R Модуль «Промис-тайм», фирма «Баррафс бизнес мэшинз», Монреаль, провинция Квебек, Канада

S «Проуз», фирма «Конком» (консультационное обслуживание применения ЭВМ в строительстве), Монреаль, провинция Квебек, Канада

T «Спред», фирма «Компьютер сайэнс», Эль-Сегундо, шт. Калифорния, США

Таблица характеристик является удобным средством поиска и сравнения основных программ для решения задач с помощью МКП и ПЕРТ. Эта сводка характеристик была составлена на основе последних имеющихся данных и авторской интерпретации документов, предоставленных соответствующими фирмами. Однако в ней не делается попытка сравнивать эффективность отдельных характеристик различных программ. Поэтому одна программа может иметь лучшую, более гибкую и более эффективную стандартную подпрограмму сортировки, чем другая. С помощью таблицы этого установить нельзя.

Таблица характеристик (табл. 4.20) не требует пояснений. Буква X показывает, что в данной программе рассматриваемая характеристика присутствует в той или иной форме, а L обозначает ее ограниченное присутствие. Пустая ячейка означает, что рассматриваемая характеристика в программе отсутствует.

Сетевые модели, представленные в табл. 4.20, относятся к следующим типам: а) модель типа дуга-работа, б) модель типа узел-работа и в) модель типа узел-работа с ограничениями на отношения предшествования. Первые две модели, введенные в разд. 2.1.2, были подробно рассмотрены в разд. 4.4—4.10. Третья модель представляет собой обобщение второй модели и позволяет вводить задержки сроков начала и завершения работ. Более подробно с этой моделью читатель может познакомиться в книге Модера и Филлипса [29].

УПРАЖНЕНИЯ

1. Строительная фирма-подрядчик пытается составить план работ, связанных со строительством дома по заказу. Ниже в таблице приводятся данные о последовательности работ, отношениях предшествования и продолжительностях работ. Постройте соответствующую сетевую модель для последовательности работ, используя модель типа дуга-работа и модель типа узел-работа.

2. Рассмотрим проект по организации сбыта нового изделия. В таблице приводятся продолжительности работ, необходимых для выполнения проекта. Найдите минимальное время выполнения проекта.

Таблица к упражнению 1

Работа	Описание	Непосредственно предшествующие работы	Продолжительность, сут
a	Начало		0
b	Рытье котлована и заливка основания	*a*	4
c	Заливка бетонного фундамента	*b*	2
d	Сооружение деревянного каркаса, в том числе крыши	*c*	4
e	Выполнение кирпичной кладки	*d*	6
f	Укладка канализационных и водопроводных труб в подвальном помещении	*c*	1
g	Заливка пола подвального помещения	*f*	2
h	Установка водопроводных труб	*f*	3
i	Прокладка проводов	*d*	2
j	Установка отопления и вентиляции	*d, g*	4
k	Крепление штукатурных плит и штукатурные работы (в том числе высушивание)	*i, j, h*	10
l	Кладка покрытия пола	*k*	3
m	Установка кухонной арматуры	*l*	1
n	Завершение слесарно-водопроводных работ	*l*	2
o	Завершение плотницких работ	*l*	3
p	Кровельные работы и нанесение гидроизоляции	*e*	2
q	Крепление водосточных желобов и водост. труб	*p*	1
r	Кладка коллектора ливневых вод	*c*	1
s	Циклевание и покрытие полов лаком	*o, t*	2
t	Покраска	*m, n*	3
u	Завершение установки электрооборудования	*t*	1
v	Земляные работы	*q, r*	2
w	Заливка пешеходных дорожек и благоустройство территории	*v*	5
x	Окончание	*s, u, w*	0

Таблица к упражнению 2

Номер	Работа	Продолжительность, нед	Предшествующие работы
0	Планирование работ	3	—
1	Составление учебного плана	6	0
2	Отбор слушателей	4	0
3	Подготовка брошюры	3	0
4	Проведение учебных занятий	1	1, 2, 3
5	Поставка образцов продукции	4	0
6	Печатание брошюры	5	3
7	Подготовка рекламных материалов	5	0
8	Выпуск рекламных материалов	1	7
9	Распространение брошюры	2	6

3. Некоторый проект состоит из работ, указанных в таблице. Для каждой работы приводятся оптимистическая (a), наиболее вероятная (m) и пессимистическая (b) оценки продолжительности каждой операции (в сутках). Найдите критический путь для этой системы ПЕРТ.

Работа	Продолжительность, сут		
	a	m	b
(1, 2)	5	8	10
(1, 3)	18	20	22
(1, 4)	26	33	40
(2, 5)	16	18	20
(2, 6)	15	20	25
(3, 6)	6	9	12
(4, 7)	7	10	12
(5, 7)	5	7	8
(6, 7)	3	4	5

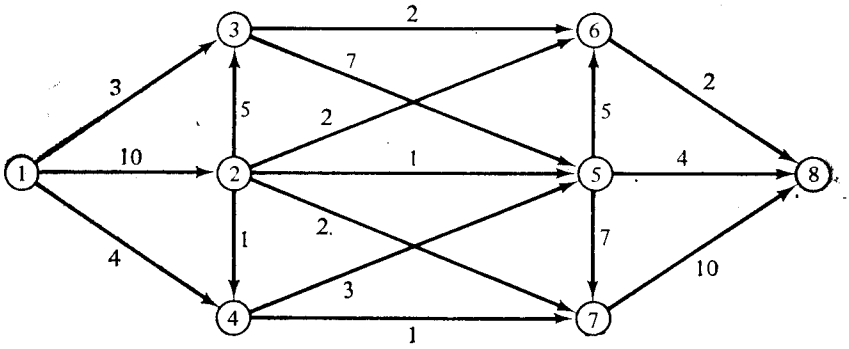
4. В таблице приводятся возможные продолжительности работ некоторого проекта. Эти данные представляют собой оптимистическую (a), наиболее вероятную (m) и пессимистическую (b) оценки продолжительности. Запланированная продолжительность проекта составляет 17,5 сут. Найдите вероятность выполнения проекта в установленный срок.

Работа	Продолжительность, сут		
	a	m	b
(1, 2)	6	8	10
(1, 3)	4	6	7
(1, 4)	4	8	12
(2, 5)	5	6	8
(3, 5)	7	8	9
(4, 6)	7	10	14
(5, 6)	3	4	5

5. Каковы три назначения фиктивных работ и фиктивных событий в модели типа дуга-работа. Проиллюстрируйте их применение на примерах.

6. Фундамент здания больницы состоит из четырех последовательно сооружаемых секций. Для сооружения каждой секции необходимо выполнение таких работ, как рытье котлована, монтаж арматуры и заливка бетоном. Рытье котлована для какой-либо одной секции не может начинаться до завершения этой работы для предыдущей секции. Это же относится и к заливке бетоном. После того как все котлованы вырыты, могут начинаться слесарно-водопроводные работы, но до заливки бетона можно выполнить только 15% этой работы. После подготовки фундамента каждой секции можно начинать выполнение еще 10% слесарно-водопроводных работ, если выполнены предыдущие 15% работы. Постройте сетевую модель для этого проекта.

7. Рассмотрим представленную сетевую модель. Допустим, что наиболее поздний допустимый срок завершения проекта составляет 49 сут. $[T_8(L) \neq T_8(E)]$. Найдите критический путь для этой сети.



8. Вычислите общий, свободный, гарантированный и независимый резервы времени для каждой работы из упражнения 2.

9. Рассмотрим проект по организации сбыта нового изделия. Вероятностные оценки продолжительностей работ приводятся в таблице.

Предшествующие работы	Номер	Работа	Продолжительность, недели: Оптимальная оценка, a	Наиболее вероятная оценка, m	Пессимистическая оценка, b
Нет	0	Планирование работ	2	3	5
0	1	Составление учебного плана	2	6	10
1	2	Отбор слушателей	3	4	5
0	3	Подготовка брошюры	1	3	4
9, 10, 5, и 8	4	Практическая проверка материалов	1	1	1
13	5	Поставка образцов продукции	3	4	4
3	6	Печатание брошюры	4	5	6
3	7	Подготовка рекламных материалов	2	5	7
7	8	Выпуск рекламных материалов	1	1	1
6	9	Распространение брошюры	2	2	3
6 и 2	10	Подготовка торговых работников	3	5	6
4	11	Обзор состояния рынка	2	4	5
0	12	Разработка опытного образца продукции	5	7	8
12	13	Изготовление образца продукции	2	3	4

а) Вычислите среднюю продолжительность и ее дисперсию для каждой работы.

б) Вычислите критический путь, используя модель типа узел-работа.

в) Вычислите критический путь, используя модель типа дуга-работа.

г) Какова вероятность того, что весь проект будет завершен менее чем за 30 сут? 40 сут? 50 сут?

д) Определите продолжительность проекта, вероятность превышения которой составляет всего 10%.

10. Для данных упражнения 1 определите: а) критический путь; б) общий резерв времени; в) свободный резерв времени; г) гарантированный резерв времени; д) независимый резерв времени.

11. Ниже в таблице дается упрощенное описание сети проекта в виде модели дуга-работа.

Задание	Работа	Продолжительность, сут
A	(1, 2)	8
B	(2, 3)	10
C	(2, 4)	2
D	(3, 4)	16
E	(3, 5)	4
F	(4, 5)	8
G	(3, 6)	7
H	(4, 6)	12
I	(5, 7)	3
J	(6, 7)	8
K	(7, 8)	2

а) Постройте сетевую модель с помощью МКП при такой структуре заданий.

б) Определите продолжительность проекта.

в) Полагая, что требуется только один человек для выполнения каждой работы, постройте гистограмму и график распределения ресурсов, показывающий потребление ресурсов во времени (для всех работ принят наиболее ранний возможный срок начала).

г) С помощью процедуры распределения ресурсов, описанной в разд. 4.14, обеспечьте минимальную суточную потребность в рабочей силе за все время работы над проектом.

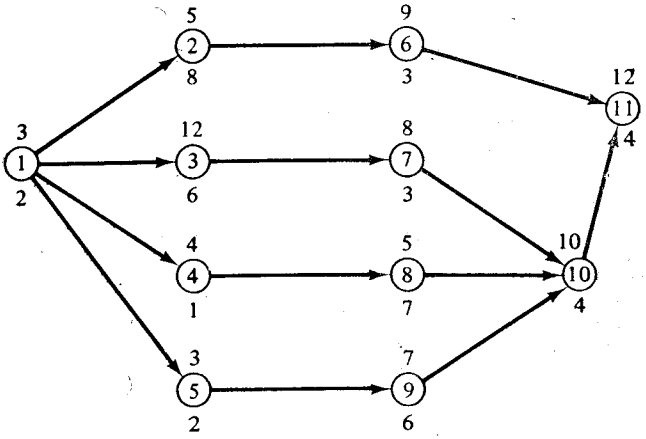
12. Для некоторого множества работ установлены следующие показатели.

Работа	Продолжительность, сут		Затраты, долл.		Предшествующие операции
	Нормальная	При сокращении сроков	Нормальная	При сокращении сроков	
A	7	4	95	100	Нет
B	6	3	90	97	Нет
C	5	4	86	104	Нет
D	7	7	92	98	*A* и *C*
E	6	5	87	93	*A*, *B*, и *C*
F	7	5	112	120	*B*
G	8	5	101	113	*F*
H	9	6	97	109	*K*, *E*, и *D*
I	12	10	95	100	*A* и *C*
J	10	7	100	110	Нет
K	9	8	105	114	*F*

а) Постройте сеть с помощью МКП, используя модели типа дуга-работа и узел-работа.

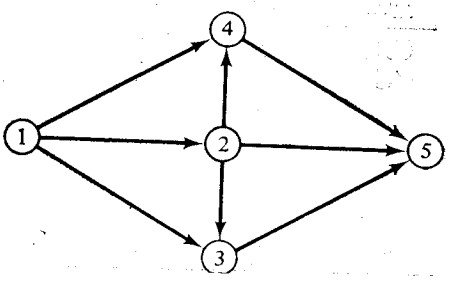
б) Найдите критический путь, используя модель типа узел-работа.
в) Найдите критический путь, используя модель типа дуга-работа.
г) Вновь постройте сетевую модель путем перехода от минимальных затрат к минимальным срокам. Постройте график для полученных результатов.

13. Ниже приводится сетевая модель типа узел-работа. Над узлом записана продолжительность работы в неделях, а под узлом — число работников, необходимое для выполнения работы.



а) Найдите критический путь для этой сетевой модели.
б) Вычислите полный, свободный, гарантированный и независимый резервы времени для всех работ.
в) Постройте гистограмму и график распределения ресурсов, показывающий потребление ресурсов за весь период осуществления проекта (для всех работ принят наиболее ранний возможный срок начала).
г) С помощью методов распределения ресурсов проведите сглаживание потребности в рабочей силе.

14. Рассмотрим следующую сетевую модель типа дуга-работа и соответствующие данные о затратах. Требуемым ресурсом являются денежные суммы в долларах, выделяемые на проект в сутки.



Работа	При нормальных условиях: Продолжительность, сут	При нормальных условиях: Затраты, долл.	При сокращении сроков: Продолжительность, сут	При сокращении сроков: Затраты, долл.	Требуемые ресурсы
(1, 4)	2	80	1	130	400
(1, 2)	3	70	1	190	370
(1, 3)	6	110	5	135	510
(2, 4)	4	60	3	100	200
(2, 5)	7	85	6	115	150
(2, 3)	2	90	1	100	300
(2, 5)	7	85	6	115	450
(3, 5)	3	50	2	70	400
(4, 5)	4	105	3	175	270

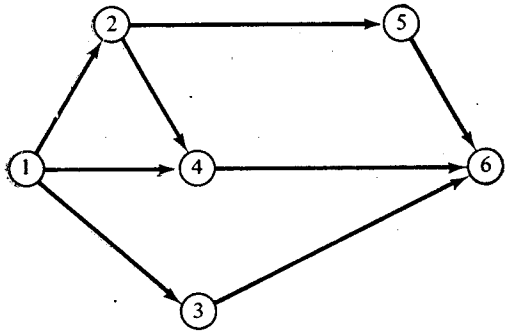
а) Вычислите критический путь.

б) Сократите продолжительность проекта до 9 сут при минимальных затратах.

в) Сократите продолжительность проекта до минимально возможного уровня и вычислите затраты.

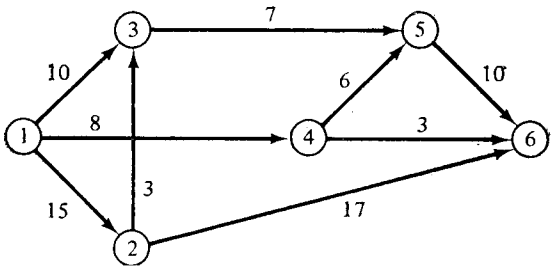
г) Используя результаты пункта а) с помощью методов распределения ресурсов обеспечьте более равномерное выделение капиталовложений на проект (установите уровень расходования денежных средств).

15. С помощью потокового алгоритма из разд. 4.12.1 найдите кривую зависимости между затратами и продолжительностью проекта, представленного следующей сетевой моделью.



Дуга	L_{ij}	U_{ij}	a_{ij}
(1, 2)	4	6	8
(1, 3)	4	8	9
(1, 4)	3	5	3
(2, 4)	3	3	—
(2, 5)	3	5	4
(3, 6)	8	12	20
(4, 6)	5	8	5
(5, 6)	6	6	—

16. С помощью сетевой модели, в которой продолжительности работ имеют нормальное распределение, определите такие значения продолжительности w_i, чтобы вероятность завершения каждой работы к моменту w_i была не меньше 0,95. Над дугами записаны средние значения, а все дисперсии равны 4. Постройте модель наступления событий с ограничениями на вероятность.



17. Рассмотрим сеть ПЕРТ с p независимыми путями, длины которых распределены по экспоненциальному закону. Средняя продолжительность пути равна $1/b$. Запишите формулу, выражающую через b и p вероятность того, что продолжительность проекта составит от 10 до 12 сут.

ЛИТЕРАТУРА

1. Baker B. N., Eris R. L., An Introduction to PERT-CPM, Richard D. Irwin, Inc., Homewood, Ill., 1964.
2. Benson L. A., Sewall R. F., Dynamic Crashing Keeps Projects Moving, *Computer Decisions* (February 1972).
3. Buffa E. S., Production-Inventory Systems: Planning and Control, Richard D. Irwin, Inc., Homewood, Ill., 1968.
4. Burton M. R., Some Mathematical Models for the Allocation of Limited Resources to Critical Path Type Scheduling, Unpublished dissertation, University of Illinois, Urbana, October 1967.
5. Crowston W. B. S., Decision CPM: Network Reduction and Solution, *Operational Research Quarterly* (1970).
6. Crowston W. B. S., Decision Network Planning Models, Management Sciences Research Report No. 138, Carnegie-Mellon University, Pittsburgh, Pa., 1968.
7. Davis E. W., Networks: Resource Allocation, *Journal of Industrial Engineering* (April 1974). Portions reproduced by permission of the authors and the American Institute of Industrial Engineers.
8. Davis E. W., An Exact Algorithm for the Multiple Constrained Resource Project Scheduling Problem, Unpublished dissertation, Yale University, New Haven, Conn., 1968.
9. Davis E. W. (частное сообщение), Fall 1979.
10. Davis E. W., Heidorn G. E., An Algorithm for Optimal Project Scheduling under Multiple Resource Constraints, *Management Science* (August 1971).
11. Davis E. W., Resource Allocation in Project Network Models — A Survey, *Journal of Industrial Engineering* (April 1966).
12. Evarts H. F., Introduction to PERT, Allyn and Bacon, Inc., Boston, 1967.
13. Fisher M. L., Optimal Solution of Resource Constrained Network Scheduling Problems, Technical Report No. 56, Operations Research Center, Massachusetts Institute of Technology, 1970.

14. Fondahl J. W., A Noncomputer Approach to the Critical Path Method for the Construction Industry, Construction Institute, Technical Report No. 9, Construction Institute, Stanford University, 1962.
15. Fulkerson D. R., A Network Flow Computation for Project Cost Curves, *Management Science* (January 1961).
16. Gomory R. E., Wade C. W., Write-up for Integer Programming Z, 7090, PK IPO2 and PK IPM2, IBM Corporation, Yorktown Heights, N. Y., 1961.
17. Hollander G., Integrated Project Control, *Project Management Quarterly* (April 1973).
18. Horowitz J., Critical Path Scheduling, The Ronald Press Co., New York, 1967.
19. ICP Software Directory, 2506 Willowbrook Parkway, Indianapolis, Ind.
20. Johnson T. J. R., An Algorithm for the Resource-constrained Project Scheduling Problem, Unpublished Ph. D. thesis, School of Management, Massachusetts Institute of Technology, 1967.
21. Kelley J. E., Walker M. R., Scheduling Activities to Satisfy Resource Constraints, in: Industrial Scheduling, by J. Muth, G. Thompson, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1963.
22. Kelley J. E., Walker M. R., Critical Path Planning and Scheduling, *Proceedings of the Eastern Joint Computer Conference* (1959).
23. Kelley J. E., Walker M. R., Critical Path Planning and Scheduling: Mathematical Basis, *Operations Research* (May — June 1961).
24. Levin R. I., Kirkpatrick C. A., Planning and Control with PERT/CPM, McGraw-Hill Book Co., New York, 1966.
25. Levy F. K., Thompson G. L., Wiest J. D., The ABC's of the Critical Path Method, *Harvard Business Review,* **41** (5) (September — October 1963).
26. Levy F. K., Wiest J. D., A Management Guide to PERT/CPM, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1969.
27. Lockyer K. G., An Introduction to Critical Path Analysis, Sir Isaac Pitman and Sons Ltd., London, 1967.
28. Miller R. W., Schedule, Cost and Profit Control with PERT, McGraw-Hill Book Co., New York, 1963.
29. Moder J. J., Phillips C. R., Project Management with CPM and PERT, Van Nostrand Reinhold Co., New York, 1964; 2nd ed., 1970. [Имеется перевод: Модер Дж., Филлипс С. Метод сетевого планирования и организации работ. — М — Л.: Энергия, 1966.]
30. Morris L. N., Critical Path Construction and Analysis, Pergamon Press, Inc., Elmsford, N. Y., 1967.
31. Muth J. F., Thompson G. L., eds., Industrial Scheduling, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1963.
32. NASA/DOD Guide, PERT/Cost, Office of the Secretary of Defense and National Aeronautics and Space Administration, Washington, D. C., 1964.
33. PERT-Summary Report Phase 1, Special Projects Office, Bureau of Ordinance, Dept. of the Navy, Washington, D. C., July 1958.
34. Phillips D. T., Hogg G. L., Maggard M. J., GERTS III or: A GERTS Simulator for Labor-limited Queueing Systems, Research Memo No. 72-2, Purdue University, 1972.
35. Pritsker A. A. B., Watters L. J., Wolfe P. M., Happ W., GERT: Graphical Evaluation and Review Technique, Part I, *Journal of Industrial Engineering* (1966).
36. Pritsker A. A. B., Watters L. J., Wolfe P. M., Whitehouse G. E., GERT: Graphical Evaluation and Review Technique, Part II, *Journal of Industrial Engineering* (1966).
37. Pritsker A. A. B., Watters L. J., Wolfe P. M., Burgess R., The GERT Simulation Programs: GERTS III, GERTS IIIQ, GERTS IIIC, GERTS IIIR, Department of Industrial Engineering, Virginia Polytechnic Institute, Blacksburg, Va.

38. Pritsker A. A. B., Watters L. J., Wolfe P. M., Multi-project Scheduling with Limited Resources: A Zero-One Programming Approach, *Management Science* (1969).
39. Riggs J. L., Heath C. O., Guide to Cost Reduction through Critical Path Scheduling, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1966.
40. Roper D. E., Critical Path Scheduling, *Journal of Industrial Engineering,* **15** (2) (March — April 1964).
41. Shaffer L. R., Ritter J. B., Meyer W. L., The Critical Path Method, McGraw-Hill Book Co., New York, 1964.
42. Siemens N., A Simple CPM Time-Cost Tradeoff Algorithm, *Management Science* (February 1971).
43. Smith L. A., Mahler P., Comparing Commercially Available CPM/PERT Computer Programs, *Industrial Engineering,* **10**, 4 (1978). Portions reproduced by permission of the author and the American Institute of Industrial Engineers.
44. Smith K. M., Critical Path Planning, Management Publications Ltd., London, 1971.
45. Thomas W. H., Four Float Measures for Critical Path Scheduling, *Industrial Engineering* (October 1969).
46. Viarisio G., Management Decision System Based on Decision CPM, Presented at INTERNET 72 Congress, Stockholm, 1972.
47. Woolpert B., Computer Software for Project Planning and Control, a paper for J. R. Freeland and C. A. Holloway, Stanford University, Graduate School of Business, May 1966.

Глава 5

НОВЫЕ ВОПРОСЫ

— Я могу попробовать, — сказал Петя.
— Тебе понадобится помощь, — сказал Саша.
— Тогда пойдем вместе, — сказал Петя.

Уолт Дисней. Петя и волк

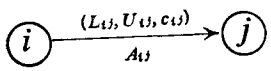
Как Пете вскоре стало ясно, преследование волка — трудная задача, и чтобы победить его, помощь Саши крайне необходима. Случай с Петей особенно примечателен для тех, кто серьезно занимается сетевым анализом — при изучении новых вопросов часто требуется «помощь» из других источников. Настоящая глава знакомит читателя с основными понятиями, необходимыми для изучения трех новых вопросов: 1) обобщенных сетей с выигрышами и проигрышами, 2) стохастических сетей и 3) многопродуктовых сетевых потоков. Каждая из этих трех задач представляет собой обобщение задач, рассмотренных в гл. 1—4, но имеет более широкое практическое применение.

ЧАСТЬ I. ОБОБЩЕННЫЕ СЕТИ. СЕТИ С ВЫИГРЫШАМИ И ПРОИГРЫШАМИ

Во всех алгоритмах, ранее описанных в данной книге, предполагалось, что если в промежуточный узел поступает несколько единиц потока, то столько же единиц потока выходит из этого узла. Это означает, что при прохождении потока по путям, соединяющим источники со стоками, не происходит ни образования новых, ни уничтожения старых единиц потока. Однако ясно, что существует широкий класс сетевых моделей, в которых требуется производить модификацию потока в сети, заключающуюся в его увеличении или уменьшении. Такими моделями являются модели водных бассейнов (испарение, выпадение осадков), модели, связанные с проектированием конструкций (изменение соотношения между давлением и силой), модели капиталовложений (прибыль, убытки), модели производства и распределения (отбраковка, отмена заказа), модели, связанные с составлением расписания работы бригад (прогулы) и расписания движения самолетов (ремонт самолетов), и многие другие.

Сети, при прохождении через дуги которых возможно увеличение или уменьшение потока, будем называть *обобщенными сетями.*

Более точно потоки, протекающие по дугам обобщенной сети, изменяются коэффициентами выигрыша (или проигрыша) таким образом, что поток, входящий в дугу, может отличаться от потока, выходящего из нее. Фактически для получения величины выходного потока величину входного потока следует домножить на коэффициент выигрыша (проигрыша). Рассмотрим ориентированную сеть, состоящую из дуг (i, j), причем из узла i в узел j протекает поток величины f_{ij}, а стоимость единицы потока равна c_{ij}. Допустимая величина потока по дуге (i, j) должна быть не меньше величины L_{ij} и не должна превосходить величину U_{ij}. Дуговой коэффициент будем обозначать через A_{ij}. Графически эти параметры дуг будем обозначать следующим образом:



Отметим, что при $A_{ij}=1$ мы приходим к традиционной сетевой постановке, при $A_{ij}>1$ поток увеличивается (выигрыш), при $A_{ij}<1$ поток уменьшается (проигрыш). Поскольку изменение потока происходит при его прохождении по дугам, то принцип сохранения потока по-прежнему имеет силу. Однако очевидно, что величины потока теперь могут принимать не только целые, но и любые положительные вещественные значения. Несмотря на то, что обобщенные сетевые задачи являются достаточно общими и имеют широкую область применения, ранее им не уделяли должного внимания. Лишь недавно были предприняты попытки разработки эффективных алгоритмов решения этих задач и написания соответствующих машинных программ для широкого пользования.

5.1. ПРИМЕНЕНИЕ ОБОБЩЕННЫХ СЕТЕЙ

Как уже отмечалось, с помощью обобщенных сетей могут быть решены многие задачи, не допускающие обычную сетевую интерпретацию. Это становится возможным благодаря тому, что дуговые множители имеют двоякий физический смысл. Во-первых, они могут отражать количественное изменение потока некоторого однородного продукта. Например, обобщенные сети позволяют моделировать такие процессы, как испарение, фильтрация, износ, расширенное воспроизводство, процентный прирост капитала, очистка сточных вод, ректификация с различной эффективностью, работа станков с различной производи-

тельностью, проектирование силовых конструкций. Во-вторых, дуговые множители могут отражать превращение одного типа продукта в другой. Это позволяет моделировать такие процессы, как обработка продукции, производство, превращение топлива в энергию, смешение, а также решать задачи, связанные с составлением расписания работы бригад, определением потребности в рабочей силе и обменом валюты. (Более подробно см. [4, 8, 9, 13].) Следующие примеры показывают практическое значение обобщенных сетей.

С помощью обобщенной сетевой модели Бомик [1] описал полную систему распределения водных запасов с учетом возможных потерь. В основном эта модель отражает процесс движения воды по каналам к различным резервуарам. Но при этом также рассматривается возможность задержания воды на некоторое время. Множители в данном случае учитывают уменьшение количества воды вследствие испарения и фильтрации.

Ким [19] использовал обобщенные сети для описания процесса очистки меди. Процедура электролитической очистки меди моделируется сложной сетью, дуги которой соответствуют линиям постоянного тока, а множители — величинам сопротивлений. С помощью такой модели можно исследовать влияние короткого замыкания на процесс очистки.

Чарнес и Купер [4] нашли применение обобщенным сетям как в изучении модели предела пластичности, так и в потоковой модели склад-фонды. В первом случае сеть строится с помощью уравнений равновесия для горизонтального и вертикального направлений с учетом перекрестного взаимодействия. Потоковая модель склад-фонды фактически является многопериодной моделью. Дуги здесь используются для описания производства и продажи продукции, пополнения ее запасов, а также осуществления денежных вкладов. Множители вводятся для того, чтобы учесть взаимозаменяемость товара и денег.

Крам [5] с помощью обобщенной сети описал модель контроля и регулирования денежных операций, которая позволяет многонациональной корпорации суммировать изменение цен, дебиторские задолженности, счета к оплате, денежные сборы, выплаты девидендов, уплаты процентов, лицензионные платежи и сборы на управление. Дуги представляют собой возможные варианты потока денег, а коэффициенты соответствуют затратам, накоплению, превращению в наличные деньги и обменному курсу. Кроме того, обобщенные сети находят применение при решении задач загрузки оборудования [4, 6, 28], различных задач о смешивании [4, 28], задач о поставщике [6, 28] и задач составления расписаний (например, задач о производстве и распределении, составлении расписания работы бригад, движения самолетов и задач подготовки кадров [4, 6, 28]).

5.2. ОБОБЩЕННАЯ СЕТЕВАЯ ЗАДАЧА КАК ЗАДАЧА ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Обобщенную сетевую задачу можно рассматривать как задачу о потоке минимальной стоимости в сети с выигрышами и проигрышами. Цель — добиться, чтобы в сток втекал поток величины F, имеющий минимальную стоимость, или чтобы из источника вытекал поток величины $\bar{F}$, имеющий минимальную стоимость. Мы опишем алгоритм, который позволяет решать обе эти задачи, однако для простоты изложения будем рассматривать первую из них. Задача о потоке минимальной стоимости в сети с выигрышами и проигрышами может быть сформулирована следующим образом:

$$\text{минимизировать} \sum_i \sum_j c_{ij} f_{ij} \tag{5.1}$$

при условии, что

$$\sum_j f_{sj} - \sum_i A_{js} f_{js} \leqslant \bar{F}, \tag{5.2}$$

$$\sum_j f_{ij} - \sum_i A_{ji} f_{ji} = 0, \quad i \neq s, t, \tag{5.3}$$

$$\sum_j f_{tj} - \sum_i A_{jt} f_{jt} = -F, \tag{5.4}$$

$$0 \leqslant f_{ij} \leqslant U_{ij} \quad \text{для всех} \quad (i, j). \tag{5.5}$$

Отметим несколько предположений, касающихся данной постановки.

1. Все дуговые множители являются положительными вещественными числами.

2. Нижние границы потоков по всем дугам равны нулю. Данное предположение не ограничивает общности рассматриваемой задачи, поскольку для дуг с положительными нижними границами можно произвести замену $\bar{f}_{ij} = f_{ij} - L_{ij}$.

3. Источники и стоки объединяются таким образом, что сеть содержит только один источник и один сток. Этого можно достигнуть, вводя главный источник и главный сток (см. описание метода дефекта).

4. В отличие от традиционных постановок входной поток $\bar{F}$ может отличаться от выходного потока F. Это происходит в силу того, что потоки по дугам изменяются в соответствии с дуговыми множителями. Таким образом, требуемый поток F будет определен для стока. Входные потоки не известны до тех пор, пока не найдено окончательное решение, однако на них можно наложить ограничение, при котором они не будут превосходить

величины $\overline{F}$. Отметим, что при данном предположении для входного потока $\overline{F}$ может не существовать выходного потока F.

Для решения данной задачи будут предложены два различных алгоритма. Первый из них используется для решения обобщенных сетевых задач частного вида, в то время как второй представляет нам общую схему решения. Перед тем как перейти к описанию этих алгоритмов, остановимся на некоторых основных свойствах обобщенных сетей.

5.3. ХАРАКТЕРИСТИКИ СЕТИ

Напомним, что множество ориентированных дуг, соединяющих источник и сток, называется *цепью,* или *ориентированной цепью*[1]. Произведение всех дуговых множителей цепи **E** будем называть изменением цепи и обозначать символом

$$C_E = \prod_{(i,\, j)\, \in E} A_{ij}. \tag{5.6}$$

Отметим, что если начальный и конечный узлы цепи совпадают, то она называется *циклом.* Произведение (5.6) дуговых множителей цикла будем называть *изменением цикла.*

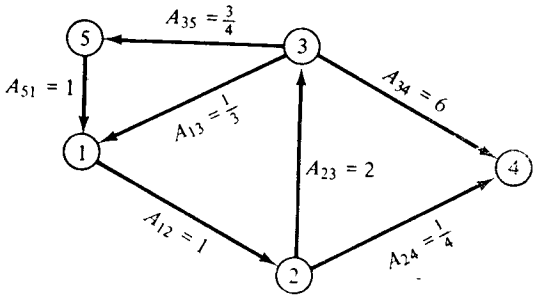


Рис. 5.1. Сеть с выигрышами и проигрышами.

Рассмотрим сеть, изображенную на рис. 5.1, вне связи с обобщенными сетевыми задачами. Цепь, соединяющая узлы 1 и 4, задается множеством дуг $E=\{(1, 2), (2, 3), (3, 4)\}$, а изменение цепи равно $C_E = 1 \cdot 2 \cdot 6 = 12$. Отметим, что если в дугу (1, 2) входит 1 единица потока, то в узел 2 также входит 1 единица; если входной поток по дуге (2, 3) равен 1, то в узел 3 поступают 2 единицы потока; если в дугу (3, 4) входят 3 едини-

[1] Более точное определение цепи дано в гл. 1. — *Прим. перев.*

цы потока, то в узел 4—12 единиц. Такое изменение потока по цепи называется *выигрышем цепи,* поскольку в результате прохождения по ней поток увеличивается. С другой стороны, величина потока, выходящего из узла 1 и протекающего по цепи $\mathbf{E}=\{(1, 2), (2, 4)\}$, в узле 4 уменьшится в 4 раза. Следовательно, изменение цепи $C_E=1\cdot{}^1/_4={}^1/_4$. Величина C_E в данном случае называется *проигрышем цепи.* Таким образом,

Изменение цепи

$$C_E=\prod_{(i,\,j)\,\in\mathbf{E}} A_{ij}\begin{cases}=1\text{—нет изменений,}\\ >1\text{—выигрыш цепи,}\\ <1\text{—проигрыш цепи.}\end{cases}$$

Отметим далее, что последовательность дуг $\mathbf{E}=\{(1, 2), (2, 3), (3, 1)\}$ образует цикл, для которого величина $C_E=1\cdot2\cdot{}^1/_3={}^2/_3$. Такой цикл называется *поглощающим циклом.* Если $\mathbf{E}=\{(1, 2), (2, 3), (3, 5), (5, 1)\}$, то $C_E=1\cdot2\cdot{}^3/_4\cdot1=1{,}5$. Такой цикл называется *генерирующим циклом.* Очевидно, что любой поток, входящий в узел 1 и проходящий по этому генерирующему циклу, увеличивается в 1,5 раза. Узел 1 будем называть *выпускающим,* поскольку через него может вытекать поток, протекающий по циклу. Действительно, любой поток F, поступающий в этот цикл, возрастает в C_E раз, где C_E — выигрыш цикла. В последующих разделах все эти понятия будут использованы для построения общего алгоритма поиска решения.

5.4. СЛУЧАЙ I. ОБОБЩЕННЫЕ СЕТИ, НЕ СОДЕРЖАЩИЕ ГЕНЕРИРУЮЩИХ И ПОГЛОЩАЮЩИХ ЦИКЛОВ

Если на дуговые множители сети не накладывать ограничения, согласно которому они должны быть меньше 1, то, вообще говоря, потоки минимальной стоимости из источника в сток могут не образовать простую ориентированную цепь. Труемпер [26] доказал следующую теорему.

Теорема. Любая обобщенная сетевая задача с произвольными дуговыми множителями, в которой изменение каждого цикла $\mathbf{E}$ равно $C_E=\prod_{(i,j)\in\mathbf{E}} A_{ij}=1$, может быть сведена к обычной сетевой задаче, в которой все дуговые множители $A_{ij}=1$.

Следствие. Для того чтобы данный результат имел место, необходимо и достаточно, чтобы для каждого узла $i=1, 2, \ldots, n$ существовало такое число M_i, что для всех дуг (i, j) выполняются равенства

$$M_iA_{ij}/M_j=1. \tag{5.7}$$

Следовательно, если в обобщенной сети выигрыш каждого цикла равен 1, то соответствующая обобщенная сетевая задача

может быть сведена к обычной задаче о потоке минимальной стоимости и решена с помощью любого подходящего метода. Для описания необходимых преобразований рассмотрим обобщенную и обычную сетевые задачи.

Обобщенная

Минимизировать $\sum_i \sum_j c_{ij} f_{ij}$ при условии, что

$$\sum_j f_{sj} - \sum_j A_{js} f_{js} \leqslant \overline{F},$$

$$\sum_j f_{ij} - \sum_j A_{ji} f_{ji} = 0,$$

$$\sum_j f_{tj} - \sum_j A_{jt} f_{jt} = -F,$$

$$L_{ij} \leqslant f_{ij} \leqslant U_{ij}.$$

Обычная

Минимизировать $\sum_i \sum_j c_{ij} f_{ij}$ при условии, что

$$\sum_j f_{sj} - \sum_j f_{js} \leqslant \overline{F},$$

$$\sum_j f_{ij} - \sum_j f_{ji} = 0,$$

$$\sum_j f_{tj} - \sum_j f_{jt} = -F,$$

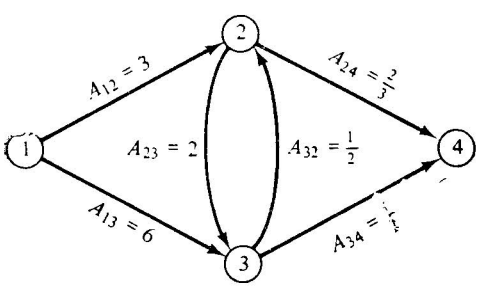
$$L_{ij} \leqslant f_{ij} \leqslant U_{ij}.$$

Если выполнено условие теоремы, то, домножив на $1/M_i$ ($i=1, 2, \ldots, n$) правую и левую части уравнений, выражающих принцип сохранения потока для обобщенной задачи, мы получим обычную сетевую задачу. Величины M_i вычисляются следующим образом:

Шаг 1. Положить $M_1=1$.

Шаг 2. Для каждой дуги (i, j) последовательно вычислить величины $M_j=M_iA_{ij}$, где $i=1, 2, \ldots, n$, а j принимает значения номеров всех узлов, соединенных дугой с узлом i.

5.4.1. ПРИМЕР



Дуга (i, j)	M_i	A_{ij}	M_j
(1, 2)	1	3	3
(1, 3)	1	6	6
(3, 4)	6	$\frac{1}{3}$	2

Следовательно, $M_1=1$, $M_2=3$, $M_3=6$, $M_4=2$.

Задача линейного программирования (обобщенная)

$$\text{Минимизировать} \sum_i \sum_j c_{ij} f_{ij}$$

При условии, что (1, 2)	(1, 3)	(2, 3)	(3, 2)	(2, 4)	(3, 4)	
$f_{12}+f_{13}$						$\leq \bar{F}$
$-f_{12}$		$+\frac{1}{3}f_{23}$	$-\frac{1}{6}f_{32}$	$+\frac{1}{3}f_{24}$		$=0$
	$-f_{13}$	$-\frac{1}{3}f_{23}$	$+\frac{1}{6}f_{32}$		$+\frac{1}{6}f_{34}$	$=0$
				$-\frac{1}{3}f_{24}$	$-\frac{1}{6}f_{34}$	$=F$

Отметим, что $1/M_1=1$, $1/M_2=1/3$, $1/M_3=1/6$, $1/M_4=1/2$. После умножения элементов i-й строки на $1/M_i$ $(i=1, 2, 3, 4)$ будет получена следующая задача.

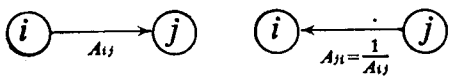
Задача линейного программирования (обычная)

$$\text{Минимизировать} \sum_i \sum_j c_{ij} f_{ij}$$

При условии, что (1, 2)	(1, 3)	(2, 3)	(3, 2)	(2, 4)	(3, 4)	
$f_{12}+f_{13}$						$\leq \bar{F}$
$-f_{12}$		$+\frac{1}{3}f_{23}$	$-\frac{1}{6}f_{32}$	$+\frac{1}{3}f_{24}$		$=0$
	$-f_{13}$	$-\frac{1}{3}f_{23}$	$+\frac{1}{6}f_{32}$		$+\frac{1}{6}f_{34}$	$=0$
				$-\frac{1}{3}f_{24}$	$-\frac{1}{6}f_{34}$	$=F$

Пусть $\bar{f}_{23}=1/3 f_{23}$, $\bar{f}_{32}=1/6 f_{32}$, $\bar{f}_{34}=1/6 f_{34}$. Тогда производя указанную замену переменных и изменяя соответствующим образом верхние и нижние границы потоков по дугам, мы получаем обычную сетевую задачу.

В заключение читателю предлагается проверить, что для каждого цикла **E** исходной сети $\prod_{(i,j)\in \mathbf{E}} M_i A_{ij}/M_j=1$. Отметим, что при изменении ориентации дуги значение соответствующего множителя заменяется на обратное, как это показано ниже:



5.5. СЛУЧАЙ II. ОБОБЩЕННЫЕ СЕТИ С ГЕНЕРИРУЮЩИМИ И (ИЛИ) ПОГЛОЩАЮЩИМИ ЦИКЛАМИ

В отличие от алгоритмов, рассмотренных ранее, алгоритмы решения общей сетевой задачи с выигрышами и проигрышами являются значительно более сложными. Их описания содержатся в работах Майника [22], Иттли и Прагера [24], Грайнолда [14], Йенсена и Барнеса [17], Глоувера и Клингмана [13], Джервиса и Джезайера [15], Джуэлла [18]. Алгоритм, описанный в настоящем разделе, принадлежит Йенсену и Бомику [16]. Единственное ограничение данного алгоритма заключается в том, что сеть не должна содержать циклов отрицательной стоимости. Алгоритм основан на использовании простой процедуры расстановки пометок, являющейся расширением основной процедуры пометок, применяемой в обычных алгоритмах поиска кратчайших цепей. Остановимся вначале на кратком описании трех шагов алгоритма, а затем дадим математическое описание каждого из них.

5.5.1. ШАГ 1. ПОСТРОЕНИЕ НАЧАЛЬНОГО ПОТОКА

а. Находится цепь минимальной стоимости, соединяющая источник и сток. Если такой цепи не существует, то задача не имеет решения.

б. Поток по цепи минимальной стоимости, соединяющей источник и сток, увеличивается до тех пор, пока хотя бы одна дуга не будет насыщена.

5.5.2. ШАГ 2. ПОСТРОЕНИЕ МАРГИНАЛЬНОЙ СЕТИ

После того как поток будет направлен по цепи, ведущей из источника в сток, как это описано на шаге 1, исходная сеть модифицируется с тем, чтобы отобразить состояние текущего потока и определить, возможно ли дальнейшее изменение потока в сети. Модифицированную сеть будем называть *маргинальной сетью*. Она содержит все ненасыщенные дуги исходной сети и по одной «зеркальной дуге» для каждой исходной дуги, поток по которой строго положителен.

5.5.3. ШАГ 3. ПРОЦЕСС УВЕЛИЧЕНИЯ ПОТОКА

Используя маргинальную сеть, построенную на шаге 2, определяется новый аугментальный путь потока из источника в сток и находится протекающий по этому пути максимальный поток минимальной стоимости. По окончании выполнения шага 3 либо будет найден искомый поток, либо повторяются шаги 1 и 2. Перейдем к подробному описанию шагов алгоритма, основываясь на работе Йенсена и Бомика [16].

5.6. ШАГ 1. АУГМЕНТАЛЬНАЯ ЦЕПЬ ПОТОКА МИНИМАЛЬНОЙ СТОИМОСТИ

Для обобщенной сети цепь минимальной стоимости определяется как цепь, по которой стоимость доставки потока из источника в сток является минимально возможной. Как показано

Джуэллом [18], сети с выигрышами (дуговые множители больше 1) могут содержать генерирующие циклы. Хотя первоначально сеть может не содержать дуг с множителями, большими 1, но в результате выполнения последовательных итераций алгоритма строятся зеркальные дуги с коэффициентами выигрыша, большими 1. Зеркальная дуга определяется как дуга, направление потока по которой противоположно направлению потока по «исходной» дуге, а множитель которой равен обратной величине исходного множителя. Таким образом, в общем случае цепью минимальной стоимости может быть либо простая ориентированная цепь из источника в сток, либо составная цепь, содержащая генерирующий цикл.

Для построения цепи минимальной стоимости необходимо для каждого узла i сети определить новую переменную V_i, которую будем называть *потенциалом узла*. Экономический смысл этого понятия состоит в том, что потенциал узла равен стоимости получения единицы потока из этого узла. В ориентированной цепи потенциал узла равен минимальной стоимости транспортировки единицы потока из источника в рассматриваемый узел. В цикле он соответствует стоимости генерации единицы потока в данном узле. Потенциал узла зависит от стоимостей и дуговых множителей цепи, по которой поток поступает в данный узел. Если стоимость доставки единицы потока в узел i равна V_i, то стоимость доставки единицы потока в узел j через дугу (i, j) равна

$$\overline{V}_{ij}=(V_i+c_{ij})/A_{ij}, \tag{5.8а}$$

где c_{ij} — стоимость пересылки единицы потока из узла i в узел j, а A_{ij} — множитель дуги (i, j).

Вычислив по формуле (5.8а) значения $\overline{V}_{ij}$ для всех дуг, входящих в узел j, можно определить потенциал данного узла:

$$[V_j=\min_{i\in\beta_j}[\overline{V}_{ij}]=\min_{i\in\beta_j}[(V_i+c_{ij})/A_{ij}], \tag{5.8б}$$

где β_j — это множество узлов, непосредственно связанных с узлом j дугами, ведущими из i в j. Соотношение (5.8б) может быть также записано в виде

$$V_j\leqslant(V_i+c_{ij})/A_{ij} \text{ для всех } i\in\beta_j, \tag{5.8в}$$

причем по крайней мере на одной дуге, ведущей в узел j, должно достигаться равенство. Если существует только одна дуга, ведущая из i в j, то потенциал узла j может быть вычислен непосредственно из соотношения (5.8а), поскольку в этом случае $V_j=\overline{V}_{ij}$, т. е.

$$V_j=(V_i+c_{ij})/A_{ij}. \tag{5.8г}$$

Полагая вначале потенциал источника равным нулю ($V_s=0$), а потенциалы всех остальных узлов равными бесконечности ($V_i=\infty$, $i\neq s$), можно значения потенциалов узлов итеративно уменьшать до тех пор, пока неравенство (5.8в) не будет выполняться для каждой дуги. Данный процесс выполняется с помощью процедуры анализа дуг. Для каждой дуги (i, j) проверяется выполнение неравенства (5.8в). Если оно выполнено, то никаких действий не производится. В противном случае потенциал V_j узла принимается равным $(V_i+c_{ij})/A_{ij}$. Данный процесс продолжается до тех пор, пока не будут проанализированы все дуги. Поскольку потенциалы узлов изменяются, каждая дуга может анализироваться более одного раза. Процедура анализа завершается, если ни один из потенциалов не может быть изменен. Хотя дуги можно анализировать в произвольном порядке, удобно начинать с источника и последовательно перебирать прямые дуги, проверяя для каждой из них неравенство (5.8в) и изменяя в случае необходимости потенциалы узлов.

Для трассировки цепи минимальной стоимости может быть использована любая процедура обратного поиска. Для обозначения узла i, на котором в выражении (5.8б) достигается равенство $V_j=\overline{V}_{ij}$, будем использовать обратный указатель p_j. Вначале значения всех обратных указателей полагаются равными нулю. Если для узла j $V_j>(V_i+c_{ij})/A_{ij}$, то p_j полагается равным i, а V_j, как уже отмечалось, полагается равным $(V_i+c_{ij})/A_{ij}$. Таким образом, в результате работы алгоритма каждому узлу i будет приписана пометка (p_i, V_i), где V_i — это минимальная стоимость получения единицы потока из данного узла, а p_i — узел, непосредственно предшествующий рассматриваемому узлу в цепи минимальной стоимости. Если стоку приписывается пометка $(0, \infty)$, то это означает, что ни одна из аугментальных цепей потока не входит в сток сети, и алгоритм завершает свою работу.

5.7. ШАГ 2. ПОСТРОЕНИЕ МАРГИНАЛЬНОЙ СЕТИ

После того как на шаге 1 был построен начальный поток в сети, вычисляются остаточные пропускные способности дуг, равные разности между их исходными пропускными способностями и текущими значениями потоков по дугам. В результате модификации потоков в исходной сети эти остаточные пропускные способности в дальнейшем могут быть изменены (увеличены или уменьшены). Для того чтобы произвести такие изменения, исходная сеть преобразуется в новую, называемую *маргинальной сетью*. Как отмечалось выше, маргинальная сеть содержит все ненасыщенные дуги исходной сети и по одной зеркальной дуге для каждой исходной дуги с положительным

потоком. Для дуги (i, j) соответствующая ей зеркальная дуга будет ориентирована от j к i и поэтому при наложении этих двух дуг друг на друга поток по исходной дуге уменьшится. Дуговые множители зеркальных дуг равны обратным величинам множителей соответствующих исходных дуг, а потоки по зеркальным дугам в их конечных узлах равны величинам, на которые уменьшаются потоки по соответствующим исходным дугам.

Если f_{ij} — поток по дуге (i, j), то $U_{ij}—f_{ij}$ — остаточная пропускная способность этой дуги в маргинальной сети. Если A_{ij} — коэффициент выигрыша этой дуги, то $f_{ij}A_{ij}$ — максимально возможный выходной поток дуги (i, j). Поскольку указанная редукция выполняется с использованием зеркальной дуги (j, i), то пропускная способность этой дуги в маргинальной сети должна определяться равной $f_{ij}A_{ij}$. Отметим, что коэффициент выигрыша этой дуги равен $1/A_{ij}$, а поток по ней уменьшается, причем если в узле j он будет равен $f_{ij}A_{ij}$, то в узле i — f_{ij}. Поскольку за прохождение единицы потока по дуге (i, j) уже взималась плата, равная c_{ij}, а поток по этой дуге уменьшается, то для возмещения этих расходов необходимо назначить стоимость прохождения единицы потока по дуге (j, i), равной $-c_{ij}/A_{ij}$. На каждой итерации зеркальные дуги строятся только для тех исходных дуг, по которым протекает некоторый поток. Исходные насыщенные дуги не включаются в маргинальную сеть. (Отметим, что зеркальные дуги тем не менее относятся к насыщенным дугам.)

Таким образом, если f_{ij} — новые величины потоков в исходной сети, то для каждой дуги $(i, j)^*$ маргинальной сети определяются следующие параметры:

$$
\begin{aligned}
c_{ij}^* &= c_{ij}, && \text{если} && f_{ij} < U_{ij},\\
c_{ji}^* &= -c_{ij}/A_{ij}, && \text{если} && f_{ij} > 0,\\
U_{ij}^* &= U_{ij} - f_{ij}, && \text{если} && f_{ij} \geqslant 0,\\
U_{ji}^* &= f_{ij}A_{ij}, && \text{если} && f_{ij} > 0,\\
A_{ij}^* &= A_{ij}, && \text{если} && f_{ij} < U_{ij},\\
A_{ji}^* &= 1/A_{ij}, && \text{если} && f_{ij} > 0.
\end{aligned}
$$

5.8. УВЕЛИЧЕНИЕ ПОТОКА

После того как в обобщенной сети была найдена цепь минимальной стоимости, по этой цепи необходимо пустить максимальный поток. Это можно сделать несколькими способами. Один из них заключается в простом увеличении потока за отдельный проход по цепи минимальной стоимости после того, как она была определена. Величины потоков вычисляются с по-

мощью процедуры возврата из стока в источник, при выполнении которой дугам приписываются соответствующие потоки, удовлетворяющие заданным ограничениям.

Понятие цепи минимальной стоимости непосредственно связано с понятием базиса, используемым в линейном программировании [32]. Кроме того, по своему смыслу понятие потенциала узла очень близко к понятию двойственной переменной, вводимой в алгоритме дефекта [16].

Как отмечалось выше, если на коэффициенты выигрыша дуг сети не наложено ограничение равенства их 1, то цепь минимальной стоимости может не являться простой ориентированной цепью из источника в сток. Она может содержать генерирующий цикл, не содержащий источник. Генерирующие циклы легко распознаются алгоритмически. Если при выполнении процедуры анализа дуг узлу i вначале была приписана пометка (p_i, V_i), а затем новая пометка $(p_i, \bar{V}_{ij})$, $\bar{V}_{ij} < V_i$, то обнаружен цикл. В этом случае выполнение процедуры анализа дуг повторится и потенциалы всех узлов цикла вновь уменьшатся. Йенсен и Бомик [16] показали, что при непрерывном повторении данного процесса потенциалы узлов цикла, уменьшаясь, сходятся к некоторому числу.

В дальнейшем нам понадобится следующее обозначение. Пусть $\mathbf{N}_C$ — множество узлов цикла, а $\mathbf{A}_C$ — множество его дуг. Пусть d_r и a_r — стоимость единицы потока по дуге $r \in \mathbf{A}_C$ и множитель этой дуги соответственно. При построении цикла и определении потенциалов его узлов рекуррентные соотношения (5.8а) и (5.8б) не могут быть использованы непосредственно. Пусть h — это узел из $\mathbf{N}_C$, непосредственно связанный с простой цепью, соединяющей генерирующий цикл $L = (\mathbf{N}_C, \mathbf{A}_C)$ со стоком t. Если $t \in \mathbf{N}_C$, то $h = t$. Определим h как *выпускающий узел*. Его потенциал вычисляется по формуле

$$V_h = \frac{g}{g-1} \sum_{j=1}^{n_C} \left[d_j \Big/ \prod_{i=j}^{n_C} a_i \right], \qquad (5.9)$$

где g — коэффициент выигрыша цикла, а j — номера элементов множества $\mathbf{A}_C$. Предполагается, что дугам цикла приписаны номера 1, 2, ..., n_C, причем нумерация осуществляется последовательно, начиная с дуги с начальным узлом h. После того как будет вычислена величина V_h, потенциал любого другого узла j цикла может быть найден из потенциала предшествующего узла i в результате последовательного применения соотношения (5.8г). Если выпускающий узел h не является стоком t, то пометки узлов, соединяющих h с t, могут быть легко найдены с помощью соотношения (5.8г).

В качестве примера возьмем генерирующий цикл, изображенный на рис. 5.2, и опишем для него работу рассмотренной процедуры.

Множества узлов и дуг данного цикла определяются следующим образом: $\mathbf{N}_C=\{2, 4, 3\}$, $\mathbf{A}_C=\{(2, 4), (4, 3), 3, 2)\}$. Пронумеруем дуги из множества $\mathbf{A}_C$. Будем предполагать, что дуга (2, 4) является первой дугой цикла, дуга (4, 3) — второй и дуга (3, 2) — третьей. Соответствующими параметрами этих дуг являются величины $d_1=c_{24}$, $d_2=c_{43}$, $d_3=c_{32}$, $a_1=A_{24}$, $a_2=A_{43}$ и $a_3=A_{32}$. Предполагается, что коэффициент выигрыша цикла, равный $g=a_1a_2a_3$, строго больше 1. Из (5.9) определяем потенциал узла 2 как $V_2=[d_1/a_1a_2a_3+ + d_2/a_2a_3 + d_3/a_3]\,[a_1a_2a_3/(a_1a_2\times \times a_3-1)]$. Аналогично, выбирая в качестве выпускающего узел 4 или узел 3, с помощью (5.9) можно вычислить величины V_4 и V_3. Другой способ вычисления потенциалов заключается в следующем. Вначале из (5.9) определяется потенциал выпускающего узла, а затем с помощью (5.8г) вычисляются потенциалы остальных узлов.

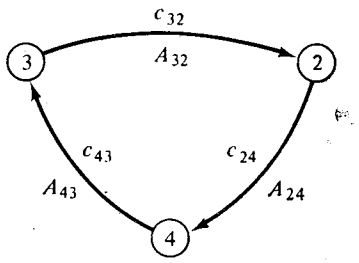
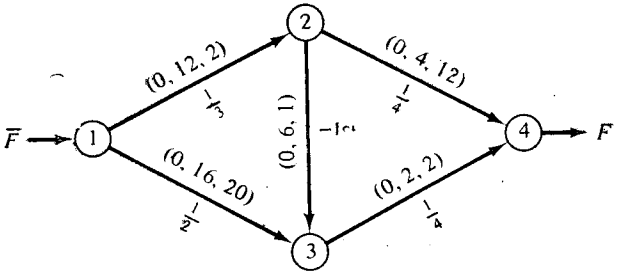


Рис. 5.2. Генерирующий цикл.

5.9. ПРИМЕР ОБОБЩЕННОЙ СЕТЕВОЙ ЗАДАЧИ

В настоящем разделе будет показана работа алгоритма расстановки пометок для сетей с выигрышами и проигрышами на примере, заимствованном у Йенсена и Бомика [16]. Задача будет решена в четыре итерации.

Итерация 1. На последующих итерациях алгоритма изображенная ниже сеть будет называться исходной сетью.



В следующей таблице приводятся результаты применения формулы (5.8б) к каждой дуге исходной сети.

Узел, j	Дуга, входящая в узел	c_{ij}	A_{ij}	V_i	$\bar{V}_{ij}$	V_j	Пометка
1	—	—	—	0	0	0	(0, 0)
2	(1, 2)	2	1/3	0	6	6	(1, 6)
3	(1, 3)	20	1/2	0	40		
	(2, 3)	1	1/2	6	14	14	(2, 14)
4	(2, 4)	12	1/4	6	72		
	(3, 4)	2	1/4	14	64	64	(3, 64)

Цепь минимальной стоимости состоит из дуг (1, 2), (2, 3) и (3, 4).

Величина максимально возможного потока из источника в сток, протекающего по цепи минимальной стоимости, является функцией как дуговых множителей, так и верхних границ потоков по этим дугам. Можно показать, что величина максимального потока в стоке t равна

$$F_t = \min_{1 \leqslant k \leqslant m} \left[U_k \prod_{r=k}^{m} a_r \right], \tag{5.10а}$$

где m — число дуг в цепи, U_k — верхняя граница потока по k-й дуге цепи, a_k — множитель k-й дуги. Формулу (5.10а) можно упростить, вводя обозначение

$$\bar{F}_k = U_k \sum_{r=k}^{m} a_r. \tag{5.10б}$$

В этом случае

$$F_t = \min_{1 \leqslant k \leqslant m} [\bar{F}_k]. \tag{5.10в}$$

После того как вычислено значение F_t, с помощью обратной рекурсии, начиная со стока, определяются потоки по дугам цепи минимальной стоимости. В следующей таблице приводятся результаты, полученные на последнем шаге итерации 1.

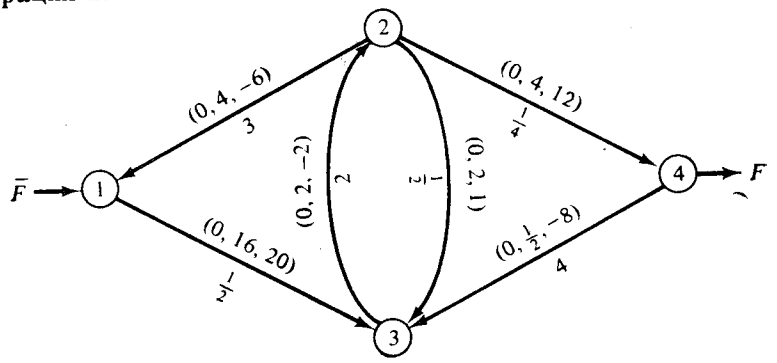
$m = 3$

Дуга (i, j)	k	U_k	$\prod_{r=k}^{m} a_r$	$\bar{F}_k$
(1, 2)	1	12	1/24	1/2
(2, 3)	2	6	1/8	3/4
(3, 4)	3	2	1/4	1/2

В силу (5.10в) $F_t = \min[1/2, 3/4, 1/2] = 1/2$. Обозначим через $f_{ij}^{(I)}$ поток по дуге (i, j) на итерации I. Двигаясь по цепи мини-

мальной стоимости от стока к источнику, получаем следующие результаты для $l=1: f_{34}^{(1)}=(1/2)/(1/4)=2$, $f_{23}^{(1)}=2/(1/2)=4$, $f_{12}^{(1)}=4/(1/3)=12$, $f_{13}^{(1)}=f^{(1)}{}_{24}=0$. Следовательно, $F=1/2$, $\bar{F}=12$, а поток величиной 1/2 генерируется за стоимость, равную $1/2\times\times 64=32$. Дуги (1, 2) и (3, 4) являются насыщенными и поэтому исключаются из исходной сети. Зеркальные дуги (2, 1), (3, 2) и (4, 3) включаются в маргинальную сеть для того, чтобы выяснить, возможно ли уменьшение потоков по соответствующим исходным дугам.

Итерация 2.



На итерации 1 узлу 1 приписывается пометка (0, 0). Ниже приводятся результаты вычислений, полученные с помощью рекуррентного соотношения (5.8б). При этом используются значение $V_1=0$ и параметры маргинальной сети.

Узел j	Дуга, входящая в узел j	c_{ij}	A_{ij}	V_i	$\bar{V}_{ij}$	V_j	Пометка
3	(1, 3)	20	1/2	0	40	—	
2	(3, 2)	−2	2	40	19	19	(3, 19)
1	(2, 1)	−6	3	19	13/3	0	(0, 0)
3	(2, 3)	1	1/2	19	40	—	
4	(2, 4)	12	1/4	19	124	124	(2, 124)
3	(4, 3)	−8	4	124	29	29	(4, 29)

С помощью этих результатов определяется генерирующий цикл. Он состоит из дуг (3, 2), (2, 4) и (4, 3) (рис. 5.3). Коэффициент выигрыша этого цикла равен 2 и поэтому $g/(g-1)=2$. Рассматривая узел 4 как выпускающий и используя (5.9), вычисляем значение V_4:

$$V_4=2\left[\frac{-8}{4\cdot 2\cdot 1/4}+\frac{-2}{2\cdot 1/4}+\frac{12}{1/4}\right]=80.$$

Значения V_3 и V_2 могут быть вычислены с помощью (5.8г): $V_3=(V_4+c_{43})/A_{43}=(80-8)/4=18$, $V_2=(V_3+c_{32})/A_{32}=(18-2)/2=8$. Узлам 3, 2 и 4 цикла, изображенного на рис. 5.3, приписываются новые пометки (4, 18), (3, 8) и (2, 80) соответственно.

Поскольку узел 4 является стоком, то увеличение потока осуществляется с минимальными затратами в том случае, если этот узел выбрать в качестве выпускающего узла. При этом может быть получен поток, стоимость единицы которого равна 80. Как и на итерации 1, максимальный поток, который может быть получен в выпускающем узле, зависит от дуговых множителей и верхних границ потоков по дугам. Однако максимальный поток в цикле определяется коэффициентом выигрыша цикла. Величина максимального потока, порождаемого циклом в выпускающем узле, вычисляется следующим образом [16]:

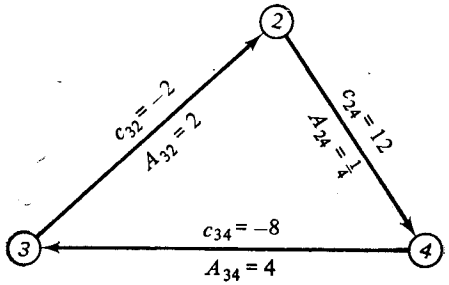


Рис. 5.3. Подсеть.

$$F_h=\frac{g-1}{g}\left\{\min_{1\leqslant k\leqslant m}\left[U_k\prod_{r=k}^{m}a_r\right]\right\}, \qquad (5.10г)$$

где h — номер выпускающего узла, g — коэффициент выигрыша цикла, m — число дуг в цикле, U_k — верхняя граница потока по k-й дуге цикла, a_k — множитель k-й дуги. Предполагается, что дуги цикла занумерованы так же, как это было сделано при рассмотрении формулы (5.9). Отметим, что правая часть равенства (5.10г) равна правой части равенства (5.10а), умноженной на функцию от коэффициента выигрыша цикла. В следующей таблице приводятся результаты вычислений, проведенных на итерации 2.

$m=3$

Дуга (i,j)	k	U_k	$\prod_{r=k}^{m} a_r$	$\bar{F}_k$
(4, 3)	1	1/2	2	1
(3, 2)	2	2	1/2	1
(2, 4)	3	4	1/4	1

Из (5.10г) следует, что $F_h = 1/2 \cdot \min[1, 1, 1] = 1/2$. Следовательно, максимальный поток в выпускающем узле равен 1/2.

Для определения нужных поправок потоков по дугам в случае, когда существует генерирующий цикл, нельзя воспользоваться обычными последовательными процедурами обратного поиска. Однако в этом случае может быть использована следующая формула [16]:

$$\bar{f}_k = F_h \left[\frac{g}{g-1} \right] \Big| \prod_{r=k}^{m} a_r. \qquad (5.10д)$$

Применяя к дугам цикла, изображенного на рис. 5.3, формулу (5.10д) при $m=3$, $F_h=1/2$ и $g/(g-1)=2$, получаем следующие поправки:

Дуга (i, j)	k	$\prod_{r=k}^{m} a_r$	$\bar{f}_k$
(4, 3)	1	2	1/2
(3, 2)	2	1/2	2
(2, 4)	3	1/4	4

Поправки потоков по дугам, не указанным в приведенной выше таблице, равны нулю. Новая величина потока по дуге равна сумме величины, вычисленной на итерации 1, и поправки, соответствующей потоку по дуге в маргинальной сети, т. е.

$$f_{12}^{(2)} = f_{12}^{(1)} + 0 = 12 + 0 = 12,$$
$$f_{13}^{(2)} = f_{13}^{(1)} + 0 = 0 + 0 = 0,$$
$$f_{23}^{(2)} = f_{23}^{(1)} - \frac{2}{1/2} = 4 - 4 = 0,$$
$$f_{24}^{(2)} = f_{24}^{(1)} + 4 = 0 + 4 = 4,$$
$$f_{34}^{(2)} = f_{34}^{(1)} - \frac{1/2}{1/4} = 2 - 2 = 0.$$

Приращение потока равно 1/2 и после выполнения итерации 2 $F=1$, $\bar{F}=12$. Стоимость получения единицы потока равна 32 (итерация 1) $+80 \cdot 1/2 = 72$.

Остановимся на обобщении полученных выше результатов. Пусть $\bar{f}_{ij}^{(I-1)}$ — поток по дуге (i, j) исходной сети на итерации $I-1$. Пусть $\bar{f}_{ij}^{(I)}$ — величина потока по дуге (i, j) маргинальной сети, построенной на I-й итерации. И наконец, пусть $\bar{f}_{ji}^{(I)}$ — ве-

личина потока по зеркальной дуге (j, i) маргинальной сети, построенной на I-й итерации. Если на итерации I—1 дуга (i, j) не насыщена, то ее входной поток может быть увеличен на величину выходного потока узла i маргинальной сети итерации I, т. е.

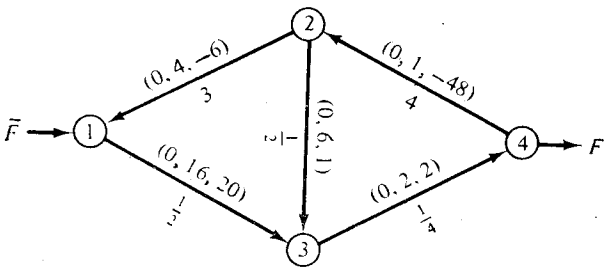
$$f_{ij}^{(I)} = f_{ij}^{(I-1)} + \bar{f}_{ij}^{(I)}. \qquad (5.11а)$$

С другой стороны, если поток по дуге (i, j) исходной сети положительный, то он может быть уменьшен на величину выходного потока зеркальной дуги (j, i) маргинальной сети итерации I, т. е.

$$f_{ij}^{(I)} = f_{ij}^{(I-1)} - \bar{f}_{ji}^{(I)}/A_{ij}. \qquad (5.11б)$$

Отслеживая величины $f_{ij}^{(2)}$ потоков по дугам, можно показать, что новая величина потока в стоке равна 1, а стоимость потока равна 72. Дуги (1, 2) и (2, 4) теперь стали насыщенными и поэтому могут быть исключены. Зеркальные дуги (2, 1) и (4, 2) включаются в маргинальную сеть для того, чтобы определить способ возможного уменьшения потоков.

Итерация 3.



Пометка (0, 0) узла 1 не изменяется, т. е. $V_1 = 0$. Результаты выполнения процедуры расстановки пометок приведены в следующей таблице.

Узел j	Дуга, входящая в узел j	c_{ij}	A_{ij}	V_i	$\bar{V}_{ij}$	V_j	Пометка
3	(1, 3)	20	1/2	0	40	40	(1, 40)
4	(3, 4)	2	1/4	40	168	168	(3, 148)
2	(4, 2)	−48	4	168	30	30	(4, 30)
1	(2, 1)	−6	3	30	8	0	(0, 0)
3	(2, 3)	1	1/2	30	64	—	

Ни один цикл не найден, и каждому узлу приписана пометка. Цепь минимальной стоимости задается последовательностью

дуг (1,3), (3, 4). Максимальный поток в стоке может быть вычислен с помощью формулы (5.10а), для применения которой необходимы величины, содержащиеся в следующей таблице.

$m=2$

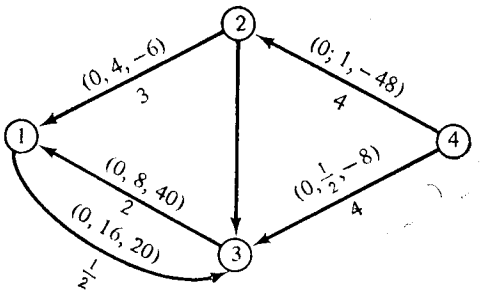
Дуга (i,j)	k	U_k	$\prod_{r=k}^{m} a_r$	$\bar{F}_k$
(1, 3)	1	16	1/8	2
(3, 4)	2	2	1/4	1/2

Из (5.10а) следует, что $F_t=\min[2,\ 1/2]=1/2$. Выполняя процедуру обратного поиска так, как это показано на итерации 1, получаем следующие поправки: $\bar{f}_{34}{}^{(3)}=(1/2)/(1/4)=2$, $\bar{f}_{13}{}^{(3)}=2/(1/2)=4$, $\bar{f}_{21}{}^{(3)}=\bar{f}_{23}{}^{(3)}=0$. Измененные потоки по дугам исходной сети могут быть вычислены с помощью соотношений (5.11а) и (5.11б):

$$f_{12}{}^{(3)}=f_{12}{}^{(2)}-\bar{f}_{21}{}^{(3)}/A_{12}=12-0=12,$$
$$f_{13}{}^{(3)}=f_{13}{}^{(2)}+\bar{f}_{13}{}^{(3)}=0+4=4,$$
$$f_{34}{}^{(3)}=f_{34}{}^{(2)}+\bar{f}_{34}{}^{(3)}=0+2=2.$$

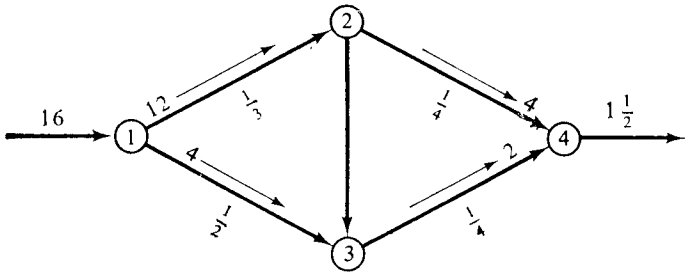
Кроме того, поскольку $\bar{f}_{23}{}^{(3)}=\bar{f}_{24}{}^{(3)}=0$, то $f_{23}{}^{(3)}=f_{23}{}^{(2)}=0$, $f_{24}{}^{(3)}=f_{24}{}^{(2)}=4$. Следовательно, увеличение потока вновь составляет 1/2 единиц, общий поток равен $F=1\frac{1}{2}$ и $\bar{F}=16$. Стоимость получения этого потока равна 32 (итерация 1) +40 (итерация 2) $+(1/2)\cdot 168=156$. Дуги (1, 2), (2, 4) и (3, 4) исключаются, а зеркальные дуги (4, 3) и (3, 1) включаются для того, чтобы определить способ возможного уменьшения потоков по соответствующим исходным дугам.

Итерация 4



Узлу 4 должна быть приписана пометка $(0,\ \infty)$. Поскольку аугментальная цепь потока не может быть построена, то теку-

щее решение является оптимальным. Как видно из приводимого ниже рисунка, в узел 1 входит 16 единиц потока, а из узла 4 выходит $1\frac{1}{2}$ единиц потока. Соответствующая минимальная стоимость равна $12 \cdot 2 + 4 \cdot 20 + 4 \cdot 12 + 2 \cdot 2 = 156$, что также следует из результатов, полученных на итерации 3.



5.10. ЗАКЛЮЧЕНИЕ

Алгоритмы, описанные в части I, были даны с той целью, чтобы показать сложность обобщенных сетевых задач и сообщить читателю основные сведения, необходимые для решения других классов задач. Процедура расстановки пометок, хотя и носит общий характер, при численной реализации становится громоздкой и относительно неэффективной для больших систем. В своей работе Йенсен и Барнес [17] обобщают основные понятия, рассмотренные в разд. 5.5—5.9, и предлагают подход, заслуживающий особого внимания с вычислительной точки зрения. Кроме того, ими разработан высокоэффективный алгоритм, основанный на теории двойственности в линейном программировании, позволяющий решать задачи большой размерности с линейной, строго выпуклой или строго вогнутой целевой функцией. Для более глубокого изучения сетевого анализа мы рекомендуем обратиться к этой работе.

Хотя в данном разделе не проводилось исследование поглощающих циклов, оно может быть проведено аналогично исследованию генерирующих циклов. Однако в задачах о максимальном потоке поглощающие циклы не представляют особого интереса и при разработке алгоритмов их можно не рассматривать.

ЧАСТЬ II. СТОХАСТИЧЕСКИЕ СЕТИ. ГРАФИЧЕСКИЙ МЕТОД ОЦЕНКИ И ПЕРЕСМОТРА ПЛАНОВ (ГЕРТ)

До сих пор мы рассматривали лишь те системы, которые описываются детерминированными сетями. Для полного выполнения типичной сети проекта необходимо выполнение всех

дуг[1]). Из этого условия следует, что в такую модель не могут быть включены операции с обратной связью, поскольку они представляются петлями, существование которых в свою очередь означает, что конечный узел операции должен быть выполнен раньше ее начального узла. В области детерминированных сетей наиболее полно были изучены две модели. В первой из них, модели критического пути, время выполнения каждой дуги фиксированно. Во второй, модели ПЕРТ, для каждой дуги существует несколько возможных времен ее выполнения.

При моделировании работы промышленных комплексов нередко наиболее гибкими и полезными оказываются сетевые модели со стохастической структурой. Стохастическую сеть определим как сеть, которая может быть выполнена только при выполнении некоторого подмножества дуг; при этом время выполнения каждой дуги (операции) выбирается в соответствии с вероятностным распределением. В стохастических сетях для выполнения узла не является необходимым выполнение всех дуг, входящих в него. Поэтому в таких моделях допускается существование циклов и петель,

5.11. СЕТЕВОЕ ПРЕДСТАВЛЕНИЕ

Узлы стохастической сети могут быть интерпретированы как состояния системы, а дуги — как переходы из одного состояния в другое. Такие переходы можно рассматривать как выполнение обобщенных операций, характеризуемых плотностью распределения, или функцией массы, и вероятностью выполнения.

Каждый внутренний узел стохастической сети выполняет две функции, одна из которых касается входа в узел, а другая — выхода. Обычно эти функции называют входной и выходной.

1. *Входная функция.* Она определяет условие, при котором узел может быть выполнен.

2. *Выходная функция.* Она определяет совокупность условий, связанных с результатом выполнения узла. Другими словами, с помощью выходной функции указывается, должны ли выполняться все операции, которым данный узел непосредственно предшествует, или только одна из них.

Отметим, что начальный узел сети выполняет только выходную функцию, в то время как конечный узел — только входную. Существуют три типа входных функций.

5.11.1. ВХОДНЫЕ ФУНКЦИИ

Тип 1. Узел выполняется, если выполнены все дуги, входящие в него.

[1]) Под выполнением дуг и узлов сети понимается выполнение соответствующих операций. — *Прим. перев.*

Тип 2. Узел выполняется, если выполнена любая дуга, входящая в него.

Тип 3. Узел выполняется, если выполнена любая дуга, входящая в него, при условии, что в заданный момент времени может выполняться только одна дуга.

5.11.2. ВЫХОДНЫЕ ФУНКЦИИ

Тип 1. Все дуги, выходящие из узла, выполняются, если этот узел выполнен. Данная функция называется детерминированной выходной функцией.

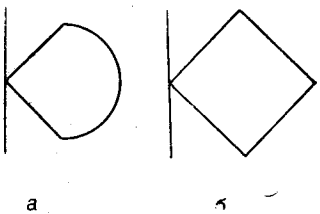


Рис. 5.4. Типы ГЕРТ-узлов.
а — детерминированный выход; *б* — вероятностный выход.

Тип 2. Ровно одна дуга, выходящая из узла, выполняется, если узел выполнен. Выбор такой дуги может быть описан с помощью вероятности. Поэтому эта функция называется вероятностной.

В настоящем разделе мы рассмотрим два типа узлов: а) узлы с третьим типом входной функции и детерминированной выходной функцией и б) узлы с третьим типом входной функции и вероятностной выходной функцией. Сети, содержащие только эти два типа узлов, называются ГЕРТ-сетями. Для ГЕРТ-узлов используются обозначения, приведенные на рис. 5.4 [34, 35].

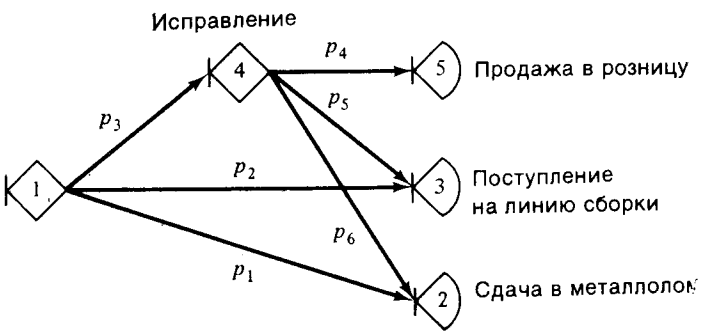


Рис. 5.5. Пример ГЕРТ-сети.

Покажем, как можно воспользоваться этими обозначениями при рассмотрении простой системы контроля качества продукции (рис. 5.5). В результате выполнения операции проверки система определяет, какие детали следует сдать в металлолом, какие исправить, а какие отправить на линию сборки. После исправления детали могут быть проданы в розницу, сданы в металлолом или отправлены непосредственно на линию сборки.

Отметим, что $p_1+p_2+p_3=1$ и $p_4+p_5+p_6=1$. Кроме того, дуги (4, 2), (1, 4), (1, 2), (1, 3), (4, 3) и (4, 5) соответствуют физическим процессам, которые можно описать плотностями распределения. Например, случайные величины, соответствующие операциям (1, 4) и (4, 5), могут быть нормально распределенными, операциям (1, 2) и (4, 3) — экспоненциально распределенными, а операциям (1, 3) и (4, 2) — равномерно распределенными. Методы системы ГЕРТ позволяют ответить на следующие вопросы:

1. Какова вероятность того, что деталь будет сдана в металлолом?
2. Какова вероятность того, что деталь будет использована на линии сборки?
3. Какова вероятность того, что деталь поступит в розничную продажу?
4. Чему равны математическое ожидание и дисперсия времени, необходимого для изготовления детали, которая поступает на линию сборки?
5. Сколько времени будет потеряно, если деталь сдается в металлолом?

В последующих разделах будет разработан аналитический подход, позволяющий ответить на эти вопросы.

5.12. ОСНОВНЫЕ ПРОЦЕДУРЫ СИСТЕМЫ ГЕРТ

Рассмотрим сеть $G=(\mathbf{N}, \mathbf{A})$, содержащую только ГЕРТ-узлы, которые образуют множество **N**. Пусть время выполнения операции (i, j) есть случайная величина Y_{ij}. По определению

Таблица 5.1. Моменты и производящие функции моментов

Тип распределения	$M_E(s)$	Математическое ожидание	Второй момент
Биномиальное (B)	$(pe^s+1-p)^n$	np	$np(np+1-p)$
Дискретное (D)	$\dfrac{p_1e^{sT_1}+p_2e^{sT_2}+\cdots}{p_1+p_2+\cdots}$	$\dfrac{p_1T_1+p_2T_2+\cdots}{p_1+p_2+\cdots}$	$\dfrac{p_1T_1^2+p_2T_2^2+\cdots}{p_1+p_2+\cdots}$
Экспоненциальное (E)	$\left(1-\dfrac{s}{a}\right)^{-1}$	$\dfrac{1}{a}$	$\dfrac{2}{a^2}$
Гамма (GA)	$\left(1-\dfrac{s}{a}\right)^{-b}$	$\dfrac{b}{a}$	$\dfrac{b(b+1)}{a^2}$
Геометрическое (GE)	$\dfrac{pe^s}{1-e^s+pe^s}$	$\dfrac{1}{p}$	$\dfrac{2-p}{p^2}$
Отрицательное биномиальное (NB)	$\left(\dfrac{p}{1-e^s+pe^s}\right)^r$	$\dfrac{r(1-p)}{p}$	$\dfrac{r(1-p)(1+r-rp)}{p^2}$
Нормальное (NO)	$e^{sm+(1/2)s^2\sigma^2}$	m	$m^2+\sigma^2$
Пуассона (P)	$e^{\lambda(e^s-1)}$	λ	$\lambda(1+\lambda)$
Равномерное (U)	$\dfrac{e^{sa}-e^{sb}}{(a-b)s}$	$\dfrac{a+b}{2}$	$\dfrac{a^2+ab+b^2}{3}$

операция (i, j) может быть выполнена только в том случае, если выполнен узел i. Поэтому для изучения вопросов, связанных с выполнением этой операции, необходимо знать условную вероятность (в дискретном случае) или плотность распределения (в непрерывном случае) случайной величины Y_{ij} при условии, что узел i выполнен. Это в свою очередь позволит нам провести исследования, связанные с выполнением всей сети. В частности, мы сможем определить моменты распределения времени выполнения сети, с помощью которых будут вычислены математическое ожидание и дисперсия времени выполнения сети.

Пусть f_{ij} — условная вероятность или плотность распределения времени выполнения операции (i, j). Условная производящая функция моментов случайной величины Y_{ij} определяется как $M_{ij}(s) = E[e^{sY_{ij}}]$, т. е.

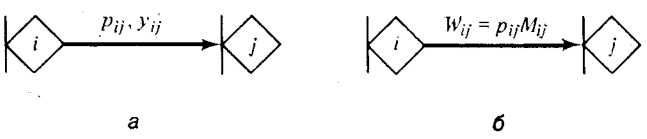
$$M_{ij}(s) = \begin{cases} \int e^{sy_{ij}} f(y_{ij})\, dy_{ij} & \text{(для непрерывной случайной величины),} \\ \sum e^{sy_{ij}} f(y_{ij}) & \text{(для дискретной случайной величины).} \end{cases}$$

В частности, $M_{ij}(s) = E[e^{sa}] = e^{sa}$ при $y_{ij} = a = \text{const}$. Если $a = 0$, то $M_{ij}(s) = 1$. В табл. 5.1 описаны некоторые наиболее важные функции распределения и указаны соответствующие производящие функции моментов и первые (математические ожидания) и вторые моменты относительно начала координат.

Пусть p_{ij} — вероятность того, что операция (i, j) будет выполнена при условии, что узел i выполнен. Для случайной величины Y_{ij} определим *W-функцию* [34, 35] как

$$W_{ij}(s) = p_{ij} M_{ij}(s). \tag{5.12}$$

С помощью преобразования (5.12) всегда можно определить сеть G', структура которой идентична структуре сети G, только вместо двух параметров дуг p_{ij} и y_{ij} присутствует один параметр W_{ij}. На рис. 5.6 изображены дуга сети G и соответствующая ей дуга сети G'.



а *б*

Рис. 5.6. Сети G и G'.
а — элемент сети G; *б* — элемент сети G'.

В описание системы ГЕРТ мы включили в качестве параметра дуги время выполнения соответствующей операции.

В действительности можно рассматривать также любой характерный параметр, который обладает аддитивностью по дугам любого пути.

Если времена выполнения операций сети G представляются независимыми случайными величинами, то G' обладает рядом свойств, представляющих интерес с вычислительной точки зрения. Для изучения этих свойств мы рассмотрим три частных случая: 1) G' состоит из двух последовательных дуг; 2) G' — из двух параллельных ветвей; 3) G' — из одной ветви и одной петли.

5.12.1. ПОСЛЕДОВАТЕЛЬНЫЕ ДУГИ

Рассмотрим простую сеть, изображенную на рис. 5.7. Она состоит из двух последовательных ветвей. Эти две ветви могут быть заменены одной эквивалентной им ветвью, как показано

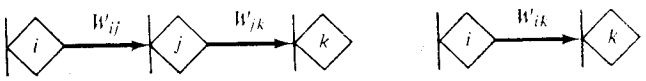


Рис. 5.7. Последовательные ветви и их эквивалентное представление.

ниже. Исходные ветви имеют W-преобразования $W_{ij}(s) = p_{ij}M_{ij}(s)$, $W_{jk}(s) = p_{jk}M_{jk}(s)$. W-функция для эквивалентной ветви (i, k) имеет вид $W_{ik}(s) = p_{ik}M_{ik}(s)$. Напомним, что производящая функция моментов суммы двух независимых случайных величин равна произведению производящих функций моментов этих величин. Тогда поскольку $p_{ik} = p_{ij}p_{jk}$ и $M_{ik}(s) = [M_{ij}(s)]\,[M_{jk}(s)]$, то

$$W_{ik}(s) = [p_{ij}M_{ij}(s)]\,[p_{jk}M_{jk}(s)] = W_{ij}(s)\,W_{jk}(s). \qquad (5.13)$$

Основной результат (5.13) может быть обобщен на случай трех и более ветвей: W-функция для эквивалентной ветви равна произведению W-функций для последовательных ветвей.

5.12.2. ПАРАЛЛЕЛЬНЫЕ ВЕТВИ

Рассмотрим простую сеть, изображенную на рис. 5.8. Она состоит из двух параллельных ветвей. Будет доказано, что эти ветви могут быть заменены эквивалентной ветвью. Пусть (i, j) — такая ветвь. По определению $W_{ij}(s) = p_{ij}M_{ij}(s)$. В этом случае $p_{ij} = p_a + p_b$ и $M_{ij}(s) = [p_aM_a(s) + p_bM_b(s)]/(p_a + p_b)$. Поэтому

$$W_{ij}(s) = [p_a + p_b]\left[\frac{p_aM_a(s) + p_bM_b(s)}{p_a + p_b}\right] = W_a(s) + W_b(s). \qquad (5.14)$$

Формула (5.14) также может быть обобщена на случай сетей, состоящих из трех и более параллельных ветвей: W-функция для эквивалентной ветви равна сумме W-функций для параллельных дуг.

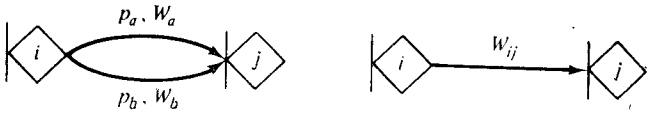


Рис. 5.8. Параллельные ветви и их эквивалентное представление.

5.12.3. ПЕТЛИ

Рассмотрим простую сеть, изображенную на рис. 5.9. Она состоит из одной петли и одной дуги и может быть преобразована в эквивалентную сеть, содержащую только одну дугу.

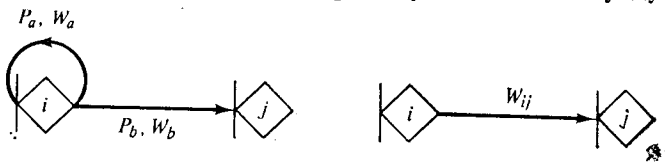


Рис. 5.9. Сеть с петлей и ее эквивалентное представление.

Отметим, что рассматриваемая сеть может быть преобразована в сеть, изображенную на рис. 5.10. Эта новая сеть состоит из бесконечной последовательности параллельных цепей, каждая из которых представляет собой совокупность последова-

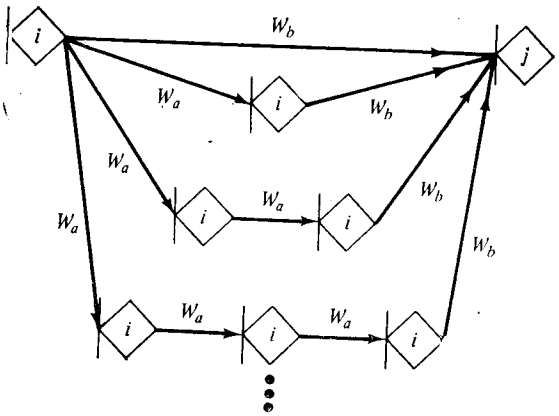


Рис. 5.10. Представление петель в виде параллельных и последовательных дуг.

тельных ветвей. Поэтому можно вначале каждую ее цепь свести к эквивалентной дуге, а затем эти дуги преобразовать в сеть, состоящую из одной ветви, эквивалентную исходной системе.

Пусть (i, j) — ветвь, эквивалентная сети, изображенной на рис. 5.10. Из (5.13) и (5.14) следует, что вес ветви (i, j) равен $W_{ij} = W_b + W_a W_b + W_a^2 W_b + \ldots = W_b[1 + \sum_{m=1}^{\infty} W_a^m]$. Данное выражение, в котором мы временно опустили аргументы W-функций, можно упростить, зная, что биномиальный ряд $(1 - W_a)^{-1}$ раскладывается следующим образом:

$$(1 - W_a)^{-1} = 1 + W_a + W_a^2 + W_a^3 + \ldots = 1 + \sum_{m=1}^{\infty} W_a^m.$$

Таким образом, окончательно имеем

$$W_{ij}(s) = W_b(s)\,[1 - W_a(s)]^{-1} = W_b(s)/[1 - W_a(s)]. \qquad (5.15)$$

Следовательно, сеть, изображенная на рис. 5.10, сводится к одной-единственной эквивалентной ей ветви, для которой W-функция равна $W_{ij}(s) = W_b(s)/[1 - W_a(s)]$. Отметим, что описанная процедура может быть использована и для контуров, поскольку с помощью формулы (5.13) контур сводится к петле.

Таким образом, если ГЕРТ-сеть состоит из параллельных и последовательных цепей и (или) петель, то она может быть преобразована в эквивалентную сеть, состоящую из одной-единственной ветви. На самом деле, данный результат обобщается на любую ГЕРТ-сеть, поскольку можно комбинировать базисные преобразования.

Перед тем как перейти к рассмотрению новых вопросов, нам хотелось бы познакомить читателя с основными понятиями, касающимися потоковых графов и необходимыми для понимания последующего материала. Данный раздел мы завершим кратким описанием основных шагов системы ГЕРТ:

1. Представить систему в виде стохастической сети с ГЕРТ-узлами.
2. Для каждой дуги сети определить условную вероятность и производящую функцию моментов.
3. Для каждой дуги сети вычислить W-функцию.
4. Преобразовать сеть в эквивалентную сеть, состоящую из одной ветви.

5.13. ОСНОВНЫЕ ПОНЯТИЯ О ПОТОКОВЫХ ГРАФАХ

Система может быть определена как совокупность активных взаимодействующих между собой элементов, которые выполняют некоторые функции. В дальнейшем мы будем рассматривать

только такие системы, в которых элементы и взаимосвязь между элементами описываются линейными уравнениями. Для графического описания таких систем наиболее широко используются потоковые графы. В потоковом графе элементы системы представляются узлами, а взаимосвязь между элементами, или функции перехода, — дугами.

Основным элементом потокового графа является ориентированная ветвь, направленная из узла i в узел j, с параметром t_{ij}. Направление ветви указывает связь по входу и выходу между двумя переменными, представленными узлами этой ветви. Узел i соответствует независимой переменной x_i, а узел j — независимой переменной x_j. Параметр t_{ij} обычно называют *коэффициентом пропускания дуги*. Он равен множителю, используемому для того, чтобы преобразовать величину x_i до рассмотрения ее как части величины x_j. Основное свойство потоковых графов заключается в том, что величина узла равна сумме преобразованных величин узлов, соединенных с рассматриваемым узлом входящими в него дугами. В качестве примера рассмотрим элементарный потоковый граф, изображенный на рис. 5.11.

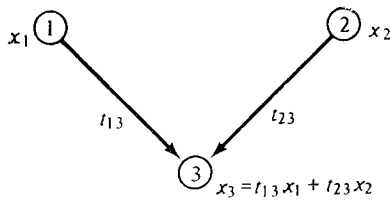


Рис. 5.11. Основное свойство потоковых графов.

Основные свойства потоковых графов могут быть использованы для разработки методов, позволяющих обращаться непосредственно с элементами графа, в результате чего он будет преобразован в эквивалентный граф с более простой структурой и, следовательно, упростится решение задачи. Наиболее известным результатом в данной области является топологическое уравнение Мейсона [34], которое может быть использовано для потоковых графов с произвольной структурой. Для рассмотрения уравнения Мейсона нам понадобятся некоторые определения.

5.14. ОПРЕДЕЛЕНИЯ

Петля: связная последовательность ориентированных ветвей, каждый узел которых является общим ровно для двух ветвей. Петлю обычно называют петлей первого порядка, чтобы указать на то, что она не содержит других петель и что каждый узел можно достичь из любого другого узла. Собственную петлю[1] можно рассматривать как вырожденную петлю первого порядка.

[1] То есть петлю, состоящую из одной ветви. — *Прим. перев.*

Петля порядка n: множество n не связанных между собой петель первого порядка.

Замкнутый потоковый граф: граф, в котором каждая ветвь принадлежит по крайней мере одной петле.

В качестве примера рассмотрим потоковый граф, изображенный на рис. 5.12, для которого найдем все петли и определим их порядок.

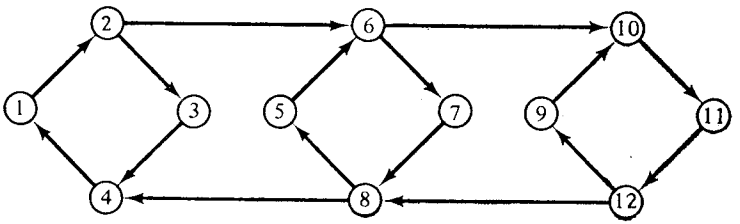


Рис. 5.12. Потоковый граф с петлями.

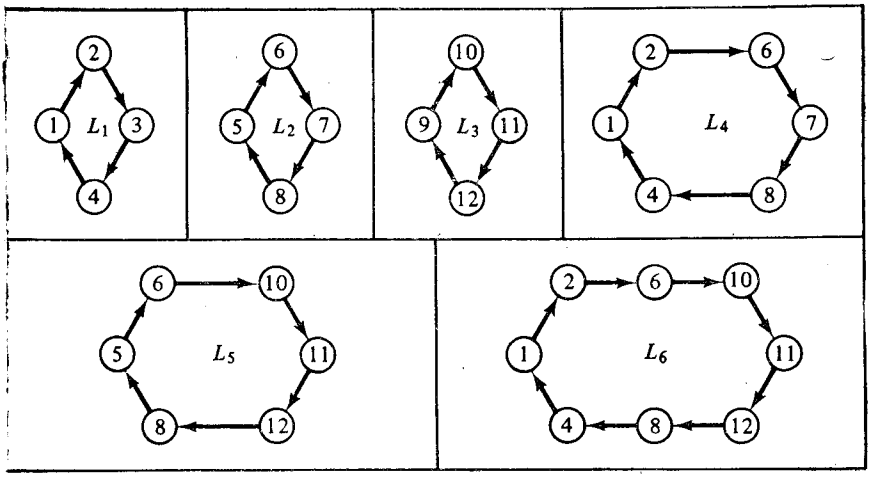


Рис. 5.13. Петли первого порядка в замкнутом потоковом графе.

Нетрудно заметить, что данный потоковый граф является замкнутым, поскольку он целиком состоит из петель. На рис. 5.13, и 5.14 и 5.15 изображены петли первого, второго и третьего порядка соответственно для графа, приведенного на рис. 5.12. Как было определено выше, петля второго (третьего) порядка — это множество, состоящее из двух (трех) не связанных между собой петель первого порядка. Например, петли L_1 и L_2

образуют петлю второго порядка, а петли L_1, L_2 и L_3 — петлю третьего порядка.

Петлю первого порядка можно рассматривать как цепь, состоящую из последовательно соединенных ориентированных ветвей, концевые узлы которой совпадают. Поэтому коэффициент

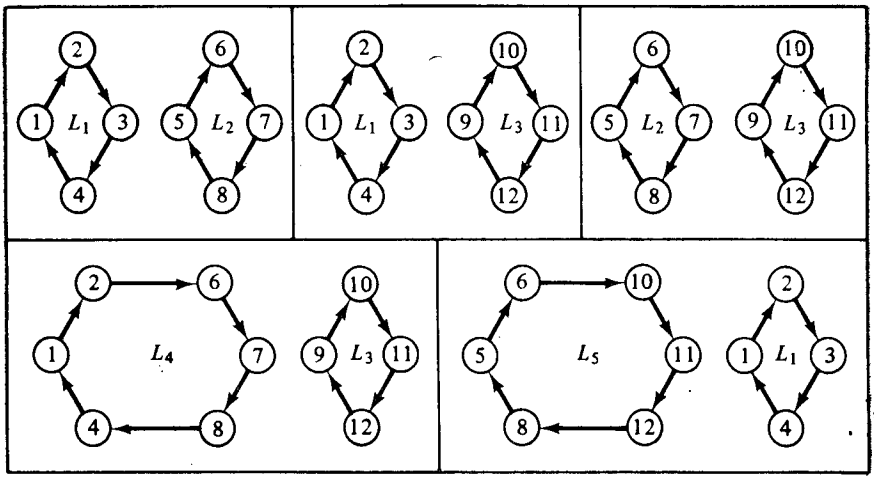


Рис. 5.14. Петли второго порядка в замкнутом потоковом графе.

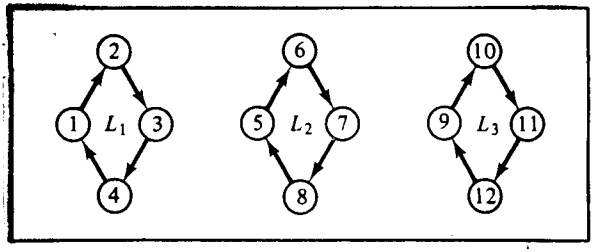


Рис. 5.15. Петли третьего порядка в замкнутом потоковом графе.

пропускания эквивалентного ей потокового графа равен произведению коэффициентов пропускания каждой ветви.

По определению петля порядка n состоит из n не связанных между собой петель первого порядка. Если каждую из этих петель первого порядка преобразовать в эквивалентный потоковый граф, состоящий из одной ветви, то исходную петлю порядка n можно будет рассматривать как связную последовательность таких ветвей. Отсюда следует, что коэффициент про-

пускания петли порядка n равен произведению коэффициентов пропускания n петель первого порядка.

Рассмотрим теперь общий случай. Пусть L_{11}, L_{21}, ..., L_{n1} — n не связанных между собой петель первого порядка заданной петли порядка n. Выше было установлено, что для каждой петли L_{k1} первого порядка эквивалентный коэффициент пропускания T_k равен произведению коэффициентов пропускания ветвей, принадлежащих этой петле, т. е.

$$T_k = \prod_{(i,\,j)\,\in L_{k1}} t_{ij}. \tag{5.16}$$

Следовательно, для петли порядка n эквивалентный коэффициент пропускания $T(L_n)$ равен

$$T(L_n) = \prod_{k=1}^{n} T_k = \prod_{k=1}^{n} \left[\prod_{(i,\,j)\,\in L_{k1}} t_{ij} \right]. \tag{5.17}$$

Основной результат (5.17) будет использован для определения поведения систем, которые могут быть описаны ГЕРТ-сетями.

5.15. ПРАВИЛО МЕЙСОНА ДЛЯ ЗАМКНУТЫХ ПОТОКОВЫХ ГРАФОВ

Цель использования системы ГЕРТ в стохастическом сетевом анализе состоит в вычислении математического ожидания и дисперсии времени выполнения сети (которые рассматриваются здесь как общий параметр сети) и вероятности выполнения

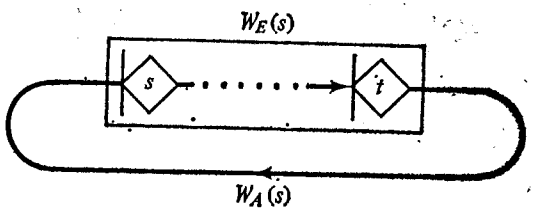


Рис. 5.16. Замкнутая стохастическая сеть.

стока (или стоков). Очевидно, что коэффициент пропускания дуги ГЕРТ-сети есть соответствующая W-функция. Напомним, что W-функция дуги определяется как произведение вероятности выполнения этой дуги и производящей функции моментов для времени выполнения операции, представленной этой дугой.

В предыдущем разделе мы показали, как определить все петли замкнутого потокового графа. Для того чтобы применить эти результаты к открытой сети, необходимо ввести дополни-

тельную дугу с W-функцией $W_A(s)$, соединяющую сток t с источником s. Затем для модифицированной сети нужно найти все петли (вплоть до максимально возможного порядка). Функция $W_A(s)$ необходима для того, чтобы найти эквивалентную W-функцию для исходной сети. На рис. 5.16 исходная сеть изображена в виде «черного ящика» с W-функцией $W_E(s)$.

Напомним, что эквивалентный коэффициент пропускания для петли порядка n равен произведению коэффициентов пропускания n не связанных между собой петель первого порядка, т. е.

$$T(L_n)=\prod_{k=1}^{n} T_k. \tag{5.18}$$

Топологическое уравнение для замкнутых графов, известное также как правило Мейсона, имеет следующий вид:

$$H=1-\sum T(L_1)+\sum T(L_2)-\sum T(L_3)+\ldots+$$
$$+(-1)^m\sum T(L_m)+\ldots=0, \tag{5.19}$$

где $\Sigma T(L_i)$ — сумма эквивалентных коэффициентов пропускания для всех возможных петель i-го порядка. Иными словами, при использовании топологического уравнения необходимо выполнить следующие шаги:

1. Вычислить эквивалентный коэффициент пропускания для всех петель порядка m.

2. Просуммировать полученные величины по всем петлям порядка m и результат умножить на $(-1)^m$. Отметим, что для петель с четным порядком величина $(-1)^m$ положительная, а для петель с нечетным порядком — отрицательная. В топологиком уравнении (5.19) перед каждым слагаемым поставить знак плюс или минус.

3. К выражению, полученному на шаге 2, прибавить 1 и результат приравнять к нулю.

В качестве примера рассмотрим рис. 5.16. Данный замкнутый потоковый граф содержит одну петлю первого порядка с эквивалентным коэффициентом пропускания, равным $W_A(s)W_E(s)$. По правилу Мейсона получаем, что $1-W_A(s)W_E(s)=0$ или $W_A(s)=1/W_E(s)$. Отметим, что функция $W_A(s)$ содержится в топологическом уравнении, поскольку она является элементом по крайней мере одной петли первого порядка. Важность результата, полученного при рассмотрении данного примера, состоит в том, что если в топологическом уравнении $W_A(s)$ заменить на $1/W_E(s)$ и решить его относительно $W_E(s)$, то будет получена эквивалентная W-функция для исходной стохастической сети.

В качестве другого примера рассмотрим открытую сеть, изображенную на рис. 5.17. Чтобы определить эквивалентную W-функцию для этой сети, нужно выполнить следующие шаги:

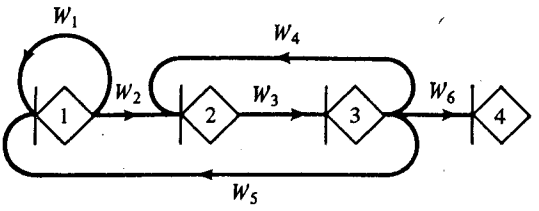


Рис. 5.17. Открытая стохастическая сеть.

1. Замкнуть данную сеть дугой, ведущей из узла 4 в узел 1.
2. Найти все петли порядка n.
3. С помощью топологического уравнения (5.19) получить выражение для $W_E(s)$.

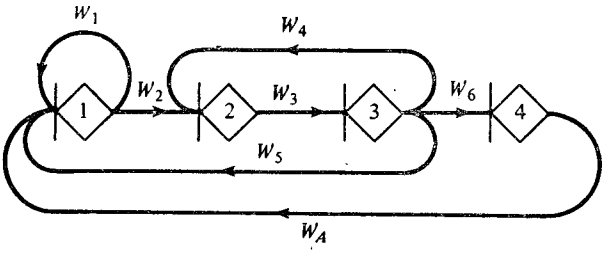


Рис. 5.18. Замкнутая стохастическая сеть.

На рис. 5.18 изображена модифицированная сеть, в которой для удобства записи опущен аргумент s у W-функций.

Если $W_A(s)$ заменить на $1/W_E(s)$, то из (5.17) для петель сети будут получены следующие коэффициенты пропускания.

Петли первого порядка: W_1, W_3W_4, $W_2W_3W_5$, $W_2W_3W_6(1/W_E)$.

Петля второго порядка: $W_1W_3W_4$.

Используя (5.19), получаем

$$H = 1 - W_1 - W_3W_4 - W_2W_3W_5 - W_2W_3W_6(1/W_E) + W_1W_3W_4 = 0.$$

Поэтому

$$W_E(s) = W_2W_3W_6/(1 - W_1 - W_3W_4 - W_2W_3W_5 + W_1W_3W_4),$$

что является эквивалентной W-функцией для сети, изображенной на рис. 5.17. В следующем разделе покажем, как с помощью этого равенства можно вычислить математическое ожидание и дисперсию времени выполнения сети (или подсети).

5.16. ВЫЧИСЛЕНИЯ МАТЕМАТИЧЕСКОГО ОЖИДАНИЯ И ДИСПЕРСИИ

Из топологического уравнения (5.19) было получено выражение для эквивалентной W-функции $W_E(s)$ сети. Напомним, что $M_E(s)=1$ при $s=0$. Поскольку $W_E(s)=p_E M_E(s)$, то $p_E=W_E(0)$, откуда следует, что

$$M_E(s)=W_E(s)/p_E=W_E(s)/W_E(0). \tag{5.20}$$

Отметим, что $W_E(s)$ можно выразить через W-функции всех или некоторых ветвей исходной сети. Нетрудно вычислить значение $W_E(0)$; для этого в выражении для $W_E(s)$, получаемом из (5.19), надо положить $s=0$.

Вычисляя j-ю частную производную по s функции $M_E(s)$ и полагая $s=0$, находим j-й момент μ_{jE} относительно начала координат, т. е.

$$\mu_{jE}=\frac{\partial^j}{\partial s^j}[M_E(s)]\,|_{s=0}. \tag{5.21}$$

В частности, первый момент μ_{1E} относительно начала координат есть математическое ожидание времени выполнения сети, а дисперсия времени выполнения сети равна разности между μ_{2E} и квадратом величины μ_{1E}, т. е.

$$\sigma^2=\mu_{2E}-(\mu_{1E})^2. \tag{5.22}$$

5.17. ПРИМЕНЕНИЕ СИСТЕМЫ ГЕРТ

5.17.1. ПРОИЗВОДСТВО ПРЕЦИЗИОННЫХ ДЕТАЛЕЙ

Фирма, выполняющая государственные подряды, согласилась изготавливать прецизионные детали. Из-за строгого технического контроля вероятность того, что из заготовки будет произведена доброкачественная деталь, равна 0,20. Чему равно ожидаемое число проб, необходимых для изготовления двух доброкачественных деталей?

В данной задаче предполагается, что время изготовления одной детали постоянно. Напомним, что если случайная величина Y_{ij} равна постоянной величине a, то

$$M_{ij}(s)=E[e^{sa}]=e^{sa}.$$

Будем считать, что $a=1$. Тогда $M_{ij}(s)$ является производящей функцией моментов для продолжительности каждой дуги исходной сети. ГЕРТ-сеть для данной задачи изображена на рис. 5.19. Поскольку вероятность изготовления доброкачественной детали равна 0,20, то $W_1=W_3=0{,}80e^s$, $W_2=W_4=0{,}20e^s$. Из тополо-

гического уравнения следует, что

$$H = 1 - W_1 - W_3 - W_2W_4W_A + W_1W_3 = 0,$$
$$1 - W_1 - W_3 - W_2W_4\,(1/W_E) + W_1W_3 = 0,$$
$$W_E(s) = W_2W_4/(1 - W_1 - W_3 + W_1W_3).$$

Используя выражения для W_1, W_2, W_3 и W_4, получаем, что $W_E(s) = 0{,}04e^{2s}/(1 - 1{,}6e^{s} + 0{,}64e^{2s})$. Поскольку $p_E = 1$, то $W_E(s) = M_E(s)$. Таким образом, ожидаемое число проб, необходимых для изготовления двух доброкачественных деталей, равно

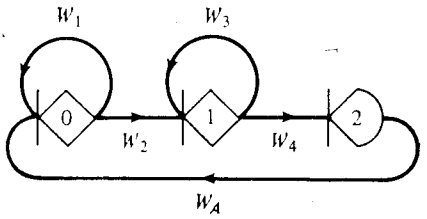


Рис. 5.19. ГЕРТ-сеть в задаче изготовления прецизионных деталей.

$$\mu_{1E} = \left.\frac{\partial M_E(s)}{\partial s}\right|_{s=0} = 10.$$

Аналогично дисперсия общего числа проб равна $\sigma^2 = 40$.

5.17.2. ПРОЦЕСС ПЕРЕРАБОТКИ СЫРЬЯ (ПРИТСКЕР)

Рассмотрим процесс производства полупроводника. Как показано на рис. 5.20, сырье вначале поступает в печь для очистки от примесей. Продукт, выходящий из печи, либо вновь по-

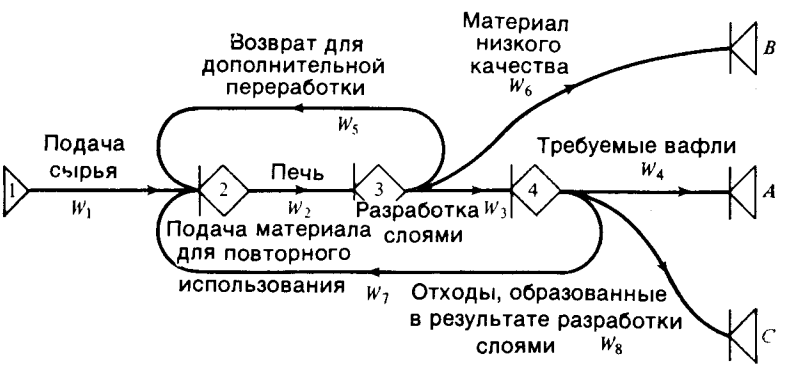


Рис. 5.20. Процесс переработки сырья.

Таблица 5.2. Характеристика операций процесса переработки сырья

Ветвь	p_i	Тип распределения	Параметры (ч)	Производящая функция моментов
(1, 2)	1	Постоянная величина	$a = 1$	exp $(1s)$
(2, 3)	1	Нормальное	$m = 0{,}5$	exp $[0{,}5s + \frac{1}{2}(0{,}01)s^2]$
			$\sigma = 0{,}1$	
(3, 2)	0,12		$m = 0{,}1$	exp $[0{,}1s + \frac{1}{2}(0{,}01)s^2]$
			$\sigma = 0{,}1$	
(3, *B*)	0,03	Постоянная величина	$a = 0{,}25$	exp $(0{,}25s)$
(3, 4)	0,85	Нормальное	$m = 0{,}25$	exp $[0{,}25s + \frac{1}{2}(0{,}04)s^2]$
			$\sigma = 0{,}20$	
(4, *A*)	0,75	Постоянная величина	$a = 0{,}20$	exp $(0{,}20s)$
(4, *C*)	0,05	Постоянная величина	$a = 0{,}05$	exp $(0{,}05s)$
(4, 2)	0,20	Постоянная величина	$a = 0{,}10$	$(0{,}10s)$

ступает в нее, либо используется как продукт низкого качества, либо проходит обработку слоями. После обработки слоями продукт либо идет на изготовление требуемых вафель, либо

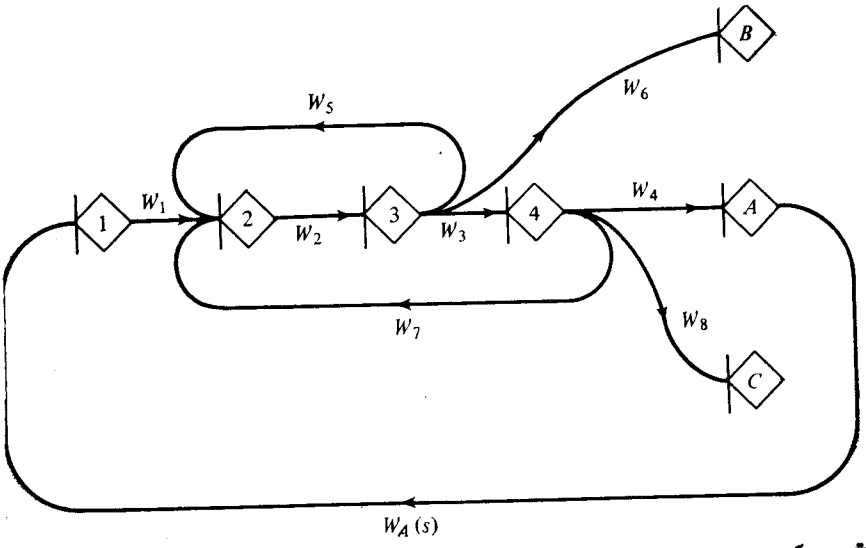


Рис. 5.21. Сеть, применяемая для анализа времени изготовления требуемой вафли.

поступает в отходы, образуемые в результате этого процесса, либо вновь используется как сырье. Вычислим среднюю величину и дисперсию времени получения требуемой вафли (т. е.

времени перехода в состояние A). В табл. 5.2 содержится информация о каждой ветви сети.

Поскольку нас интересует математическое ожидание и дисперсия времени изготовления требуемой вафли, необходимо ввести дугу $(A, 1)$, как это показано на рис. 5.21. Используя равенство $W_A(s) = 1/W_E(s)$, запишем топологическое уравнение в следующем виде:

$$H = 1 - W_1W_2W_3W_4(1/W_E) - W_2W_5 - W_2W_3W_7 = 0.$$

Отметим, что поскольку узел A должен быть выполнен, то дуги $(3, B)$ и $(4, C)$ при решении задачи не рассматриваются.

Решая топологическое уравнение относительно $W_E(s)$, получаем $W_E(s) = W_1W_2W_3W_4/(1 - W_2W_5 - W_2W_3W_7)$. Заменяя каждую W-функцию произведением соответствующей вероятности и производящей функции моментов, получаем, что

$$W_E(s) = \frac{0{,}6375 \exp(1{,}95s + 0{,}025s^2)}{1 - 0{,}12 \exp(0{,}6s + 0{,}01s^2) - 0{,}17 \exp(0{,}85s + 0{,}025s^2)}.$$

Нетрудно проверить, что в рассматриваемой задаче $W_E(0) = 0{,}8979$. Данную величину можно интерпретировать как вероятность изготовления требуемой вафли.

$$M_E(s) = \frac{W_E(s)}{W_E(0)} = \frac{0{,}71 \exp(1{,}95s + 0{,}025s^2)}{1 - 0{,}12 \exp(0{,}6s + 0{,}01s^2) - 0{,}17 \exp(0{,}85s + 0{,}025s^2)}.$$

Вычисляя первую и вторую частные производные по s функции $M_E(s)$ и полагая $s = 0$, получаем

$$\mu_{1E} = \left.\frac{\partial M_E(s)}{\partial s}\right|_{s=0} = 2{,}255 \text{ ч},$$

$$\mu_{2E} = \left.\frac{\partial^2 M_E(s)}{\partial s^2}\right|_{s=0} = 5{,}477 \text{ ч}^2,$$

$$\sigma^2 = \mu_{2E} - [\mu_{1E}]^2 = 0{,}392 \text{ ч}^2.$$

Таким образом, математическое ожидание времени изготовления требуемой вафли равно 2,255 ч, а дисперсия равна 0,392 ч^2. С помощью аналогичных вычислений могут быть определены характеристики состояний B и C.

5.17.3. ОПРЕДЕЛЕНИЕ ВЕРОЯТНОСТНЫХ НОРМАТИВНЫХ ВРЕМЕН ДЛЯ ЗАДАЧ, РЕШАЕМЫХ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ [31]

В данном примере мы вычислим математическое ожидание и стандартное отклонение нормативного времени, требуемого оператору для выполнения задания. Когда рабочему дается сложное задание, подобное тому, которое описано в настоящем

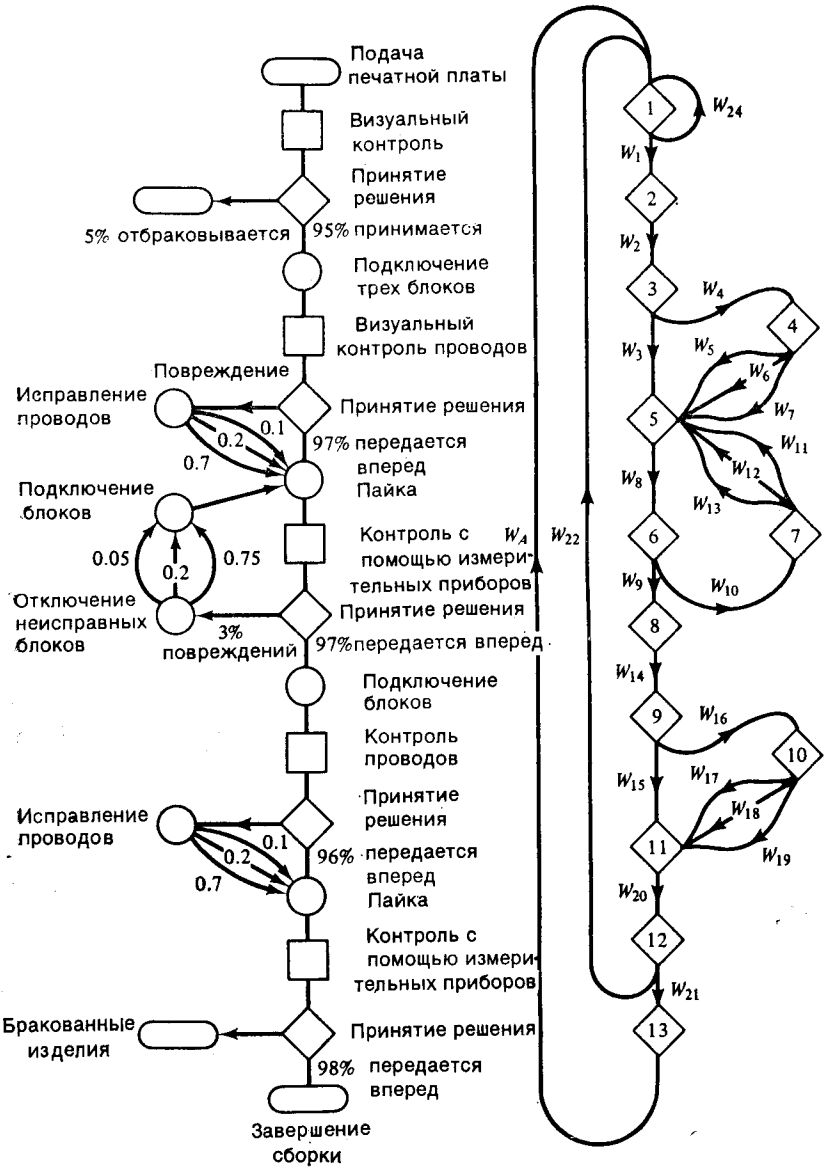


Рис. 5.22. Графическая постановка задачи: *а* — стохастическая схема сборки; *б* — ГЕРТ-сеть.

Таблица 5.3. Параметры ветвей ГЕРТ-сети, описывающей процесс сборки

Из/в	Описание	Вероятность	Распределение времени выполнения	Мат. ожидание	Дисперсия
1/2	W_1; благоприятный исход при контроле печатной платы	0,95	Нормальное	1,40	0,35
1/1	W_{24}; неблагоприятный исход при контроле печатной платы	0,05	Нормальное	1,40	0,35
2/3	W_2; подключение блоков	1,0	Нормальное	2,65	0,30
3/5	W_3; благоприятный исход при контроле проводов	0,97	Нормальное	0,75	0,20
3/4	W_4; неблагоприятный исход при контроле проводов	0,03	Нормальное	0,85	0,25
4/5	W_5; отладка блоков; повреждены провода одного блока	0,70	Нормальное	1,65	0,35
4/5	W_6; отладка блоков; повреждены провода двух блоков	0,20	Нормальное	2,15	0,40
4/5	W_7; отладка блоков; повреждены провода трех блоков	0,10	Нормальное	2,65	0,55
5/6	W_8; пайка	1,0	Нормальное	1,30	0,10
6/8	W_9; благопр. исход при контроле измерит. приборами	0,97	Нормальное	0,90	0,25
6/7	W_{10}; неблагоприятный исход при контроле измер. приборами	0,03	Нормальное	1,10	0,30
7/6	W_{11}; отключение и замена блоков; поврежден один блок	0,75	Нормальное	2,75	0,55
7/6	W_{12}; отключение и замена блоков	0,20	Нормальное	3,65	0,68
6/6	W_{13}; отключение и замена трех блоков	0,05	Нормальное	4,25	0,79
8/9	W_{14}; подключение блоков	1,0	Нормальное	2,65	0,30
9/11	W_{15}; благоприятный исход при контроле проводов	0,96	Нормальное	0,80	0,25
9/10	W_{16}; неблагоприятный исход при контроле проводов	0,04	Нормальное	0,95	0,30
10/11	W_{17}; отладка блоков; повреждены провода одного блока	0,70	Нормальное	2,0	0,5
10/11	W_{18}; отладка блоков; повреждены провода двух блоков	0,20	Нормальное	2,65	0,60
10/11	W_{19}; отладка блоков; повреждены провода трех блоков	0,10	Нормальное	3,15	0,75
11/12	W_{20}; пайка	1,0	Нормальное	2,10	0,40
12/13	W_{21}; благоприятный исход при контроле измер. приборами	0,98	Нормальное	0,80	0,15
12/1	W_{22}; неблагопр. исход при контроле измерит. приборами	0,02	Нормальное	0,85	0,15

Таблица 5.4. W-функция для ГЕРТ-сети

Начало	Конец	Ветвь (i,j)	Вероятность p_{ij}	W-функция $W = p\exp(\mu t + t^2\sigma^2/2)$
1	2	1	0,95	$0{,}95\exp(1{,}4t + 0{,}175t^2)$
2	3	2	1,0	$1{,}0\exp(2{,}65t + 0{,}15t^2)$
3	5	3	0,97	$0{,}97\exp(0{,}75t + 0{,}10t^2)$
3	4	4	0,03	$0{,}03\exp(0{,}85t + 0{,}125t^2)$
4	5	5	0,70	$0{,}70\exp(1{,}65t + 0{,}175t^2)$
4	5	6	0,20	$0{,}20\exp(2{,}15t + 0{,}20t^2)$
4	5	7	0,10	$0{,}10\exp(2{,}6t + 0{,}275t^2)$
5	6	8	1,0	$1{,}0\exp(1{,}3t + 0{,}05t^2)$
6	8	9	0,97	$0{,}97\exp(0{,}9t + 0{,}125t^2)$
6	7	10	0,03	$0{,}03\exp(1{,}1t + 0{,}15t^2)$
7	8	11	0,75	$0{,}75\exp(2{,}75t + 0{,}275t^2)$
7	8	12	0,20	$0{,}20\exp(3{,}65t + 0{,}34t^2)$
7	8	13	0,05	$0{,}05\exp(4{,}25t + 0{,}395t^2)$
8	9	14	1,0	$1{,}0\exp(2{,}65t + 0{,}15t^2)$
9	11	15	0,96	$0{,}96\exp(0{,}8t + 0{,}125t^2)$
9	10	16	0,04	$0{,}04\exp(0{,}95t + 0{,}15t^2)$
10	11	17	0,70	$0{,}70\exp(2t + 0{,}25t^2)$
10	11	18	0,20	$0{,}20\exp(2{,}65t + 0{,}325t^2)$
10	11	19	0,10	$0{,}10\exp(3.15t + 0{,}378t^2)$
11	12	20	1,0	$1{,}0\exp(2{,}1t + 0{,}20t^2)$
12	13	21	0,98	$0{,}98\exp(0{,}8t + 0{,}075t^2)$
12	1	22	0,02	$0{,}02\exp(0{,}85t + 0{,}075t^2)$
13	1	23	1,0	$1{,}0/W_E$
1	1	24	0,05	$0{,}05\exp(1{,}4t + 0{,}175t^2)$

разделе, разумно рассматривать нормативное время как случайную величину с конечным математическим ожиданием и дисперсией, описанную подходящей функцией распределения. Для примера остановимся на изучении процедуры сборки, описание которой дается на рис. 5.22, *а*.

Для получения дисперсионных оценок необходимы некоторые предположения, касающиеся стохастических характеристик каждого элемента (работы) в стандартных условиях. Такое описание задачи по сравнению со случаем, когда заданы только временны́е характеристики работ, является более сложным и, очевидно, более точным. В табл. 5.3 приводятся математическое ожидание и дисперсия для каждого элемента. Например, стохастическое поведение каждого элемента может быть описано плотностью нормального распределения.

ГЕРТ-сеть, изображенная на рис. 5.22, *б*, состоит из узлов, соответствующих началу и завершению каждой отдельной работы, и дуг, представляющих действительное время выполнения

каждой работы. В табл. 5.4 даны W-функции для дуг рассматриваемой ГЕРТ-сети. Перед тем как перейти к поиску петель сети, изображенной на рис. 5.22, *б*, следует произвести три преобразования, существенно упрощающих сеть. Соответствующие

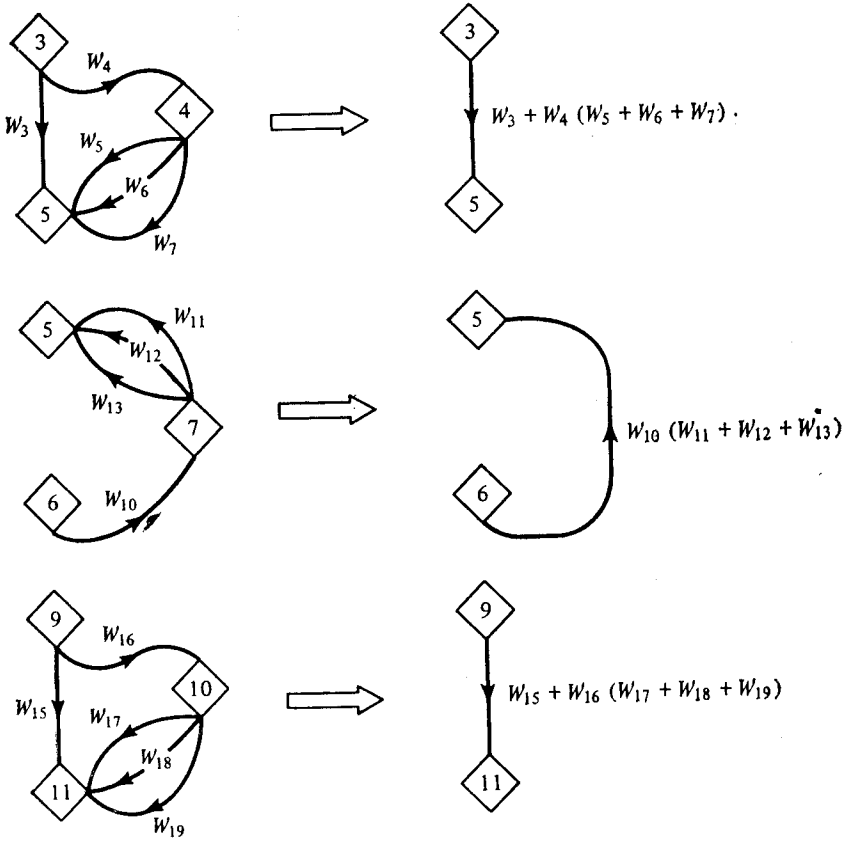


Рис. 5.23. Преобразования, упрощающие ГЕРТ-сеть.

замены показаны на рис. 5.23. Таким образом, мы построили эквивалентную сеть (рис. 5.24), для которой существенно упростился поиск петель.

По сети, изображенной на рис. 5.24, определяем следующие эквивалентные коэффициенты пропускания петель первого и второго порядка.

Петли первого порядка: W_{24}, $W_8W_{10}(W_{11}+W_{12}+W_{13})$, $W_1W_2W_8W_9W_{14}W_{20}W_{22}[W_3+W_4(W_5+W_6+W_7)][W_{15}+W_{16}(W_{17}+$

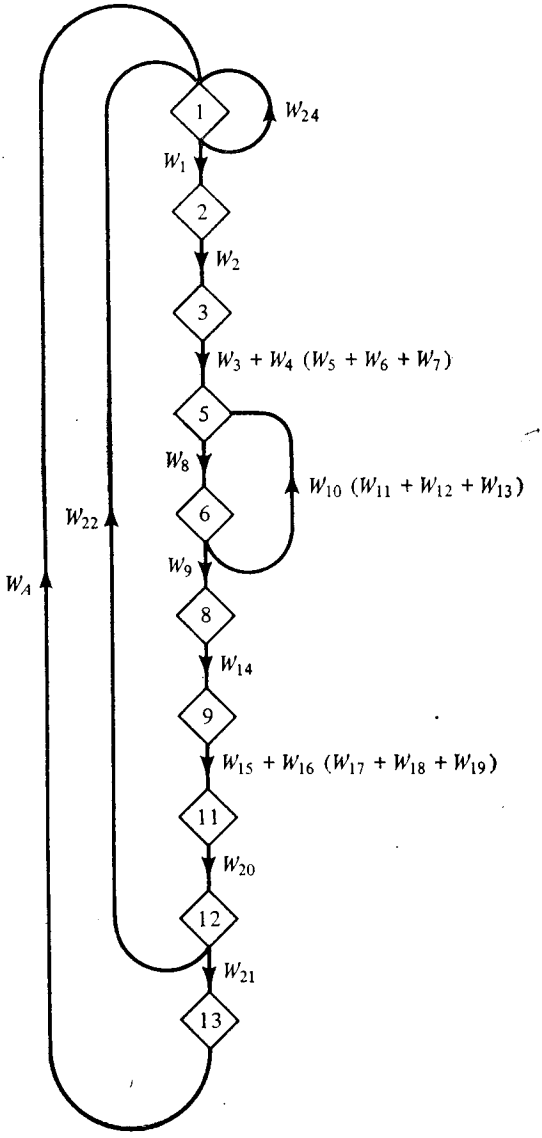


Рис. 5.24. Упрощенная ГЕРТ-сеть.

$+W_{18}+W_{19})]$, $W_1W_2W_8W_9W_{14}W_{20}W_{21}(1/W_E)[W_3+W_4(W_5+W_6+W_7)][W_{15}+W_{16}(W_{17}+W_{18}+W_{19})]$.

Петля второго порядка: $W_8W_{10}W_{24}(W_{11}+W_{12}+W_{13})$. Используя топологическое уравнение, получаем следующую эквивалентную W-функцию:

$$W_E(s)=\frac{W_1W_2W_8W_9W_{14}W_{20}W_{21}[W_3+W_4(W_5+W_6+W_7)]}{1-W_{24}-W_8W_{10}(W_{11}+W_{12}+W_{13})-W_1W_2W_8W_9W_{14}W_{20}W_{22}}$$
$$\frac{[W_{15}+W_{16}(W_{17}+W_{18}+W_{19})]}{[W_3+W_4(W_5+W_6+W_7)][W_{15}+W_{16}(W_{17}+W_{18}+W_{19})]}.$$

Вычисления, необходимые для нахождения математического ожидания и дисперсии, мы опускаем и предоставляем провести их в качестве упражнения читателю.

Отметим, однако, что $W_E(s)$ — это производящая функция моментов для дуги (13,1), а поскольку последняя является функцией только переменной преобразования s, то первые два центральных момента μ_1 и μ_2 относительно начала координат могут быть получены путем дифференцирования по s функции $W_E(s)$ и вычисления первой и второй производных при $s=0$. Поскольку μ_1 — это ожидаемая величина нормативного времени, а $\mu_2-(\mu_1)^2$ по определению есть дисперсия этого норматива, то требуемый результат нами получен. Проводя все необходимые вычисления, можно показать, что $\mu_1=14{,}032$ мин, а $\sigma^2=7{,}15$ мин^2.

Отметим, что данный результат дает много полезной информации о нормативе. Используя неравенство Чебышева, можно показать, что фактическое время изготовления одного изделия в 89% случаев изменяется в пределах от 6,104 мин до 21,96 мин. В рассматриваемом примере могут быть получены более сильные утверждения. Поскольку время выполнения каждой операции имеет нормальное распределение, то нормативное время также нормально распределено. Это позволяет получить вероятностные оценки времени выполнения задания. Кроме того, если норматив не соответствует этим оценкам, то можно построить ряд критериев для проверки гипотезы, позволяющих определить нормативы в будущем. Интересно отметить, что ГЕРТ является своеобразной альтернативой традиционным методам определения нормативных времен. При использовании традиционных методов предполагается, что время выполнения каждой отдельной работы *постоянно*. После суммирования этих времен в полученный результат вносится некоторая поправка с целью учесть случайные колебания или устранить неустойчивость действительных времен обслуживания. С другой стороны, система ГЕРТ позволяет включать случайные отклонения и неопреде-

ленность, возникающие *непосредственно* во время выполнения каждой отдельной работы. Следовательно, в полученный норматив *уже* включены все случайные колебания и нет необходимости вносить в него дополнительные поправки, не считая тех, которые соответствуют нормам времени на личные нужды и на усталость. Это дает возможность получить дисперсию нормативного времени, с помощью которой для него строятся доверительные интервалы.

В заключение отметим, что, хотя для решения данного примера потребовались сложные и трудоемкие вычисления, все они могут быть выполнены на ЭВМ. Для этого разработана специальная ГЕРТ-процедура, написанная на языке ФОРТРАН IV (см. [33]). Для обращения к ней необходимо пробить на стандартных картах данные о каждой непрерывной плотности распределения.

ЧАСТЬ III. МНОГОПРОДУКТОВЫЕ ПОТОКИ В СЕТЯХ

Результаты получены при участии
Дж. Эванса,
Университет г. Цинциннати

В гл. 3 было показано, каким мощным средством для решения ряда детерминированных потоковых задач является алгоритм дефекта. При рассмотрении каждого примера делалось одно важное предположение, согласно которому по дугам сети должен протекать однопродуктовый поток. Однако существует немало практических задач о перевозке нескольких различных продуктов, которые должны сохраняться при прохождении по дугам сети. Необходимо иметь возможность различать продукты. Рассмотрим, например, упрощенную задачу транспортировки трех видов фруктов из садов, расположенных в Калифорнии, в магазины, ведущие оптовую торговлю. Предположим, что сады, принадлежащие частной компании, расположены в городах Санта-Барбара, Бейкерсфилд и Сакраменто. В период уборки урожая из этих садов поставляются в различных количествах

Таблица 5.5. Производительность (ящики в неделю)

	Апельсины	Лимоны	Лаймы
Санта-Барбара	800	700	700
Бейкерсфилд	1000	800	500
Сакраменто	500	1000	500

Таблица 5.6. Величины спроса (ящики в неделю)

	Апельсины	Лимоны	Лаймы
Денвер	900	900	700
Сиэтл	700	1000	500
Канзас-Сити	700	600	500

Таблица 5.7. Пропускная способность железных дорог (ящики в неделю)

	В		
Из	Денвер	Сиэтл	Канзас-Сити
Санта-Барбара	1200	900	1000
Бейкерсфилд	1200	1000	1100
Сакраменто	950	1300	1000

Таблица 5.8. Транспортные затраты (центы на ящик)

	В		
Из	Денвер	Сиэтл	Канзас-Сити
Санта-Барбара			
Апельсины	3,2	2,5	5,6
Лимоны	2,4	1,9	4,3
Лаймы	2,3	1,7	4,0
Бейкерсфилд			
Апельсины	3,0	2,1	5,5
Лимоны	2,3	1,7	4,4
Лаймы	2,0	1,4	4,3
Сакраменто			
Апельсины	3,1	2,0	5,7
Лимоны	2,2	1,6	4,8
Лаймы	2,0	1,5	4,3

апельсины, лимоны и лаймы (табл. 5.5). По договорному соглашению требуется доставлять в оптовые магазины, расположенные в Денвере, Сиэтле и Канзас-Сити, такое количество фруктов, которое указано в табл. 5.6. Перевозка осуществляется по железной дороге, однако в каждом поезде для фруктов

отведено ограниченное место. Пропускные способности путей между различными пунктами отправления и пунктами назначения приведены в табл. 5.7. Независимо от вида фруктов данные величины задаются числом ящиков, перевозимых в неделю (предполагается, что все фрукты упакованы в ящики одинаковых размеров). Вес каждого ящика, а следовательно, и соответствующие транспортные затраты зависят от вида упакованных в него фруктов. Затраты на транспортировку одного ящика приведены в табл. 5.8.

Задача определения схемы перевозки фруктов, минимизирующей транспортные затраты, является задачей о многопродуктовом потоке. Отметим, что данная задача аналогична транспортной задаче, рассмотренной в разд. 2.10, в том отношении, что перевозки могут осуществляться только между пунктами отправления и пунктами назначения. Характерная особенность задач о многопродуктовом потоке состоит в том, что по дугам сети протекает несколько неоднородных потоков. При этом суммарная величина потоков всех продуктов по дуге ограничена ее пропускной способностью. Если бы это было не так, то можно было бы для каждого продукта независимо решить минимизационную транспортную задачу, используя, например, алгоритм дефекта. Однако указанные выше ограничения на пропускные способности дуг существенно усложняют решение задачи.

5.18. ФОРМУЛИРОВКИ ЗАДАЧ О МНОГОПРОДУКТОВОМ ПОТОКЕ В ВИДЕ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Задачи о многопродуктовом потоке, так же как и задачи об однопродуктовом потоке, могут быть сформулированы в виде задач линейного программирования. Задача о транспортировке фруктов является примером многопродуктовой транспортной задачи. Обозначим через x^k_{ij} поток k-го продукта из i-го источника в j-й сток, а через c^k_{ij} — стоимость транспортировки единицы этого продукта. Пусть далее $a_i{}^k$ и $b_j{}^k$ — это соответственно предложение узла i и спрос узла j для k-го продукта, а u_{ij} — пропускная способность дуги (i, j). Математическая постановка многопродуктовой транспортной задачи выглядит следующим образом:

$$\text{минимизировать} \sum_{k=1}^{r} \sum_{i=1}^{m} \sum_{j=1}^{n} c^k{}_{ij} x^k{}_{ij} \qquad (5.23)$$

при условии, что

$$\sum_i x^k{}_{ij} = b^k{}_j \quad \text{для всех } j, k, \tag{5.24}$$

$$\sum_j x^k{}_{ij} = a^k{}_i \quad \text{для всех } i, k, \tag{5.25}$$

$$\sum_k x^k{}_{ij} \leqslant u_{ij} \quad \text{для всех } i, j, \tag{5.26}$$

$$x^k{}_{ij} \geqslant 0 \quad \text{для всех } i, j, k. \tag{5.27}$$

Как и в однопродуктовой транспортной задаче, предполагается, что для каждого продукта суммарное предложение равно суммарному спросу, т. е.

$$\sum_i a^k{}_i = \sum_j b^k{}_j \quad \text{для всех } k. \tag{5.28}$$

Во многих задачах, называемых многопродуктовыми задачами о перевозках, вводятся промежуточные узлы, т. е. такие узлы, которым не приписывается ни предложение, ни спрос, а только требуется, чтобы для них выполнялось условие сохранения потока. В качестве одного из таких примеров рассмотрим сеть, изображенную на рис. 5.25. Узлы 1 и 2 являются источниками продукта 1, а узел 2 — источником продукта 2. Узел 5 является пунктом назначения для обоих продуктов, а узлы 3 и 4 — промежуточными. Многопродуктовая задача о перевозках может быть сформулирована в виде следующей задачи линейного программирования:

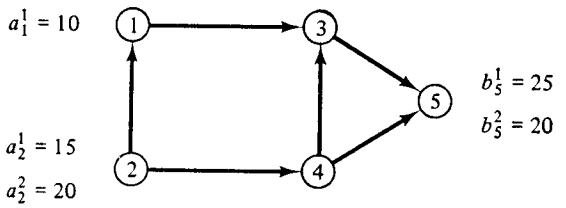


Рис. 5.25. Сеть в многопродуктовой задаче о перевозках.

$$\text{минимизировать} \sum_{k=1}^{r} \sum_{(i,j) \in A} c^k{}_{ij} x^k{}_{ij} \tag{5.29}$$

при условии, что

$$\sum_j x^k{}_{ij} - \sum_j x^k{}_{ji} = a^k{}_i, \quad \text{если узел } i \text{ является источником продукта } k, \tag{5.30}$$

$$\sum_j x^k{}_{ij} - \sum_j x^k{}_{ji} = 0, \quad \text{если узел } i \text{ является промежуточным узлом,} \tag{5.31}$$

$$\sum_i x^k{}_{ij} - \sum_i x^k{}_{ji} = -b^k{}_j, \quad \text{если узел } j \text{ является стоком продукта } k, \tag{5.32}$$

$$\sum_k x^k{}_{ij} \leqslant u_{ij} \quad \text{для всех } (i, j) \in \mathbf{A}, \tag{5.33}$$

$$x^k{}_{ij} \geqslant 0 \quad \text{для всех } k \text{ и } (i, j) \in \mathbf{A}. \tag{5.34}$$

Через **A** здесь обозначается множество дуг сети.

Одна из трудностей решения задач о многопродуктовом потоке вызвана тем, что оптимальные решения могут быть нецелочисленными. В качестве примера рассмотрим двухпродуктовую транспортную задачу, сетевая формулировка которой задана на рис. 5.26. Дугам сети приписаны числа $c^1{}_{ij}$, $c^2{}_{ij}$, u_{ij}. Оптимальное решение соответствующей задачи линейного программирования представлено в табл. 5.9.

Таблица 5.9

Дуга	Продукт 1	Продукт 2
(1, 1)	$\frac{3}{2}$	$\frac{1}{2}$
(1, 2)	0	$\frac{3}{2}$
(1, 3)	$\frac{1}{2}$	0
(2, 1)	0	$\frac{3}{2}$
(2, 2)	2	0
(2, 3)	0	$\frac{1}{2}$
(3, 1)	$\frac{1}{2}$	0
(3, 2)	0	$\frac{1}{2}$
(3, 3)	$\frac{3}{2}$	$\frac{3}{2}$

Нецелочисленные решения возникают по той причине, что матрицы ограничений в задачах линейного программирования, вообще говоря, не являются унимодулярными. Напомним, что условие унимодулярности матрицы ограничений, введенное в гл. 2, является достаточным для существования целочисленных оптимальных решений. Поскольку в задачах об однопродуктовом потоке данное условие выполнено, то для них можно разработать высокоэффективные алгоритмы, в которых по существу выполняются только две операции — сложение и вычитание, а такие операции, как, например, деление и обращение матрицы, не требуются. С вычислительной точки зрения арифметические операции, выполняемые с целыми числами, являются намного более быстрыми, чем операции, использующие числа (десятич-

ные) с плавающей точкой. Благодаря этому создаются программы, позволяющие очень быстро решать задачи большой размерности.

Для задач о многопродуктовом потоке были разработаны специальные алгоритмы, в которых используется специфика структуры и свойства данного класса задач. Было показано,

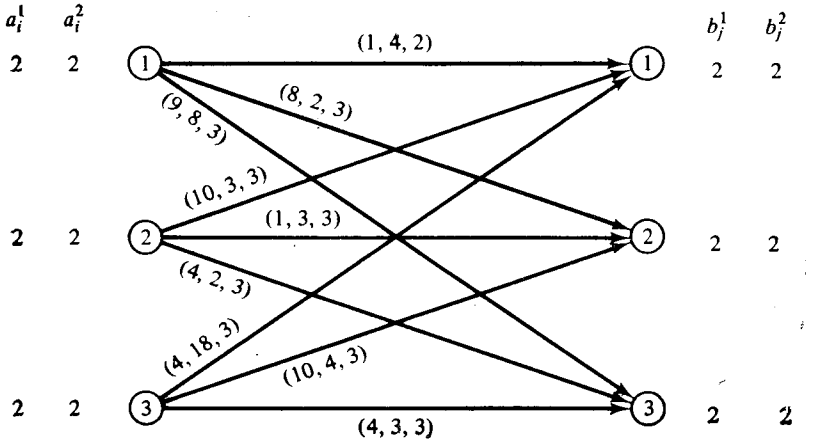


Рис. 5.26. Двухпродуктовая транспортная задача.

что эти алгоритмы являются более быстрыми, чем процедуры решения общей задачи линейного программирования, но значительно более медленными, чем соответствующие алгоритмы для задач об однопродуктовом потоке. Детальное изучение этих методов выходит за рамки нашей книги. Однако в настоящей главе мы рассмотрим ряд специальных вопросов, связанных с задачами о многопродуктовом потоке и методами их решения.

5.19. СПЕЦИАЛЬНЫЙ КЛАСС ЦЕЛОЧИСЛЕННЫХ ЗАДАЧ О МНОГОПРОДУКТОВОМ ПОТОКЕ

Как отмечалось выше, общая задача о многопродуктовом потоке может не иметь целочисленного оптимального решения. Однако существует несколько классов этих задач, в которых матрицы ограничений являются унимодулярными, и, следовательно, целочисленные оптимальные решения существуют. Интересно отметить, что многие из этих задач могут быть сведены к эквивалентным задачам об однопродуктовом потоке, которые несложно решаются с помощью алгоритма дефекта или любого другого алгоритма решения задачи о потоке минимальной стоимости.

Многопродуктовая транспортная задача была сформулирована в разд. 5.18. Для нее известен следующий результат.

Матрица ограничений в многопродуктовой транспортной задаче является унимодулярной в том и только в том случае, когда число источников (m) или число стоков (n) не превосходит 2.

Этот результат гарантирует существование целочисленного оптимального решения в случае, когда $m \leqslant 2$ или $n \leqslant 2$ *независимо* от числа продуктов. Очевидно, что если m или n равно 1, то решение тривиальное. Однако когда и m, и n превосходят 1, то возникает более общая задача линейного программирования. Покажем, что в этом случае существует более простой метод решения.

Вводя слабые переменные в ограничения на пропускные способности дуг, переформулируем многопродуктовую транспортную задачу (5.23)—(5.27) следующим образом:

$$\text{минимизировать} \sum_{k=1}^{r} \sum_{i=1}^{m} \sum_{j=1}^{n} c^k{}_{ij} x^k{}_{ij} \tag{5.35}$$

при условии, что

$$\sum_i x^k{}_{ij} = b^k{}_j \quad \text{для всех } j, k, \tag{5.36}$$

$$\sum_j x^k{}_{ij} = a^k{}_i \quad \text{для всех } i, k, \tag{5.37}$$

$$\sum_k x^k{}_{ij} + s_{ij} = u_{ij} \quad \text{для всех } i, j, \tag{5.38}$$

$$x^k{}_{ij},\ s_{ij} \geqslant 0 \quad \text{для всех } i, j, k. \tag{5.39}$$

При $m=2$ данная задача сводится к следующей задаче линейного программирования:

$$\text{минимизировать} \sum_{k=1}^{r} \sum_{j=1}^{n} [(c^k{}_{2j} - c^k{}_{1j}) x^k{}_{2j} + c^k{}_{1j} b^k{}_j] \tag{5.40}$$

при условии, что

$$\sum_k x^k{}_{2j} + s_{2j} = u_{2j} \quad \text{для всех } j, \tag{5.41}$$

$$-\sum_j x^k{}_{2j} = -a_2{}^k \quad \text{для всех } k, \tag{5.42}$$

$$-\sum_j s_{2j} = -\sum_j u_{2j} + \sum_k a_2{}^k. \tag{5.43}$$

$$x^k{}_{2j} + x^k{}_{1j} = b^k{}_j \quad \text{для всех } j, k, \tag{5.44}$$

$$s_{2j} + s_{1j} = u_{2j} + u_{1j} - \sum_k b^k{}_j \quad \text{для всех } j, \tag{5.45}$$

$$x^k{}_{ij},\ s_{ij} \geqslant 0 \quad \text{для всех } i, j, k. \tag{5.46}$$

Нетрудно заметить, что в равенствах (5.41)—(5.43) каждая из переменных $x^k{}_{2j}$ и s_{2j} появляется дважды: один раз со знаком плюс и один раз со знаком минус. Поэтому матрица коэффициентов этих трех ограничений имеет вид матрицы инциденций узел-дуга. Отметим также, что переменные $x^k{}_{1j}$ и s_{1j} входят только в равенства (5.44) и (5.45) (со знаком плюс). Поэтому можно рассматривать эти переменные как слабые и исключить их, записав (5.44) и (5.45) в следующем виде:

$$x_{2j} \leqslant b^k{}_j \quad \text{для всех } j, k,$$

$$s_{2j} \leqslant u_{2j} + u_{1j} - \sum_k b^k{}_j \quad \text{для всех } j.$$

Величины, стоящие в правых частях равенств (5.41)—(5.43), представляют собой предложение (если они положительные) и

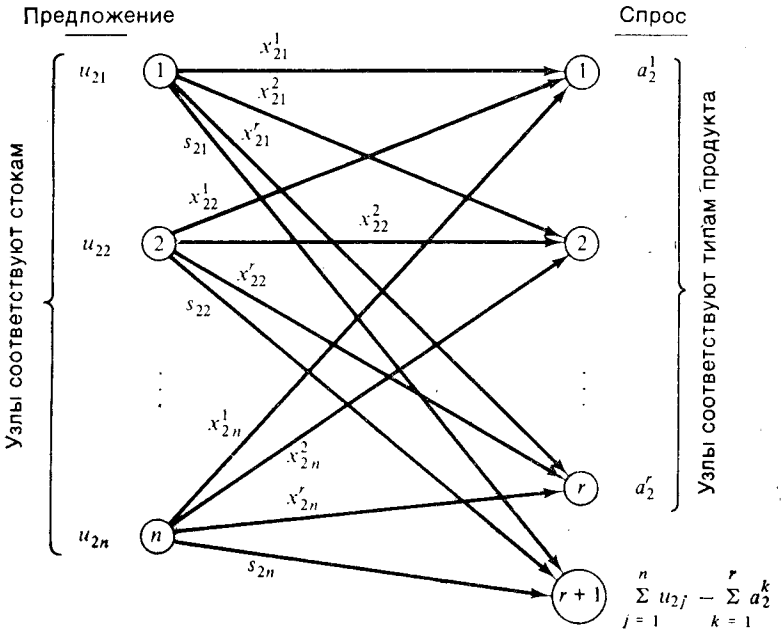


Рис. 5.27. Сеть в многопродуктовой транспортной задаче, сведенной к задаче об однопродуктовом потоке.

спрос (если отрицательные). Равенства (5.41)—(5.43) описывают транспортную модель, сетевая структура которой показана на рис. 5.27. Правые части равенств (5.44) и (5.45) представляют собой пропускные способности дуг, соответствующих переменным x^k_{2j} и s_{2j}. Для решения данной задачи можно использовать алгоритм дефекта, который позволит определить оптимальные потоки x^k_{2j} и s_{2j}. Затем из (5.44) и (5.45) могут быть найдены величины x^k_{1j} и s_{1j}.

Данная формулировка была получена с помощью преобразования строк в исходных ограничениях аналогично тому, как это делалось при рассмотрении задачи о календарном планировании трудовых ресурсов (разд. 2.1.7). Нетрудно показать, что эти две задачи эквивалентны. Интересно отметить, что исходная задача линейного программирования не имела форму сетевой задачи. Однако с помощью преобразования ограничений нам удалось свести ее к сетевой задаче и тем самым существенно упростить решение.

5.19.1. ТРАНСПОРТИРОВКА КОРОБОК ПЕРЕДАЧ ДЛЯ АВТОМОБИЛЕЙ

В распоряжении автомобилестроительной компании находятся два завода, производящие коробки передач; один из них — в Цинциннати, другой — в Сент-Луисе. Собранные коробки пере-

Таблица 5.10. Входная информация для модельной задачи

СТОИМОСТИ ТРАНСПОРТИРОВКИ ЕДИНИЦЫ ПРОДУКЦИИ

Из	Тип коробки передачи	В			Производительность (единицы/месяц)
		Детройт	Даллас	Атланта	
Цинциннати	СК	4,65	5,50	5,00	4000
	К	4,95	5,90	5,45	5000
	СР	5,30	6,30	6,00	2000
Сент-Луис	СК	5,15	4,85	5,25	2500
	К	5,70	4,95	5,90	3000
	СР	6,00	5,15	6,25	5000
Спрос (единицы/месяц)	СК	3500	1500	1500	
	К	4000	1500	2500	
	СР	3000	2000	2000	

Из	В		
	Детройт	Даллас	Атланта
Цинциннати	6000	3000	4000
Сент-Луис	6000	4000	3000

дач должны быть перевезены на сборочные заводы, расположенные в Детройте, Далласе и Атланте. Производятся три вида коробок передач: субкомпактные (СК), компактные (К) и средних размеров (СР). Величины производительности, спроса и транспортных затрат для каждого вида коробок передач, а также пропускные способности дорог приводятся в табл. 5.10. На рис. 5.28 изображена преобразованная сеть, для которой применим алгоритм дефекта.

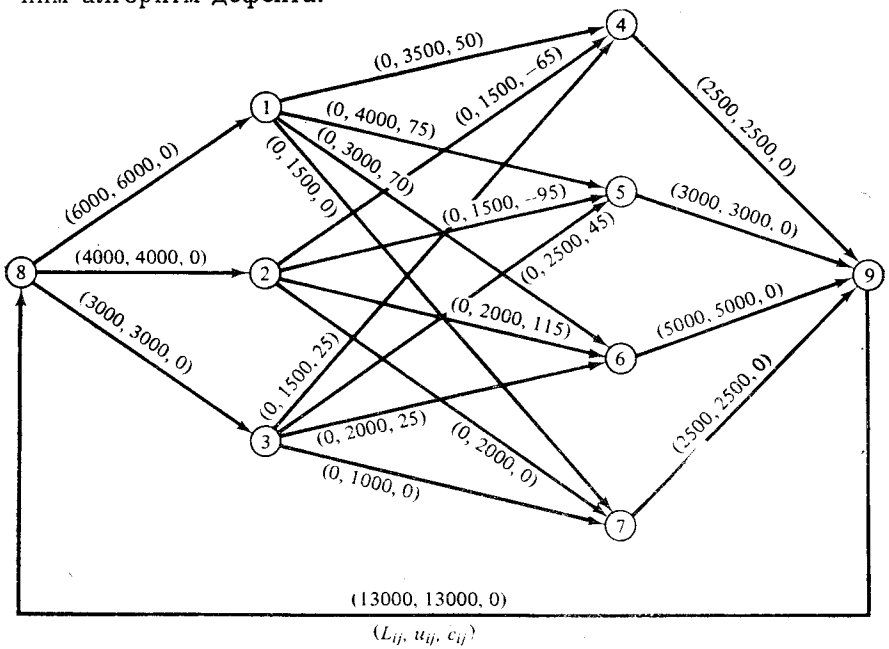


Рис. 5.28. Сеть, для которой применим алгоритм дефекта (все стоимости домножены на 100).

5.20. ПРИБЛИЖЕННОЕ РЕШЕНИЕ МНОГОПРОДУКТОВОЙ ТРАНСПОРТНОЙ ЗАДАЧИ МЕТОДОМ АГРЕГИРОВАНИЯ

Нередко лицо, принимающее решение, удовлетворяют хорошие, хотя и субоптимальные, решения сложных задач. Как правило, это происходит в тех случаях, когда отсутствуют программы, позволяющие проводить точную оптимизацию, или когда использование таких программ требует слишком больших затрат из-за циклического обращения к ним, или же когда нет возможности усовершенствовать основное программное обеспечение вследствие жестких ограничений на время ответа.

Одним из методов, позволяющих упростить решение задачи, является метод *агрегирования*, заключающийся в замене мно-

жества объектов (таких, как переменные, узлы сети и т. п.) одним объектом. В этом случае сокращаются как время решения задачи, так и требуемая машинная память, однако полученное решение, как правило, не является оптимальным.

В настоящем разделе, используя результаты, полученные в разд. 5.19, мы опишем метод решения многопродуктовых транспортных задач, основанный на агрегировании источников или стоков. А именно, мы рассмотрим случаи, когда сеть в агрегированной задаче содержит два источника и поэтому ее решение сводится к решению задачи об однопродуктовом потоке.

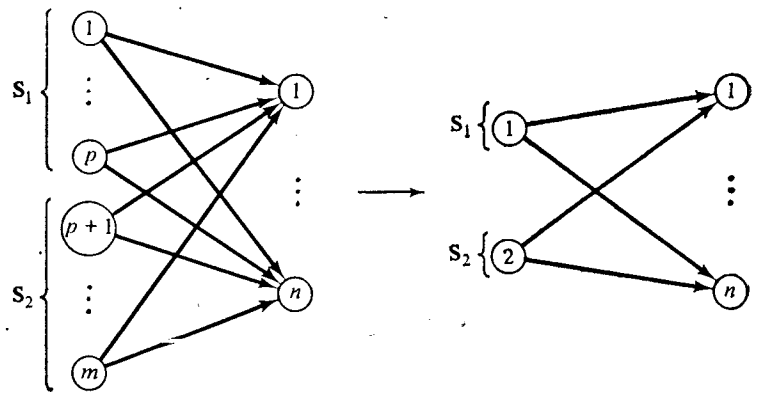


Рис. 5.29. Агрегирование источников в многопродуктовой транспортной задаче.

Будем рассматривать многопродуктовую транспортную задачу с m источниками, n стоками и r продуктами. Разобъем множество источников на два подмножества — S_1 и S_2, и заменим эти подмножества узлов двумя «фиктивными узлами», как показано на рис. 5.29. Тем самым мы построили сеть с двумя источниками для агрегированной задачи. Далее, необходимо установить взаимосвязь между параметрами агрегированной и исходной задач. Кроме того, нужно описать способ построения допустимого решения исходной задачи с помощью решения агрегированной задачи.

Поскольку каждый из двух фиктивных узлов представляет собой совокупность источников в исходной задаче, то величины предложения этих двух узлов естественно определить как

$$\bar{a}_l^k = \sum_{i \in S_l} a_i^k, \tag{5.47}$$

где $l=1, 2$. Пусть y^k_{lj} — поток продукта k из узла l в узел j в агрегированной сети. Поскольку дуга (l, j) получена в ре-

зультате агрегирования дуг, ведущих из узлов $i \in S_l$ в сток j, то

$$y^k{}_{lj} = \sum_{i \in S_l} x^k{}_{ij}. \tag{5.48}$$

Теперь нужно решить вопрос о стоимостях и пропускных способностях агрегированных дуг. Существует несколько способов их задания. Метод, который будет нами использован, имеет ряд преимуществ по сравнению с другими методами. Для определения стоимостей мы воспользуемся понятием *взвешенного агрегирования*. Стоимости взвешиваются пропорционально величинам предложения источников, представленных фиктивными узлами:

$$\bar{c}^k{}_{lj} = \sum_{i \in S_l} c^k{}_{ij} (a^k{}_i / \bar{a}^k{}_l). \tag{5.49}$$

Что касается пропускной способности агрегированной дуги (l, j), то на первый взгляд кажется разумным определить ее как сумму пропускных способностей исходных дуг из источников $i \in S_l$ в сток j. Однако в этом случае могут возникнуть трудности при построении допустимого решения исходной задачи. Поэтому мы воспользуемся следующим методом. Определим величины

$$\delta_i = \min_k [\bar{a}^k{}_l / a^k{}_i] \text{ для } i \in S_l. \tag{5.50}$$

Пусть далее

$$\bar{u}_{lj} = \min_{i \in S_l} [\delta_i u_{ij}]. \tag{5.51}$$

Сформулируем агрегированную задачу в виде следующей задачи линейного программирования:

$$\text{минимизировать } \sum_{k=1}^{r} \sum_{l=1}^{2} \sum_{j=1}^{n} \bar{c}^k{}_{lj} y^k{}_{lj} \tag{5.52}$$

при условии, что

$$\sum_l y^k{}_{lj} = b^k{}_j \quad \text{для всех } j, k, \tag{5.53}$$

$$\sum_j y^k{}_{lj} = \bar{a}^k{}_l \quad \text{для всех } l, k, \tag{5.54}$$

$$\sum_k y^k{}_{lj} \leqslant \bar{u}_{lj} \quad \text{для всех } l, j, \tag{5.55}$$

$$y^k{}_{lj} \geqslant 0 \quad \text{для всех } l, j, k. \tag{5.56}$$

Теперь необходимо построить решение исходной задачи, используя оптимальное решение агрегированной задачи (5.52)—(5.56). Определим величины x^k_{ij} следующим образом:

$$x^k_{ij} = y^k_{lj}(a^k_i/\bar{a}^k_l) \text{ для } i \in S_l. \tag{5.57}$$

Из (5.57) следует, что

$$\sum_{i \in S_l} x^k_{ij} = y^k_{lj}\left(\sum_{i \in S_l} a^k_i/\bar{a}^k_l\right) = y^k_{lj}(\bar{a}^k_l/\bar{a}^k_l) = y^k_{lj},$$

что совпадает с равенством (5.48). Процедура, определенная соотношением (5.57), называется *дезагрегированием с фиксированными весами.* С помощью несложных вычислений можно показать, что значения x^k_{ij} образуют допустимое решение исходной задачи (величины $\bar{u}_{lj}$ были определены по формуле (5.51) с тем, чтобы не нарушались ограничения на пропускные способности дуг). Кроме того,

$$\sum_k \sum_i \sum_j c^k_{ij} x^k_{ij} = \sum_k \sum_l \sum_j \bar{c}^k_{lj} y^k_{lj}. \tag{5.58}$$

Иными словами, значение целевой функции в результате дезагрегирования с фиксированными весами не изменяется. Однако следует помнить, что в силу определения величин $\bar{u}_{lj}$ в агрегированной задаче может не существовать допустимого решения, даже если в исходной задаче оно существует. В этом случае можно попытаться воспользоваться другими методами агрегирования.

5.20.1. ЗАДАЧА О ТРАНСПОРТИРОВКЕ ФРУКТОВ

Перейдем к решению задачи о транспортировке фруктов, сформулированной во введении к настоящей части. Агрегируем источники 2 и 3. Величины предложения, стоимости и пропускные способности для агрегированной задачи содержатся в табл. 5.11.

Оптимальное решение исходной задачи содержится в табл. 5.12. Его стоимость равна 18 570 долл. Стоимость оптимального решения агрегированной задачи (табл. 5.13) равна 18 799,10 долл, и на 0,26% превосходит стоимость действительного оптимального решения. В качестве упражнения читателю предлагается найти дезагрегированное решение и проверить, что его стоимость равна стоимости агрегированного решения.

Таблица 5.11. Параметры агрегированной задачи

ПРЕДЛОЖЕНИЯ

	Апельсины	Лимоны	Лаймы
Санта-Барбара	800	700	700
Бейкерсфилд/ Сакраменто	1500	1800	1000

СТОИМОСТИ

	в		
Из	Денвер	Сиэтл	Канзас-Сити
Санта-Барбара			
Апельсины	3,200	2,500	5,600
Лимоны	2,400	1,900	4,300
Лаймы	2,300	1,700	4,000
Бейкерсфилд/ Сакраменто			
Апельсины	3,033	2,067	5,567
Лимоны	2,244	1,644	4,622
Лаймы	2,000	1,450	4,300

ПРОПУСКНЫЕ СПОСОБНОСТИ

	в		
Из	Денвер	Сиэтл	Канзас-Сити
Санта-Барбара	1200	900	1000
Бейкерсфилд/ Сакраменто	1500	1800	1000

5.21. ГРАНИЦЫ ПОГРЕШНОСТИ ПРИ АГРЕГИРОВАНИИ[1)]

Рассматривая двойственную задачу по отношению к агрегированной задаче (5.52)—(5.56), мы получим границу погрешности для дезагрегированного решения. Данная величина используется при оценке погрешности, вызванной агрегированием. Однако не следует забывать, что действительная погрешность, вообще говоря, меньше этой границы.

Двойственная задача линейного программирования по отношению к агрегированной задаче формулируется следующим об-

[1)] Данный раздел может быть опущен без ущерба для понимания последующего материала.

Таблица 5.12. Решение задачи о транспортировке фруктов

	В		
Из	Денвер	Сиэтл	Канзас-Сити
Санта-Барбара			
Апельсины	800	0	0
Лимоны	200	0	500
Лаймы	0	200	500
Бейкерсфилд			
Апельсины	100	200	700
Лимоны	500	200	100
Лаймы	200	300	0
Сакраменто			
Апельсины	0	500	0
Лимоны	200	800	0
Лаймы	500	0	0

Таблица 5.13. Решение агрегированной задачи

	В		
Из	Денвер	Сиэтл	Канзас-Сити
Санта-Барбара			
Апельсины	800	0	0
Лимоны	200	0	500
Лаймы	0	400	300
Бейкерсфилд/ Сакраменто			
Апельсины	100	700	700
Лимоны	700	1000	100
Лаймы	700	100	200

разом:

$$\text{максимизировать} \sum_k \sum_l \bar{a}^k{}_l \alpha^k{}_l + \sum_k \sum_j b^k{}_j \beta^k{}_j - \sum_j \sum_l \bar{u}_{lj} \gamma_{lj} \quad (5.59)$$

при условии, что

$$\alpha^k{}_l + \beta^k{}_j - \gamma_{lj} \leqslant c_{lj} \quad \text{для всех } l, j, k, \quad (5.60)$$

$$\gamma_{lj} \geqslant 0 \quad \text{для всех } l, j. \quad (5.61)$$

Предположим, что $\bar{\alpha}$, $\bar{\beta}$ и $\bar{\gamma}$ образуют оптимальное решение двойственной задачи (5.59)—(5.61), а значение целевой функ-

ции равно $\bar{z}$. Пусть $\bar{x}$ — оптимальное решение исходной многопродуктовой транспортной задачи (5.23)—(5.27), а z^* — значение целевой функции. Тогда

$$z^* \geqslant \sum_k \sum_i \sum_j c^k{}_{ij}\bar{x}^k{}_{ij} + \sum_j \sum_k \Big(b^k{}_j - \sum_i \bar{x}^k{}_{ij}\Big)\bar{\beta}^k{}_j + \\ + \sum_l \sum_{i\in S_l} \sum_k \Big(a^k{}_i - \sum_j \bar{x}^k{}_{ij}\Big)\bar{\alpha}^k{}_l - \sum_j \sum_l \sum_{i\in S_l} \Big(u_{ij} - \sum_k \bar{x}^k{}_{ij}\Big)\bar{\gamma}_{ij}.$$

Прибавляя и вычитая в правой части полученного неравенства величину $\sum\limits_l \sum\limits_j \bar{u}_{lj}\bar{\gamma}_{lj}$, производя перегруппировку слагаемых и используя определение величин агрегированного предложения, получаем, что

$$z^* \geqslant \Big[\sum_k \sum_l \bar{a}^k{}_l\bar{\alpha}^k{}_l + \sum_k \sum_j b^k{}_j\bar{\beta}^k{}_j - \sum_l \sum_j \bar{u}_{lj}\bar{\gamma}_{lj}\Big] - \\ - \sum_l \sum_{i\in S_l} \sum_j \sum_k \bar{x}^k{}_{ij}(\bar{\alpha}^k{}_l + \bar{\beta}^k{}_j - \bar{\gamma}_{lj} - c^k{}_{lj}) + \\ + \sum_l \sum_j \Big(\bar{u}_{lj} - \sum_{i\in S_l} u_{ij}\Big)\bar{\gamma}_{lj}.$$

Отметим, что величина в квадратных скобках равна $\bar{z}$. Следовательно,

$$\bar{z} - z^* \leqslant \sum_l \sum_{i\in S_l} \max_{j,\,k} [\bar{\alpha}^k{}_l + \bar{\beta}^k{}_j - \bar{\gamma}_{lj} - c^k{}_{ij}] \sum_j \sum_k \bar{x}^k{}_{ij} - \\ - \sum_l \sum_j \Big(\bar{u}_{lj} - \sum_{i\in S_l} u_{ij}\Big)\bar{\gamma}_{lj}.$$

Поскольку $\sum\limits_j \sum\limits_k \bar{x}^k{}_{ij} = \sum\limits_k a^k{}_i$, то

$$\bar{z} - z^* \leqslant \sum_l \sum_{i\in S_l} \max_{j,\,k} [\bar{\alpha}^k{}_l + \bar{\beta}^k{}_j - \bar{\gamma}_{lj} - c^k{}_{ij}] \sum_k a^k{}_i - \\ - \sum_l \sum_j \Big(\bar{u}_{lj} - \sum_{i\in S_l} u_{ij}\Big)\bar{\gamma}_{lj} \triangleq \varepsilon.$$

Величина ε представляет собой максимальное отклонение стоимости агрегированного решения от действительно минимальной стоимости. Поскольку $\bar{z}$ является верхней границей минимальной стоимости, то $\bar{z} - \varepsilon \leqslant z^* \leqslant \bar{z}$.

5.22. МАКСИМАЛЬНЫЕ МНОГОПРОДУКТОВЫЕ ПОТОКИ

Для задач об однопродуктовом потоке было показано, что величина максимального потока равна величине минимального разреза. Кроме того, для этих задач известен изящный и эффективный алгоритм нахождения максимального потока. На первый взгляд может показаться, что эти результаты обобщаются на случай многопродуктовых потоков. Однако, если не считать

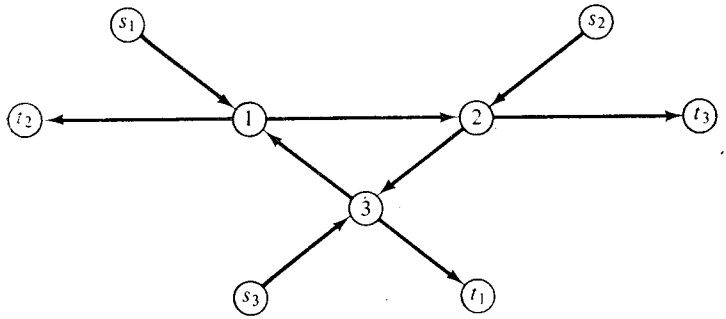


Рис. 5.30. Задача о максимальном многопродуктовом потоке.

некоторых частных примеров, это не так. Рассмотрим пример, изображенный на рис. 5.30. Узлы s_k и t_k являются соответственно источником и стоком продукта k; пропускная способность каждой дуги равна 1. Требуется максимизировать *сумму* пото-

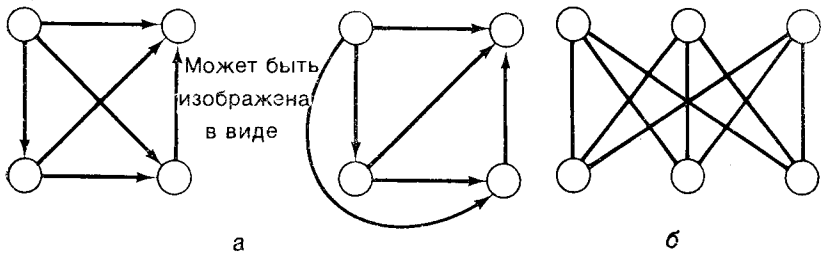


Рис. 5.31. Плоская и не являющаяся плоской сети.

ков трех продуктов из их источников в соответствующие стоки. Если, например, из s_1 в t_1 по единственно возможному пути посылается единица потока, то никакой другой продукт протекать по сети не сможет. Этот поток не является максимальным, так как можно из s_k в t_k послать 1/2 единиц потока продукта k.

В задачах о многопродуктовом потоке аналогом разреза является *разделяющее множество,* определяемое как совокупность дуг, исключение которых из сети разрывает все цепи из источников в соответствующие стоки. В рассматриваемом примере минимальное разделяющее множество состоит из любых двух

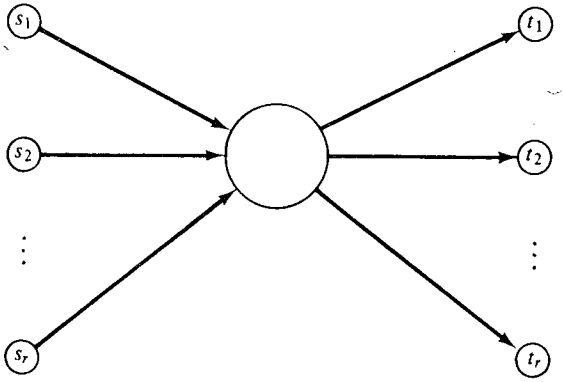


Рис. 5.32. Структура вполне плоских сетей.

дуг, соединяющих узлы 1, 2 и 3, и его величина равна 2. Следовательно, величина максимального многопродуктового потока может быть *меньше* величины минимального разделяющего множества.

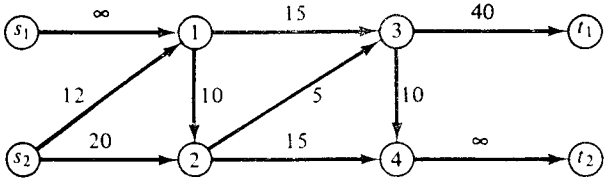


Рис. 5.33. Двухпродуктовая вполне плоская сеть.

Вообще задача о максимальном многопродуктовом потоке, так же как и задача о многопродуктовом потоке минимальной стоимости, является весьма сложной. Рассмотрение большинства алгоритмов решения данной задачи выходит за рамки нашей книги. Однако для специальных типов сетей максимальный поток равен величине минимального разделяющего множества и потоки по всем дугам являются целочисленными. Можно показать, что данный результат имеет место для сетей, называемых *вполне плоскими.* Сеть называется *плоской,* если она

может быть изображена таким образом, что никакие две ее дуги не пересекаются. Например, сеть, изображенная на рис. 5.31, *а*, является плоской, а сеть на рис. 5.31, *б* не является плоской. Сеть называется вполне плоской, если она может быть изображена так, как показано на рис. 5.32, так что полученная в результате сеть является плоской. Пример сети, по которой протекает двухпродуктовый поток, дан на рис. 5.33.

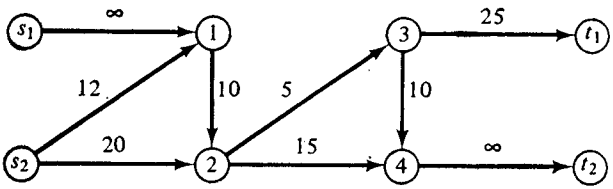


Рис. 5.34.

Существует очень простой алгоритм решения данного типа задач. Начать с рассмотрения продукта 1. Выбрать самую верхнюю цепь из s_1 в t_1 и приписать ей максимально возможный поток. По крайней мере одна дуга будет насыщенной. Исклю-

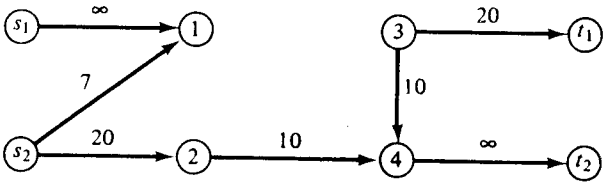
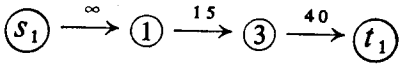


Рис. 5.35.

чить из сети эту дугу, а также все остальные дуги, поток по которым равен их пропускной способности. Изменить пропускную способность каждой дуги выбранной цепи, вычитая величину потока по ней. Если аугментальных цепей потока из s_1 в t_1 не существует, то повторять данную процедуру для каждого из продуктов 2, 3 и т. д. до тех пор, пока нельзя будет послать дополнительный поток из источника в сток. Полученный поток будет максимальным.

Найдем решение задачи, сеть для которой изображена на рис. 5.33. Самой верхней цепью из s_1 в t_1 является следующая цепь:



По этой цепи может протекать 15 единиц потока. Изменяя пропускные способности дуг и исключая насыщенную дугу, получаем сеть, изображенную на рис. 5.34. Далее в качестве самой верхней выберем цепь s_1—1—2—3—t_1 и припишем ей поток величиной 5 единиц. Новая сеть изображена на рис. 5.35. В моди-

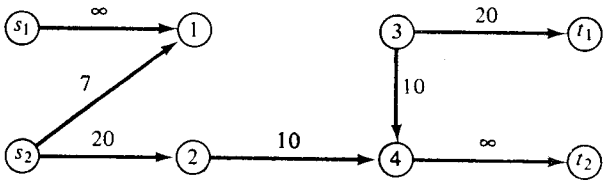
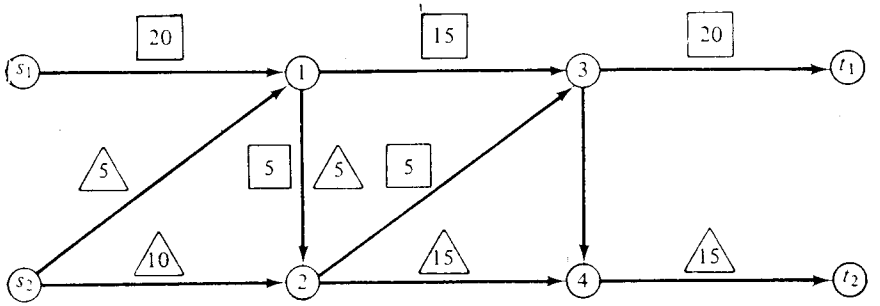


Рис. 5.36.



Продукт 1

Продукт 2

Рис. 5.37. Максимальный двухпродуктовый поток.

фицированной сети (рис. 5.35) не существует аугментальных путей потока из s_1 в t_1. Рассмотрим продукт 2. Выберем цепь s_2—1—2—4—t_2, по которой может протекать поток величиной 5 единиц. Модифицированная сеть изображена на рис. 5.36.

Наконец, по цепи s_2—2—4—t_2 может протекать поток величиной 10 единиц. С помощью полученных результатов определяется максимальный поток (рис. 5.37). Величина максимального двухпродуктового потока равна 35, а минимальное разделяющее множество задается дугами (1, 3), (2, 3) и (2, 4).

5.23. МНОГОПРОДУКТОВЫЕ ПОТОКИ В НЕОРИЕНТИРОВАННЫХ СЕТЯХ

В сетях с неориентированными дугами поток по дуге может протекать в любом направлении. В случае когда поток однопродуктовый, неориентированную дугу можно заменить двумя

противоположно направленными дугами. Это можно сделать благодаря тому, что потоки, протекающие по противоположным направлениям, поглощают друг друга. Однако в случае, когда поток многопродуктовый, такую замену произвести нельзя, так как потоки *различных продуктов,* протекающие по противоположным направлениям, не поглощают друг друга, а суммируются, и суммарная величина потока не должна превосходить пропускной способности дуги. Как и для ориентированных сетей, максимальные потоки могут быть нецелочисленными и, кроме того, теорема о максимальном потоке и минимальном разрезе

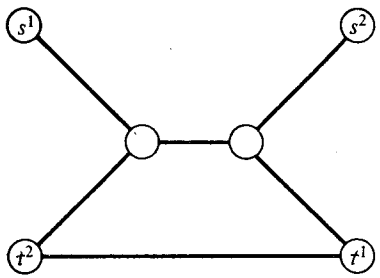


Рис. 5.38. Двухпродуктовая сеть.

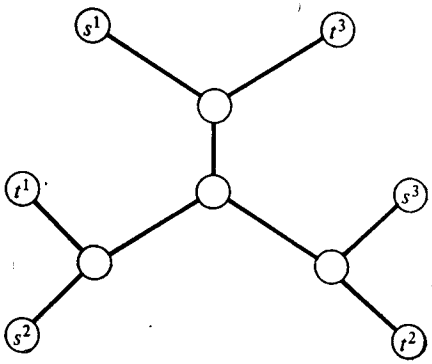


Рис. 5.39. Трехпродуктовая сеть. Пропускные способности всех дуг равны 1.

не всегда справедлива. В качестве примеров рассмотрим сети, изображенные на рис. 5.38 и 5.39. На рис. 5.38 изображена сеть, по которой протекает двухпродуктовый поток. Пропускные способности всех дуг равны 1. Читателю предлагается найти максимальный поток. (*Указание:* максимальный поток не является целочисленным.) На рис. 5.39 изображена сеть, по которой протекает трехпродуктовый поток. Читателю предлагается проверить, что величина максимального потока строго меньше величины минимального разделяющего множества.

Общая задача о многопродуктовом максимальном потоке является весьма сложной. Мы опишем алгоритмы решения специального класса задач о двухпродуктовом потоке. Можно показать, что оптимальные потоки по дугам в задачах из этого класса всегда кратны 1/2. Кроме того, величина максимального потока равна величине минимального разделяющего множества.

Для описания алгоритма необходимо ввести несколько новых понятий и определений. Поток продукта k по дуге (i, j) будем обозначать через f^k_{ij}. Поскольку поток может протекать по лю-

бому направлению, то положительный поток из i в j можно рассматривать как отрицательный поток из j в i. Следовательно, $f^k{}_{ij}=-f^k{}_{ji}$. Независимо от направления потока чистую величину продукта k, протекающего по дуге (i, j), будем обозначать через $|f^k{}_{ij}|$. Как и раньше, через s^k и t^k будем обозначать соответственно источник и сток продукта k, а через u_{ij} — пропускную способность дуги (i, j). Предполагается, что u_{ij} — целая величина.

Предположим, что по сети протекает некоторый поток, который удовлетворяет условию сохранения. По данному потоку построим два множества узлов **F** и **B**, следующим образом:

1. $s^2\in\mathbf{F}$. Если $i\in\mathbf{F}$ и $f^1{}_{ji}+f^2{}_{ij}<u_{ij}$, то $j\in\mathbf{F}$.
2. $s^2\in\mathbf{B}$. Если $i\in\mathbf{B}$ и $f^1{}_{ij}+f^2{}_{ij}<u_{ij}$, то $j\in\mathbf{B}$.

Предположим, что из i в j посылается дополнительный поток продукта 1. Если $f^1{}_{ij}>0$, то *остаточная пропускная способность* определяется следующим образом:

$$u^1{}_{ij}=u_{ij}-f^1{}_{ij}-|f^2{}_{ij}|. \tag{5.62}$$

Если дополнительный поток посылается из j в i и $f^1{}_{ij}>0$, то остаточная пропускная способность определяется как

$$u^1{}_{ij}=u_{ij}+f^1{}_{ij}-|f^2{}_{ij}|, \tag{5.63}$$

поскольку однородные потоки противоположных направлений поглощают друг друга. Аналогично если $f^2{}_{ij}>0$, то для продукта 2 остаточная пропускная способность в направлении от i к j равна

$$u^2{}_{ij}=u_{ij}-|f^1{}_{ij}|-f^2{}_{ij}, \tag{5.64}$$

а в направлении от j к i

$$u^2{}_{ij}=u_{ij}-|f^1{}_{ij}|+f^2{}_{ij}. \tag{5.65}$$

Рассмотрим произвольный путь, соединяющий s^2 с t^2 и содержащий дугу (i, j). Если при движении от s^2 к t^2 узел i достигается раньше, чем узел j, то говорят, что дуга (i, j) имеет *положительное направление* по отношению к данному пути. В этом случае также говорят, что поток протекает из i в j *в прямом направлении*, а из j в i — *в обратном направлении*.

Если $t^2\in\mathbf{B}$, то существует такой путь из s^2 в t^2, что $u_{ij}+f^1{}_{ji}-f^2{}_{ij}>0$ для всех дуг этого пути и положительное направление каждой дуги (i, j) — это направление от i к j. Будем называть этот путь *обратным путем* из s^2 в t^2. Аналогично если $t^2\in\mathbf{F}$, то $u_{ij}+f^1{}_{ij}-f^2{}_{ij}>0$ для всех дуг этого пути, который называется *прямым путем*. Пара, состоящая из прямого и обратного пу-

тей, называется *двойным путем.* Если положительным направлением всех дуг некоторого пути является направление от i к j, то *пропускная способность обратного пути* равна

$$\alpha_{\text{В}} = \min [u_{ij} + f^1_{ji} - f^2_{ij}], \tag{5.66}$$

а *пропускная способность прямого пути* равна

$$\alpha_{\text{F}} = \min [u_{ij} + f^1_{ij} - f^2_{ij}]. \tag{5.67}$$

Эти величины равны максимальным потокам, которые могут протекать по данным путям.

Пусть $\Delta^1{}_{\text{F}} = \min [f^1_{ij} > 0]$, где f^1_{ij} — поток прямого направления по дуге прямого пути; $\Delta^1{}_{\text{В}} = \min [f^1_{ij} > 0]$, где f^1_{ij} — поток обратного направления по дуге обратного пути.

Для каждого двойного пути введем обозначения

$$\Delta^1 = \min [\Delta^1{}_{\text{F}}, \Delta^1{}_{\text{В}}], \tag{5.68}$$

$$\alpha = \min [\alpha_{\text{В}}, \alpha_{\text{F}}]. \tag{5.69}$$

Перейдем к описанию алгоритма нахождения максимального двухпродуктового потока в неориентированной сети.

Шаг 0. С помощью алгоритма для однопродуктовых потоков найти максимальный поток продукта 1.

Шаг 1. Вычислить остаточные пропускные способности u^2_{ij} и определить максимальный поток продукта 2.

Шаг 2. Построить двойной путь из s^2 в t^2. Если такого пути не существует, то максимальный поток найден. В противном случае положить $\varepsilon = \min[\Delta^1, 1/2\alpha]$. Послать ε единиц продукта 1 из s^2 в t^2 по обратному пути и из t^2 в s^2 по прямому пути.

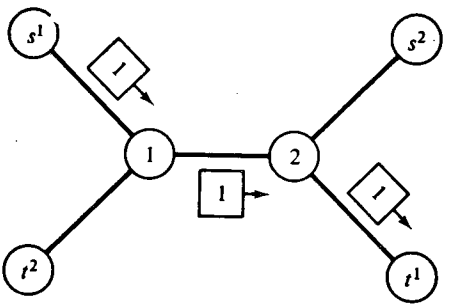
Шаг 3. Увеличить общий поток продукта 2, посылая ε единиц его из s^2 в t^2 по прямому и обратному путям.

По существу при работе данного алгоритма поток продукта 1 перераспределяется по двойному пути так, что поток продукта 2 по прямому и обратному путям может быть увеличен.

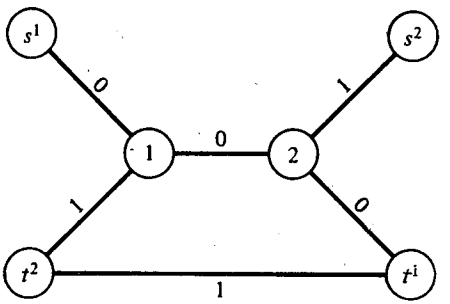
5.23.1. ПРИМЕР ЗАДАЧИ О ДВУХПРОДУКТОВОМ ПОТОКЕ

Решим задачу о двухпродуктовом потоке для сети, изображенной на рис. 5.38.

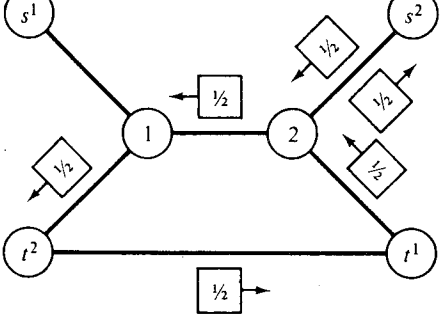
Шаг 0. Максимальный поток продукта 1 равен 1 (символом $\square\!\!\rightarrow$ обозначается продукт 1, а символом $\triangle\!\!\rightarrow$ — продукт 2).



Шаг 1. Ниже указаны остаточные пропускные способности u^2_{ij}. Максимальный поток из s^2 в t^2 равен 0.

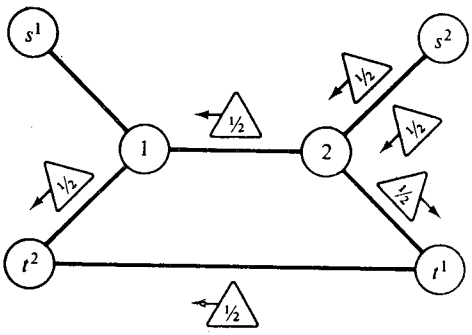


Шаг 2. Для построения двойных путей определим множества **F** и **B**: $\mathbf{F}=\{s^2, 2, t^1, t^2\}$, $\mathbf{B}=\{s^2, 2, 1, t^2\}$. Поскольку $t^2 \in \mathbf{F}$ и $t^2 \in \mathbf{B}$, то мы имеем пару двойных путей. Прямой путь задается последовательностью узлов s^2—2—t^1—t^2, и его пропускная способность равна $\alpha_{\mathbf{F}}=\min[1, 2, 1]=1$. Обратный путь задается последовательностью узлов s^2—2—1—t^2, и его пропускная способность равна $\alpha_{\mathbf{B}}=\min[1, 2, 1]=1$. Таким образом, $\alpha=\min[\alpha_{\mathbf{F}},$

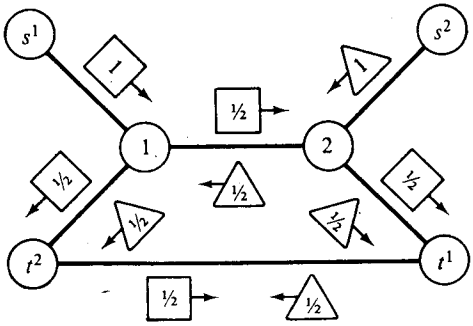


$\alpha_B] = 1$. Кроме того $\Delta^1{}_F = 1$, $\Delta^1{}_B = 1$. Поэтому $\Delta^1 = \min[\Delta^1{}_F, \Delta^1{}_B] = 1$ и $\varepsilon = 1/2$. Для завершения шага 2 следует послать 1/2 единиц продукта 1 из s^2 в t^2 по обратному пути и из t^2 в s^2 по прямому пути:

Шаг 3. Послать 1/2 единиц продукта 2 из s^2 в t^2 по обоим путям:



Текущий поток выглядит следующим образом:



Возвращаясь на шаг 2, мы определяем, что двойных путей не существует. Следовательно, полученное решение является оптимальным.

Отметим, что величина ε является либо целой, либо кратной 1/2. Поэтому величины всех потоков по дугам также целые или кратные 1/2. Если пропускные способности всех дуг задаются четными числами, то максимальный поток является целочисленным.

5.24. МАКСИМАЛЬНЫЕ ПОТОКИ И ВОРОНКООБРАЗНЫЕ УЗЛЫ

Нередко решаемая в области перевозок и снабжения, и в частности в военном деле, задача может быть описана следующим образом. Из пункта снабжения в пункт сбыта должен пе-

ревозиться товар. Транспортные средства, предназначенные для перевозки товара, располагаются по различным паркам. Предполагается, что транспортная сеть имеет ограниченную пропускную способность. Требуется определить оптимальный маршрут движения транспортных средств с мест их первоначального расположения к пункту снабжения и затем к пункту сбыта. Аналогичную структуру имеет также следующая задача. Для существующей коммуникационной сети требуется построить центральный пункт связи, через который проходила бы вся информация. Необходимо получить максимально возможный поток информации.

В обоих этих примерах выбирается специальный узел, через который должен протекать весь поток из источника в сток. Этот узел называется *воронкообразным.* Обозначим через f^1_{ij} поток из узла i в j, протекающий по направлению к воронкообразному узлу, а через f^2_{ij} — поток из i в j после прохождения воронкообразного узла. Тогда рассматриваемая задача может быть сформулирована следующим образом:

$$\text{минимизировать } v \tag{5.70}$$

при условии, что

$$\sum_j (f^1_{ij} - f^1_{ji}) = \begin{cases} v^1, & \text{если } i = s, \\ 0, & \text{если } i \neq s, a, \\ -v^1, & \text{если } i = a, \end{cases} \tag{5.71}$$

$$\sum_j (f^2_{ij} - f^2_{ji}) = \begin{cases} v^2, & \text{если } i = a, \\ 0, & \text{если } i \neq a, t, \\ -v^2, & \text{если } i = t, \end{cases} \tag{5.72}$$

$$|f^1_{ij}| + |f^2_{ij}| \leqslant u_{ij} \quad \text{для всех } i, j, \tag{5.73}$$

$$v = v^1 = v^2, \tag{5.74}$$

$$f^k_{ij} > 0. \tag{5.75}$$

Через s, a и t обозначены источник, воронкообразный узел и сток соответственно. Отметим, что данная задача аналогична задаче о двухпродуктовом потоке в неориентированной сети. Однако в рассматриваемом случае мы имеем не два вида продукта, а два различных типа потока. Кроме того, требуется, чтобы потоки «продуктов» 1 и 2 были равными. В задаче о двухпродуктовом потоке требуется максимизировать суммарный поток продуктов; теперь же максимизируется величина $\frac{1}{2}(v^1 + v^2)$.

Можно показать, что максимальный поток через воронкообразный узел равен $\min[\bar{v}^1, \bar{v}^2, \frac{1}{2}\max[v^1 + v^2]]$, где $\bar{v}^1$ и $\bar{v}^2$ — это максимальные потоки из s в a и из a в t соответственно. Дан-

ный результат лежит в основе простого алгоритма, в котором последовательно решаются обычные задачи о максимальном потоке.

Шаг 1. С помощью алгоритма решения задачи о максимальном однопродуктовом потоке найти величины $\bar{v}^1$ и $\bar{v}^2$.

Шаг 2. Построить новую сеть G', вводя дополнительный узел s' и дуги (s', s), $(s' t)$ с неограниченной пропускной способностью. Пусть $\bar{v}$ — максимальный поток из s' в a в построенной сети.

Шаг 3. Вычислить величину $v^* = \min[\bar{v}^1, \bar{v}^2, {}^1/_2 \bar{v}]$. Если $v^* = 0$, то алгоритм прекращает работу. Допустимого потока не существует.

Шаг 4. Построить новую сеть G'', добавляя к исходной сети узел s'' и дуги (s'', s) и (s'', t), пропускная способность каждой из которых равна v^*. Найти максимальный поток из s'' в a в построенной сети. Разложить этот поток на поток из s'' в a, протекающий через s, и поток из a в s'', протекающий через t. Исключить из сети дополнительные узлы и дуги. В результате будет найден максимальный поток, протекающий через воронкообразный узел a. Отметим, что на шаге 2 величина $\bar{v} = \max[v^1 + v^2]$ определяется путем решения задачи об однопродуктовом потоке. Для того чтобы различать между собой «продукты», можно каждый поток по пути из s' в a, проходящему через s, рассматривать как поток «продукта 1», а поток по пути из s' в a, проходящему через t, рассматривать как поток «продукта 2». На шаге 4 строится такой поток, что величина той его части, которая протекает через a, равна v^*.

5.25. ПРИЛОЖЕНИЯ ЗАДАЧ О МНОГОПРОДУКТОВОМ ПОТОКЕ

Задачи о многопродуктовом потоке находят широкое применение при проектировании коммуникационных систем, осуществлении железнодорожных перевозок, планировании производства и распределения, в военном деле, а также в других областях. В настоящем разделе мы остановимся на нескольких приложениях многопродуктовых сетевых моделей и дадим новые формулировки некоторых задач о многопродуктовом потоке.

5.25.1. СОСТАВЛЕНИЕ РАСПИСАНИЯ ДВИЖЕНИЯ СУДОВ

Одной из наиболее часто возникающих задач в области планирования перевозок является задача составления оптимального расписания движения транспортных средств. Предположим, что компания располагает фиксированным числом разнотипных судов, которые отличаются друг от друга скоростью передви-

жения, грузоподъемностью и эксплуатационными расходами. Функция полезности определяется размерами поставки груза отдельным судном к заданному сроку и затратами на осуществление соответствующей перевозки. Кроме того, существуют затраты (отрицательная полезность), связанные с перегоном судна из порта, в котором оно было разгружено, в порт погрузки. Задача заключается в максимизации полезности путем составления оптимального маршрута и оптимального расписания движения судов.

Данная задача может быть сформулирована в виде следующей сетевой задачи. Каждый узел j сети соответствует прибытию судна в порт назначения в один из допустимых сроков доставки. Каждый узел i соответствует исходному порту и моменту, равному разности срока доставки и времени транспортировки, которое по предположению является детерминированным. Дуга (i, j) соответствует перевозке груза, а дуга (j, i) — перегону судна из порта доставки в исходный порт. Суда k-го типа первоначально располагаются в источнике s^k, и дуги (s^k, i), таким образом, соответствуют началу их эксплуатации. Далее, вводятся фиктивный сток t и дуги (j, t), соответствующие завершению эксплуатации судов. И наконец, суммарный поток перевозимого груза по всем дугам, соответствующим различным датам перевозки и типам судов, не должен превосходить некоторой заданной величины. Математически эта задача формулируется следующим образом:

$$\text{максимизировать } z = \sum_k \sum_i \sum_j c^k{}_{ij} x^k{}_{ij} \tag{5.76}$$

при условии, что

$$\sum_k \sum_{A_v} r^k{}_{ij} x^k{}_{ij} \leqslant b_v, \tag{5.77}$$

$$\sum_i x^k{}_{ij} - \sum_j x^k{}_{ji} = \begin{cases} d^k, & i = s^k, \\ 0, & i \neq s^k, t, \\ -d^k, & i = t, \end{cases} \tag{5.78}$$

$$0 \leqslant x^k{}_{ij} \leqslant u^k{}_{ij}. \tag{5.79}$$

Здесь через $c^k{}_{ij}$ обозначена величина полезности, соответствующая отдельному судну, перевозимому грузу и маршруту. A_v — это множество допустимых маршрутов для данного груза; $r^k{}_{ij}$ — грузоподъемность судна k-го типа на маршруте $(i\ j)$. Суммарная величина перевозимого груза не может превышать спроса на него. Ограничения (5.78) описывают условие сохранения потока по числу судов, а величина $u^k{}_{ij}$ равна 0 или ∞ в зависимости от того, может ли быть использовано на дуге (i, j)

Таблица 5.14

Груз	Минимально допустимый момент начала перевозки	Допустимые сроки доставки
1	0	4, 5
2	2	6, 7, 8
3	3	7, 8

судно k-го типа или нет. В качестве примера рассмотрим задачу, описанную в табл. 5.14. Число судов каждого типа равно 2.

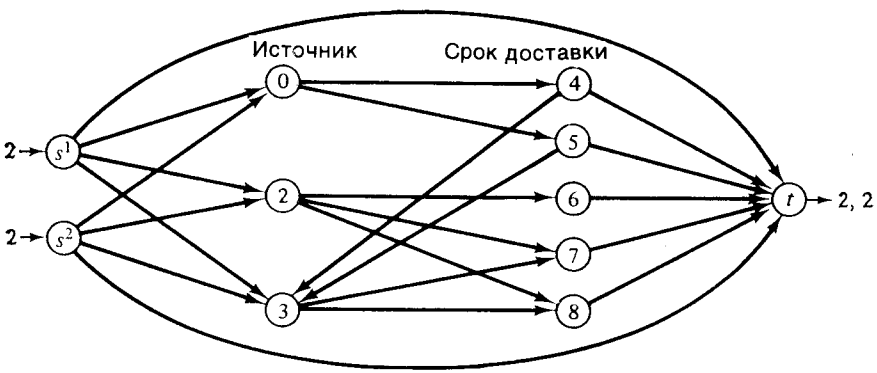


Рис. 5.40. Сеть в задаче составления расписания движения судов.

Соответствующая сеть изображена на рис. 5.40. Отметим, что в рассмотренном классе задач узлы служат для обозначения места и времени.

5.25.2. ПРОЕКТИРОВАНИЕ ГОРОДСКОЙ ТРАНСПОРТНОЙ СЕТИ

Многопродуктовые сетевые модели используются также при проектировании городских транспортных сетей. В этих моделях узлы соответствуют участкам или районам города, а дуги — улицам или дорогам других видов. Требования к транспортной сети задаются матрицей поездок **D**, элементы d_{ij} которой равны числу транспортных средств, движущихся из участка i к участку j в течение фиксированного интервала времени. Каждая дуга имеет заданную пропускную способность u_{ij}, которая может быть увеличена (например, в результате улучшения дорог). Плата за увеличение пропускной способности дуги (i, j) на единицу равна c_{ij}. Общая сумма денежных средств, предназначен-

ных на улучшение дорог, равна B. «Продуктами» в данной модели являются потоки из каждого источника во все пункты назначения. Пусть $x^k{}_{ij}$ — число транспортных средств, движущихся по дуге (i, j) и начавших свой путь в узле k; y_{ij} — увеличение пропускной способности дуги (i, j). Предположим, что время проезда по дуге (i, j) является некоторой функцией потока:

$$f_{ij}\left(\sum_k x^k{}_{ij}\right).$$

Задача проектирования сети заключается в определении дуг, пропускную способность которых следует увеличить, и вычислении потоков по каждой дуге, минимизирующих общее время поездки. Математически данная задача формулируется следующим образом:

$$\text{минимизировать} \sum_{(i,\,j)} f_{ij}\left(\sum_k x^k{}_{ij}\right) \tag{5.80}$$

при условии, что

$$\sum_j x^k{}_{ji} - \sum_j x^k{}_{ij} = d_{ki}, \tag{5.81}$$

$$\sum_k x^k{}_{ij} \leqslant u_{ij} + y_{ij}, \tag{5.82}$$

$$\sum_{(i,\,j)} c_{ij} y_{ij} \leqslant B, \tag{5.83}$$

$$x^k{}_{ij},\ y_{ij} \geqslant 0. \tag{5.84}$$

В задаче (5.80)—(5.84) было сделано одно важное допущение, касающееся поведения водителя. Это допущение основано на классическом принципе распределения транспортных средств, сформулированном Уодропом [39]: водители выбирают маршруты таким образом, что суммарное время поездок для всей системы является минимальным. Как правило, целевая функция (5.80) является нелинейной и выпуклой.

5.25.3. МОДЕЛИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Очень похожи на только что рассмотренную нами модель городской транспортной сети многие модели вычислительных систем. В этих моделях дуги соответствуют каналам, а узлы — терминалам, системным и запоминающим устройствам и т. п. Каждая дуга имеет фиксированную пропускную способность u_{ij}, которая может быть увеличена на y_{ij} единиц вплоть до мак-

симального значения b_{ij}. Стоимость увеличения пропускной способности на единицу равна c_{ij}. Роль продуктов вновь играют потоки (информация) из источника во все пункты назначения, а вместо матрицы поездок D вводится матрица, элементы d_{ij} которой соответствуют объему информации, передаваемой из узла i в узел j. Стоимость передачи единицы информации по дуге (i, j) равна e_{ij}. Математически эта задача может быть сформулирована следующим образом:

$$\text{минимизировать} \sum_k \sum_{(i,\,j)} e_{ij} x^k{}_{ij} + \sum_{(i,\,j)} c_{ij} y_{ij} \tag{5.85}$$

при условии, что

$$\sum_j x^k{}_{ji} - \sum_j x^k{}_{ij} = d_{ij}, \tag{5.86}$$

$$\sum_k x^k{}_{ij} \leqslant u_{ij} + y_{ij}, \tag{5.87}$$

$$y_{ij} \leqslant b_{ij}, \tag{5.88}$$

$$x^k{}_{ij},\ y_{ij} \geqslant 0. \tag{5.89}$$

Описанная модель может быть использована для определения пропускной способности каналов, при которой заданные требования удовлетворяются с минимальными затратами. В этом случае значения e_{ij} обычно полагаются равными 0, $u_{ij}=0$, а величины b_{ij} выбираются достаточно большими. Другим примером использования этой модели является задача минимизации стоимости передачи информации при фиксированных пропускных способностях дуг. В этом случае $b_{ij}=0$.

5.26. ЗАМЕЧАНИЯ

Обзор задач о многопродуктовом потоке, описание различных методов их решения, а также исчерпывающий список литературы по данному вопросу содержатся в работах Кеннингтона [39] и Асада [40]. Вопросы использования теории унимодулярности при решении задач о многопродуктовом потоке и преобразования последних к задачам об однопродуктовом потоке впервые были рассмотрены в работе Эванса, Джэрвиса и Дюка [41]. Более полное изложение этих вопросов содержится в работах Эванса и Джэрвиса [42] и Эванса [43—45]. Результаты, касающиеся агрегирования, были получены Джиоффрионом [46] и Зипкином [47] и обобщены в работах Эванса [48, 49]. Вполне плоские сети рассмотрены в работе Сакаровича [50]. Ху [51] был разработан алгоритм поиска максимального двухпродуктового потока в неориентированных сетях. Вопросы, касающиеся приложения потоков в сетях с воронкообразным узлом, рассмотрены в работе Джэрвиса и Миллера [52]. Задача составления расписания движения судов решена в работе Беллмора, Беннингтона и Лубра [53]. Другие прикладные задачи, рассмотренные в настоящем разделе, содержатся в обзорной работе Кеннингтона. Литературу по вопросам проектирования городской транспортной сети и вычислительных систем можно найти в работе [39].

УПРАЖНЕНИЯ

1. Классическим примером задачи о многопродуктовом потоке является задача проектирования транспортной сети с несколькими источниками. Заданы сеть $G=(\mathbf{N}, \mathbf{A})$ и величины предложения или спроса (обозначаемые через Q_{ik}, $k=1, 2, \ldots, L$) для каждого из L продуктов в узле i. Требуется для каждого звена (i, j) и каждого продукта k определить такой поток x_{ijk}, что сумма затрат на строительство сети и транспортных затрат является минимальной. Предполагается, что стоимость строительства звена (i, j) равна p_{ij}, а стоимость прохождения единицы потока по дуге (i, j) равна c_{ij}.

2. Для каждой из следующих общих задач дать физическую интерпретацию понятий узлов, дуг и потоков в терминах многопродуктовых сетей:

а. Перевозка груза между городами, осуществляемая автомобильным, железнодорожным, воздушным и водным транспортом.

б. Проектирование коммуникационной сети с несколькими центрами связи и несколькими линиями передачи информации.

в. Распределение почты, включая письма и посылки, между несколькими почтовыми отделениями.

г. Управление производственным процессом и запасами при непрерывном предложении и спросе.

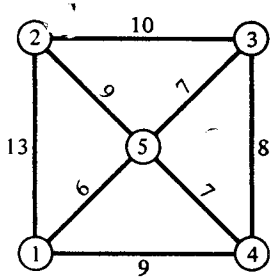
д. Проектирование системы сбора, переработки и ликвидации твердых отходов.

е. Проектирование районной сети дорог для пассажирских и грузовых транспортных средств.

ж. Проектирование городской транспортной сети.

з. Планирование перевозки различных видов продукции по сети с несколькими станциями и несколькими путями транспортировки.

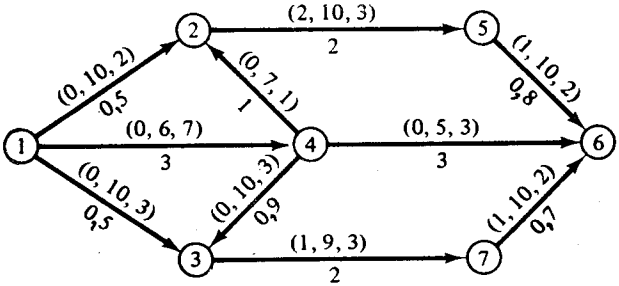
3. По изображенной ниже сети может протекать двухпродуктовый поток. Предполагается, что поток каждого из этих продуктов может протекать по каждой дуге в произвольном направлении. В узлах 1, 2, 3 находится неограниченное количество продукта 1, стоком которого является узел 4, а в узле 5 — неограниченное количество продукта 2, стоком которого является узел 1. Число, приписанное каждой дуге, равно максимально допустимому суммарному потоку обоих продуктов по ней. Доход при прохождении единицы продукта 1 из его источника в сток равен 4, а соответствующая величина для продукта 2 равна 6. Задача заключается в нахождении допустимого двухпродуктового потока, максимизирующего общий доход.



4. В стандартной транспортной задаче при удовлетворении спроса не делается различий между пунктами снабжения. Однако большинство фирм, производящих многономенклатурную продукцию, нуждается в составлении полной системы распределения, в которой учитывается каждый вид изделия в отдельности. Показать, как может быть использована однопродуктовая транспортная модель для решения простой задачи о многопродуктовом пото-

ке, в которой в каждый пункт потребления требуется доставлять различные виды продукции. Какие предположения делаются в этой задаче?

5. Найти поток минимальной стоимости в изображенной ниже сети с выигрышами и проигрышами. Каждой дуге (i, j) приписаны четыре параметра, L_{ij}, U_{ij}, c_{ij} и A_{ij}, обозначающих нижнюю границу потока, верхнюю границу потока, стоимость прохождения единицы потока и коэффициент выигрыша или проигрыша соответственно. Указание: каждая дуга, для которой $L_{ij}>0$, эквивалентна двум дугам с параметрами $(0, U_{ij}-L_{ij}, c_{ij})$ и $(0, L_{ij}, -\infty)$.



6. Финансовый директор фирмы разрабатывает план осуществления денежных вкладов. Общая сумма вкладываемых денег составляет 1 млн. долл. Имеется шесть типов ценных бумаг, доход по которым составляет 3,5, 2,5, 3,0, 4,5, 5,0 и 4,0% соответственно. По существующему соглашению не менее 35% общей суммы должны составлять вложения в 1 и 2 типы ценных бумаг и не более 40% — в 3 и 4 или 5 и 6 типы. Определить, каким образом следует вкладывать деньги.

7. Процесс производства некоторого изделия состоит из трех этапов. Операции первого этапа могут выполняться на любом из трех станков, A, B и C, операции второго этапа — на любом из двух станков, D и E, и операции третьего этапа — на любом из трех станков, F, G и H. Все детали, обрабатываемые на станке A, затем должны быть обработаны на станке D. Все детали, обрабатываемые на станке B, затем обрабатываются на станке D или E. Все детали, обрабатываемые на станке C, затем должны быть обработаны на станке E. Детали, обрабатываемые на D, затем обрабатываются на F или G. Детали, обрабатываемые на E, затем обрабатываются на G или H. Еженедельно на каждом из станков A, B и C в отдельности может быть обработано не более 50% всех деталей, на каждом из станков D и E — не более 70% и на каждом из станков F и G — не более 50%. 95% всех деталей, обработанных на станке A, являются доброкачественными и отправляются на дальнейшую обработку. Соответствующие величины для станков B, C, D, E, F, G и H равны 98, 93, 95, 90, 91, 89 и 94% соответственно. Стоимости обработки одной детали на станках A, B, C, D, E, F, G и H составляют 10, 14, 8, 18, 14, 10, 9 и 11 соответственно. Найти оптимальный план производства 100 деталей в неделю.

8. Продукция может производиться в каждом из $T=4$ периодов времени. Пусть D_t — спрос в период t, c_t — стоимость производства единицы продукции в период t, h_t — издержки хранения в течение $(t+1)$-го периода единицы продукции, произведенной в период t, и I_t — объем запасов продукции в конце периода t. Часть I. а) Сформулировать задачу линейного программирования, решение которой дает план производства продукции на T периодов с минимальными общими затратами. б) Начертить сеть для данной задачи. Часть II. Модифицировать задачу, сформулированную в части I, таким образом, что если в некотором периоде спрос на продукцию не удовлетворен, то недостающий ее объем может быть произведен в следующем

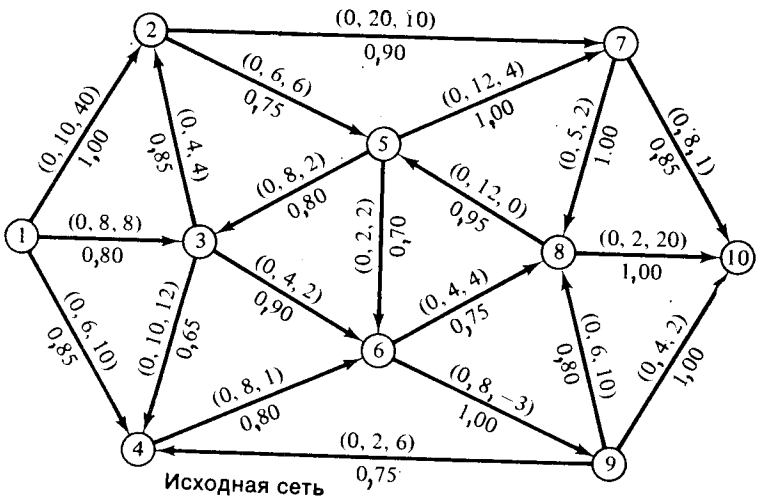
периоде. Часть III. Предположим, что если продукция не произведена в срок, то в 20% случаев заказчик отменяет свой заказ. Принимая во внимание данное условие, модифицировать сеть, построенную в части II. Решить сетевые задачи, сформулированные в частях I, II и III, используя приводимые ниже данные. Сравнить результаты частей II и III.

Период	Спрос	Стоимость производства единицы продукции (долл.)	Издержки хранения единицы продукции (долл.)
1	100	24	1
2	110	26	1
3	95	21	1
4	125	24	1

9. Рассмотрим изображенную ниже сеть, в которой дуга (i, j) помечена как

$$(i) \xrightarrow[A_{ij}]{(L_{ij}, U_{ij}, c_{ij})} (j).$$

Здесь L_{ij}, U_{ij}, c_{ij} и A_{ij} — нижняя граница потока, верхняя граница потока, стоимость прохождения единицы потока и множитель выигрыша или проигрыша соответственно. Найти наилучшее решение, при котором в узел 10 поступает 5 единиц потока.



Исходная сеть

10. Для осуществления капиталовложения в начале 1 года имеется сумма в 100 тыс. долл., а в начале 2 года — 200 тыс. долл. Вклады можно производить в начале 1, 2 и 3 года. Предположим также, что имеющиеся деньги можно помещать в банк на срочный счет, получая при этом 5% годовых. В каждом периоде вклад каждого вида не должен быть меньше 10 000 долл. и превосходить 50 000 долл. Прибыль на инвестированный капитал состав-

ляет 18, 14 и 10% на 1, 2 и 3 года соответственно. Как следует осуществлять капиталовложения, чтобы максимизировать общий доход?

11. *Обобщенная задача о назначениях* формулируется следующим образом. M рабочих назначаются на N работ. Каждый рабочий может выполнить любую работу целиком или частично. В последнем случае работа должна быть разделена между несколькими (двумя или более) рабочими. Предположим, что за один час рабочий i выполняет K_{ij}-ю часть работы j и что рабочий i не может работать более U_i часов. Пусть рабочий i занимается работой j X_{ij} часов, а C_{ij} — стоимость рабочего места в час при выполнении рабочим i работы j. Задача заключается в следующем:

$$\text{минимизировать} \sum_i \sum_j C_{ij} X_{ij}$$

при условии, что

$$\sum_j X_{ij} \leqslant U_i, \quad i = 1,2,\ldots,M,$$

$$\sum_i K_{ij} X_{ij} = 1, \quad j = 1,2,\ldots,N,$$

$$X_{ij} \text{ для всех } i, j.$$

Сформулированная задача эквивалентна задаче нахождения потока минимальной стоимости в изображенной ниже сети. Найти оптимальное назначение в случае, когда число рабочих равно 3, а число работ — 2. Исходные данные приводятся ниже.

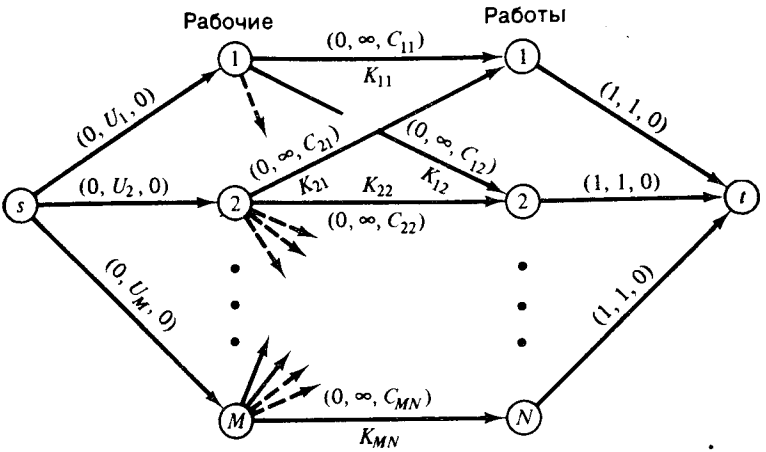
12. Решить следующую обобщенную сетевую задачу, используя алгоритм для сетей с выигрышами и проигрышами, описанный в разд. 5.5.

а) Найти решение (и его стоимость), при котором в узел 4 поступает 10 единиц потока.

б) Свести данную задачу к обычной потоковой задаче, используя метод, описанный в разд. 5.4. Сравнить результаты, полученные в п. а) и б).

13. Стрела попадает в цель с вероятностью 0,20. Построить ГЕРТ-сеть для определения ожидаемого числа выстрелов, при котором цель будет поражена дважды.

14. Дональд Ду-Райт открыл мастерскую по ремонту телевизоров в Колледж-Стейшен. Будучи выпускником Техасского университета и инженером по образованию, он решает проанализировать свою деятельность, которая заключается в следующем. В мастерскую поступают три основных типа телевизоров: 1) черно-белые портативные; 2) черно-белые обычные; 3) цветные. Согласно последним наблюдениям, доля телевизоров каждого типа от их общего числа составляет 42, 31 и 27% соответственно. Продолжительность ремонта одного портативного черно-белого телевизора является экспоненциально распределенной случайной величиной с математическим ожиданием, равным 3 ч, а обычного черно-белого телевизора — экспоненциально распределенной случайной величиной с математическим ожиданием, равным 5 ч. Дональд не имеет достаточного опыта, чтобы ремонтировать цветные телевизоры, и поэтому он вынужден отправлять их в соседнюю мастерскую. Однако он обнаружил, что приблизительно каждый третий цветной телевизор имеет незначительную неисправность и может быть отремонтирован им самим, причем продолжительность ремонта составляет 40 мин в 30% случаев, 55 мин в 30% случаев и 75 мин в 40% случаев. Продолжительность ремонта цветного телевизора в соседней мастерской является случайной величиной, имеющей гамма-распределение, математическое ожидание 7 ч и дисперсию 4 ч. Мистер Ду-Райт проводит временную проверку всех отремонтированных телевизоров. Выясняется, что 10% всех телевизоров, отремонтированных и в соседней мастерской, и 5% телевизоров, отремонтированных



Рабочие	Работы	Стоимость	Коэффициент выигрыша
1	1	4	0.10
1	2	7	0.15
2	1	5	0.20
2	2	3	0.05
3	1	4	0.10
3	2	6	0.08

Рабочие	Количество часов
1	12
2	20
3	25

Дуга	Нижняя граница	Верхняя граница	Стоимость	Дуговой множитель
(1, 2)	0	3	1	4,0
(1, 3)	0	4	5	0,50
(2, 3)	0	3	4	1,00
(3, 2)	0	6	3	0,333
(2, 4)	0	5	1	0,25
(3, 4)	0	7	9	3,00

им самим, нуждаются в дополнительном ремонте. Продолжительность дополнительного ремонта приблизительно вдвое больше продолжительности основного ремонта. После дополнительного ремонта телевизоры вновь проходят проверку. Провести анализ работы мастерской.

15. Для короны короля Ричрокса нужен идеально отшлифованный алмаз. Если ювелир отшлифовал алмаз не достаточно хорошо, то последний поступает на изготовление режущих инструментов. Предположим, что время шлифовки алмаза подчиняется закону распределения Пуассона с математическим ожиданием, равным 18 ч. Чему равно ожидаемое время шлифовки алмаза и передачи его королю, если вероятность изготовления требуемого бриллианта составляет 0,15, а время доставки равно 1 ч?

16. Предположим, что Ричрокс, как и все настоящие короли, участвует в сражении, и ювелиру приказывают доставить бриллиант лично королю. Пусть время поиска короля подчиняется экспоненциальному закону распределения с математическим ожиданием, равным 9 ч. Чему равно ожидаемое время доставки в данном случае?

17. Heavy Steel Company (Блэк-Смоук, шт. Техас) производит сталь в слитках, предназначенную для дальнейшей обработки. Нерафинированная сталь поступает из трех различных источников, A, B и C, на долю каждого из которых приходится соответственно 50, 25 и 25% общего ее объема. Сначала нерафинированная сталь плавится в вагранке, причем время плавки подчиняется нормальному закону распределения с $\mu=25$ ч и $\sigma^2=9$ ч^2. Затем раскаленные слитки стали охлаждаются. Время выполнения процедуры охлаждения равно 2,5 дням для 30% слитков, 1,8 дням для 15% и 2,7 дням для 55%. После охлаждения слитки подвергаются термической обработке. Время выполнения данной процедуры подчиняется экспоненциальному закону распределения с $\mu=46$ ч. После термической обработки слитки вынимают и проверяют. 85% всех слитков проходят контроль и поступают в цех резки. 15% слитков подвергаются повторной термической обработке, время выполнения которой подчиняется экспоненциальному закону распределения с математическим ожиданием, равным 14,6 ч. После повторной термической обработки слитки вновь проверяются. 95% их отправляют в цех резки, а 5% поступает в продажу как второсортная сталь. Каждая из процедур проверки выполняется специальным прибором в течение 5 мин. После процедуры резки, время выполнения которой подчиняется экспоненциальному закону распределения с $\mu=6{,}2$ мин, разрезанные слитки проверяются. Форма 85% слитков соответствует установленному стандарту, 10% вновь поступает в цех резки, а 5% сдается в металлолом.

а. Начертить ГЕРТ-сеть для данного примера.

б. Определить W-функцию для каждой дуги.

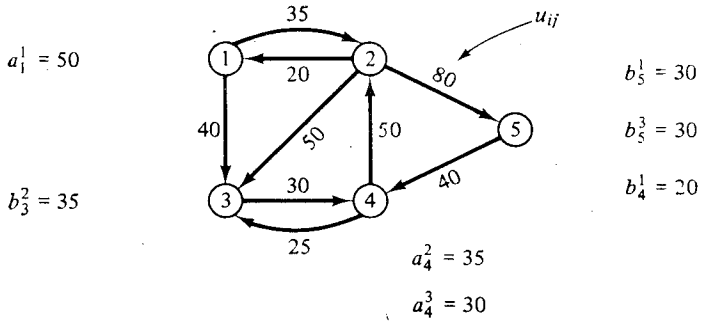
в. Какова доля продукции, которая производится из нерафинированной стали, поступающей из источников A и B, и признается недоброкачественной?

г. Чему равно математическое ожидание и дисперсия времени производства доброкачественной продукции?

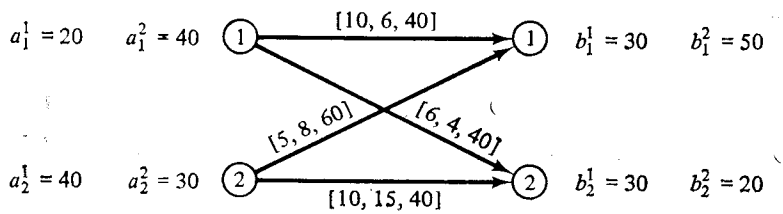
18. Показать, что многопродуктовая транспортная задача (5.23) — (5.27) и транспортная задача (5.40) — (5.46) эквивалентны, т. е. любое допустимое решение одной из них является допустимым для другой.

19. Найти дезагрегированное решение задачи о распределении фруктов, исходные данные для которой приведены в табл. 5.11. Показать, что его стоимость равна стоимости агрегированного решения.

20. Описанную ниже многопродуктовую задачу о перевозках сформулировать в виде задачи линейного программирования. Определить тип этой задачи линейного программирования.

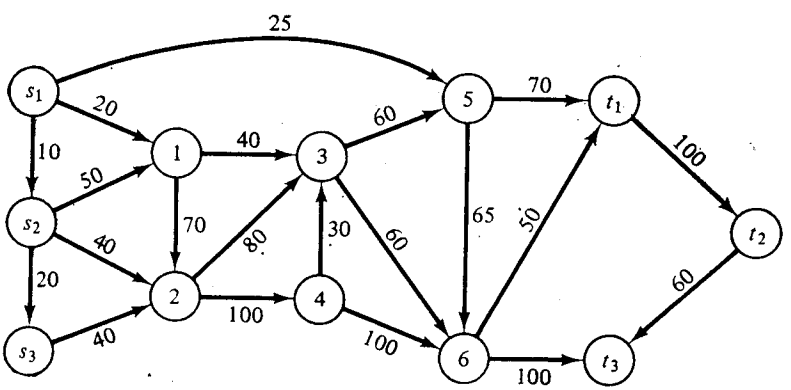


21. Свести описанную ниже задачу к эквивалентной задаче об однопродуктивном потоке и решить ее с помощью алгоритма дефекта. Рассмотреть два случая: когда заданы границы потоков по дугам и когда на потоки по дугам ограничения не накладываются.

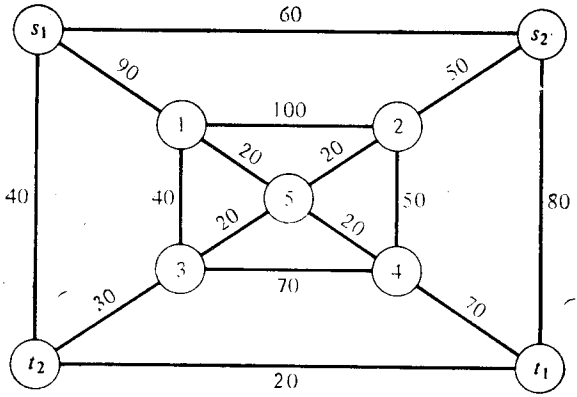


22. Решить задачу о распределении фруктов, объединяя города а) Санта-Барбара и Бейкерсфилд; б) Санта-Барбара и Сакраменто. В каком случае получено лучшее решение? Объяснить, почему.

23. Решить описанную ниже задачу о многопродуктовом максимальном потоке.



24. Найти максимальный двухпродуктовый поток.



25. Предположим, что в сети из упр. 24 s_1 и t_1 соответственно источник и сток однопродуктового потока, а узел 5 воронкообразный. Найти максимальный поток через воронкообразный узел.

ЛИТЕРАТУРА

1. Bhaumik G., Optimum Operating Policies of a Water Distribution System with Losses, Unpublished Ph. D. dissertation, The University of Texas at Austin, 1973.
2. Charnes A., Kirby N., Raike W., Chance-constrained Generalized Networks, *Operations Research,* **14**, 1113—1120 (1966).
3. Charnes A., Raike W. M., One-Pass Algorithm for Some Generalized Network Problems, *Operations Research,* **14**, 914—924 (1966).
4. Charnes A., Cooper W., Management Models and Industrial Applications of Linear Programming, Vols. 1, 2, Wiley, Inc., New York, 1961.
5. Crum R., Cash Management in the Multinational Firm: A Constrained Generalized Network Approach, Working paper, The University of Florida, Gainesville, Fla., 1977.
6. Dantzig G., Linear Programming and Extensions, Princeton University Press, Princeton, N. J., 1963. [Имеется перевод: Данциг Дж. Б. Линейное программирование и его применения. — М.: Прогресс, 1966.]
7. Glover F., Klingman D., On the Equivalence of Some Generalized Network Problems to Pure Network Problems, Research Report CS81, Center for Cybernetic Studies, The University of Texas, Austin, January 1972; *Mathematical Programming,* **4**, 269—278 (1973).
8. Glover F., Hultz J., McMillan C., The Netform Concept, *Proceeding of the ACM* (1977).
9. Glover F., Klingman D., Network Applications in Government and Industry, *AIIE Transactions,* **9** (4) (December 1977).
10. Glover F., Klingman D., Napier A., Basic Dual Feasible Solutions for a Class of Generalized Networks, *Operations Reseach,* **20** (1) (1972).
11. Glover F., Klingman D., Stutz J., Extensions of the Augmented Predecessor Index Method (APS) to Generalized Network Problems, *Transportation Science,* **7** (4), 377—384 (1973).
12. Glover F., Klingman D., Stutz J. Implementation and Computational Study

of a Generalized Network Code, Paper psesented at the 44th National ORSA Conference, San Diego, Calif., 1973.
13. Glover F., Hultz J., Klingman D., Improved Computer Based Planning Networks, *Interfaces,* **8** (4) (August 1978).
14. Grinold R. C., Calculating Maximal Flows in a Network with Positive Gains, *Operations Research,* **21**, 528—541 (1973).
15. Jarvis J. J., Jezior A. M., Maximal Flow with Gains through a Special Network, *Operations Research,* **20**, 678—688 (1972).
16. Jensen P. A., Bhaumik G., A Computationally Efficient Algorithm for the Network with Gains Problem, Paper presented at the 45th National Meeting of ORSA, Boston, April 1974.
17. Jensen P. A., Barnes W., Network Flow Programming, Wiley, Inc., New York, 1980.
18. Jewell W. J., Optimal Flows through Networks with Gains, *Operations Research,* **10**, 476—499 (1962).
19. Kim Y., An Optimal Computational Approach to the Analysis of a Generalized Network of Copper Refining Process, Joint ORSA/TIMS/AIIE Conference, Atlantic City, N. J., 1973.
20. Lawler E. L., Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and iWnston, New York, 1976, pp. 134—138.
21. Maurras J. F., Optimization of the Flow through Networks with Gains, *Mathematical Programming,* **3**, 135—144 (1972).
22. Minieka E. T., Optimization Algorithms for Networks and Graphs, Marcel Dekker, Inc., New York, 1978, pp. 151—174. [Имеется перевод: Майника Э. Алгоритмы оптимизации на сетях и графах. — М.: Мир, 1981.]
23. Minieka E. T., Optimal Flow in a Network with Gains, *INFOR,* **10**, 171—178 **(1972)**.
24. Oettli W., Prager W., Flow Networks with Amplification and Coupling, Unternehmensforschung, **10**, 42—49 (1966).
25. Truemper K., On Max Flows with Gains and Pure Min. Cost Flows, *SIAM Journal of Applied Mathematics,* **32**, 450—456 (1973).
26. Truemper K., An Efficient Scaling Procedure for Gains Networks, *Networks,* **6**, 151—160 (1976).
27. Truemper K., Optimal Flows in Nonlinear Gains, *Networks,* **8**, 17—36 (1978).
28. Wagner H., Principles of Operations Research, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1979.
29. Dudley N. A., Work Time Distribution, *International Journal of Production Research,* **1** (2), (1963).
30. Karger B. W., Bayta F. H., Engineering Work Measurement, The Industrial Press, New York, 1966.
31. Phillips D. T., Smith D. R., Determination of Probabilistic Time Standards for Tasks Performed Under Uncertainty, Proceedings on the AIEE National Conference, San Francisco, Ca., 1979.
32. Phillips D. T., Ravindran A., Solberg J. J., Operations Research; Principles and Practice, Wiley, Inc., New York, 1976.
33. Pritsker and Associates, West Lafayette, Ind. — private consulting firm.
34. Pritsker A. A. B., Happ W. W., GERT: Graphical Evaluation and Review Technique, Part I, Fundamentals, *The Journal of Industrial Engineering* (May 1966).
35. Pritsker A. A. B., Whitehouse G. E., GERT: Graphical Evaluation and Review Technique, Part II, *Journal of Industrial Engineering* (June 1966).
36. Pritsker A. A. B., *The Production Engineer,* pp. 499—506 (October 1968).
37. Whitehouse G. E., System Analysis and Design Using Network Techniques, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1973.
38. Whitehouse G. E., Pritsker A. A. B., GERT — Generating Functions, Condi-

tional Distributions, Counters, Renewal Times, and Correlations, *AIIE Transactions* (March 1969).
39. Kennington J. L., A Survey of Linear Cost Multicommodity Network Flows, *Operations Research,* **26**, 206—236 (1978).
40. Assad A. A., Multicommodity Network Flows — A Survey, *Networks,* 8, 37—91 (1978).
41. Evans J. R., Jarvis J. J., Duke R. A., Graphic Matroids and the Multicommodity Transportation Problem, *Mathematical Programming,* **13**, 323—328 (1977).
42. Evans J. R., Jarvis J. J., Network Topology and Integral Multicommodity Flow Problems, *Networks,* 8, 107—119 (1978).
43. Evans J. R., A Combinational Equivalence between a Class of Multicommodity Flow Problems and the Capacitated Transportation Problem, *Mathematical Programming,* **10**, 401—404 (1976).
44. Evans J. R., A Single Commodity Transformation for Certain Multicommodity Networks, *Operations Research,* **26**, 673—680 (1978).
45. Evans J. R., On Equivalent Formulations of Certain Multicommodity Networks as Single Commodity Flow Problems, *Mathematical Programming,* **15**, 92—99 (1978).
46. Geoffrion A., Customer Aggregation in Distribution Modeling, Working Paper No. 259, Western Management Science Institute, University of California, Los Angeles, 1976.
47. Zipkin P. H., Aggregation in Linear Programming, Ph. D. dissertation, Yale University, December 1977.
48. Evans J. R., Solving Multicommodity Transportation Problems through Aggregation, ORSA/TIMS National Conference, Los Angeles, November 1978.
49. Evans J. R., Model Simplification in Multicommodity Distribution Systems through Aggregation, Proceedings, American Institute of Decision Sciences, National Meeting, New Orleans, November 1979.
50. Sakarovitch M., The Multicommodity Maximal Flow Problems, ORC 66-25, University of California, Berkeley, 1966.
51. Hu T. C., Multicommodity Network Flows, *Operations Research,* **11**, 344—360 (1963).
52. Jarvis J. J., Miller D. D., Maximal Funnel-Node Flow in an Undirected Network, *Operations Research,* **21**, 365—369 (1973).
53. Bellmore M., Bennington G., Lubore S., A Multivehicle Tanker Scheduling Problem, *Transportation Sciences,* **5**, 36—47 (1971).

Приложение

ОПИСАНИЕ ПРОГРАММЫ СЕТЕВОЙ ОПТИМИЗАЦИИ

ПАКЕТ СЕТЕВОЙ ОПТИМИЗАЦИИ

Назначение: нахождение оптимальных решений различных задач на сетях с детерминированной структурой. Могут быть решены следующие сетевые задачи: 1) задача о кратчайшей цепи; 2) задача о многополюсной кратчайшей цепи; 3) задача о многополюсной цепи с максимальной пропускной способностью; 4) задача о кратчайшем остовном дереве; 5) задача коммивояжера; 6) задача о максимальном потоке; 7) транспортная задача; 8) задача о назначениях; 9) задача о дереве кратчайших цепей; 10) задача о перевозках; 11) задача о максимальном потоке минимальной стоимости; 12) задача о K кратчайших путях; 13) минимизация на обобщенной сети.

Ограничения: для задач 1—6 и 12 программа позволяет обрабатывать сети, содержащие до 50 узлов и 50 дуг. В задачах 7—11 и 13 сети могут содержать не более 500 узлов и 500 дуг. Размеры сетей можно увеличить, изменив соответствующим образом границы массивов в операторах размерности, записанных в основной программе и подпрограммах.

Замечание: если в вычислительную машину можно ввести несколько программ, то все они могут быть выполнены за один прогон. Для этого в конце каждой задачи следует поместить карту 'PEND' (набор 4), а в качестве последней карты набора данных использовать карту 'EXIT' (набор 5). Ниже приводится список алгоритмов с их именами вызова и именами подпрограмм:

Имя вызова	Имя подпрограммы	Алгоритм
DIJK	DIJKA	Алгоритм Дейкстры решения задачи о кратчайшей цепи
MTSC	MULTSH	Алгоритм решения задачи о многополюсной кратчайшей цепи
KSRT	KSHORT	Алгоритм нахождения K кратчайших путей между заданным источником и заданным стоком
MSTR	MINSPA	Алгоритм решения задачи о кратчайшем остовном дереве
MAXF	MAXFLO	Алгоритм решения задачи о максимальном потоке
TSPR	TRASAL	Алгоритм ветвей и границ решения задачи коммивояжера
OKAL	OKALG	Алгоритм дефекта
MCRP	MAXCAP	Алгоритм нахождения цепи с максимальной пропускной способностью между всеми парами узлов
GNET	GENPAK	Алгоритм Йенсена и Бомика решения сетевых задач с выигрышами и проигрышами

Входные данные:

Набор 1. Одна карта с именем вызова в формате (A4).

Набор 2. Данный набор состоит из одной карты, которая указывает число узлов и число дуг в сети в формате (2I10).

Набор 3. Число карт в данном наборе равно числу дуг в сети. Для задач 1—5 с каждой карты считываются следующие величины: 1) номер начального узла дуги; 2) номер конечного узла дуги; 3) значение, приписанное дуге.

Форма (4X, I6, I10, F10.2).

Для задачи 6 с каждой карты этого набора считываются следующие величины: 1) номер начального узла дуги; 2) номер конечного узла дуги; 3) пропускная способность дуги; 4) начальный поток по дуге (должно быть

выполнено условие сохранения потока). Если поставлен знак пробела, то все начальные потоки полагаются равными нулю.
Формат (4X, I6, I10, 2F10.2).

Для задач 7—11, при решении которых используется алгоритм дефекта, с каждой карты считываются следующие величины: 1) номер начального узла дуги; 2) номер конечного узла дуги; 3) верхняя граница потока по дуге; 4) нижняя граница потока по дуге; 5) стоимость прохождения единицы потока.
Формат (2I5, 3F10.2).

В задаче 12 данный набор состоит из трех разделов.

Раздел 1:

1) начальный узел дуги;
2) конечный узел дуги;
3) длина дуги.
Формат (3I10).

Для каждой дуги сети набивается одна карта. Карты в этом разделе располагаются в порядке возрастания номеров конечных узлов.

Раздел 2:

1) K, NS, IMAX.
Формат (4X, 3I5).

K — заданное число различных длин путей,

IMAX — максимально допустимое число итераций в алгоритме двойного поиска,

NS — начальный узел всех K искомых путей.

Раздел 3:

1) NF, PMAX.
Формат (4X, 2I5).

NF — конечный узел всех K искомых путей.

PMAX — максимальное число искомых путей между узлами NS и NF.

Замечание: если требуется последовательно решать задачу 12 для различных источников и(или) стоков, то надо заменять только разделы 2 и 3 набора 3.

В задаче 13 данный набор состоит из двух разделов.

Раздел 1:

1) начальный узел дуги;
2) конечный узел дуги;
3) нижняя граница потока по дуге;
4) верхняя граница потока по дуге;
5) стоимость прохождения единицы потока;
6) коэффициент выигрыша дуги.
Формат (2I10, 4F10.2).
Для каждой дуги сети набивается одна карта.

Раздел 2:

1) источник, сток, IPRINT, OUTFLOW.
Формат (3I5, F10.2).

IPRINT — опция PRINT (0 для узкой печати, 1 для широкой печати, 2 для сверхширокой печати),

OUTFLOW — требуемый выходной поток.

Набор 4. Для всех задач данный набор состоит из одной карты, содержащей слово 'PEND' в формате (A4), которая указывает конец задачи.
Набор 5. Данный набор состоит из одной карты, содержащей слово 'EXIT' в формате (A4), которая указывает конец программы сетевой оптимизации (конец всех входных данных).

Содержание пакета. Данный пакет содержит 9 подпрограмм, реализующих следующие алгоритмы:

1. Алгоритм Дейкстры решения задачи о кратчайшей цепи из источника в любой другой узел сети.

2. Алгоритм решения задачи о многополюсной кратчайшей цепи. Позволяет находить кратчайшие цепи между всеми парами узлов сети. Основан на применении трехместной операции.

3. Алгоритм построения кратчайшего остовного дерева сети.

4. Алгоритм нахождения максимального потока в сети с ограниченной пропускной способностью, основанной на применении процедуры расстановки пометок.

5. Алгоритм ветвей и границ решения задачи коммивояжера.

6. Алгоритм дефекта нахождения оптимальных решений транспортной задачи, задачи о перевозках, задачи о назначениях и т. п.

7. Алгоритм нахождения K кратчайших путей между любыми двумя узлами сети. Циклы допустимы.

8. Алгоритм нахождения цепи с максимальной пропускной способностью между всеми парами узлов сети (предложен Ху Т.), основанный на применении модифицированной трехместной операции. В сети, каждая дуга которой имеет ограниченную пропускную способность (заданы верхние границы), цепью с максимальной пропускной способностью из узла A в узел B называется цепь, у которой минимальная величина пропускной способности дуги является максимальной для всех цепей из A в B.

9. Алгоритм Йенсена и Бомика нахождения потока минимальной стоимости в обобщенных задах (в задачах на сетях с выигрышами и проигрышами). Генерирующие циклы допускаются. Алгоритм является сходящимся за конечное число шагов.

Литература: Dijkstra E. W., A Note on Two Problems in Connection with Graphs, *Num. Math.*, 1, 269—271 (1959); Hu T. C. Integer Programming and Network Flows, Addison-Wesley, 1969.

```
            MAIN PROGRAM
     DIMENSION NODE(50,50),A(50,50),LAB(50),LP(50),EPS(50),PL(50),
    1NICID(50,50),SCAN(50),TL(50)
     COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
    1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
     EQUIVALENCE (NODE,A,S),(LAB,LP,ROWUSD),(EPS,PL,COLUSD),(NICID,UP)
    1,(SCAN,TL,NP)
     INTEGER DIJK/'DIJK'/,MTSC/'MTSC'/,MSTR/'MSTR'/,MAXF/'MAXF'/,TSPR/
    1'TSPR'/,OKAL/'OKAL'/,EXIT/'EXIT'/,IPRO
     INTEGER PEND/'PEND'/,NPRO
     INTEGER KSRT/'KSRT'/,MCRP/'MCRP'/,GNET/'GNET'/
  60 READ(5,5)IPRO

   5 FORMAT (A4)
     IF(IPRO.EQ.EXIT) GO TO 90
     IF(IPRO.EQ.DIJK) GO TO 10
     IF(IPRO.EQ.MTSC) GO TO 15
     IF(IPRO.EQ.MSTR) GO TO 20
     IF(IPRO.EQ.MAXF) GO TO 25
     IF(IPRO.EQ.TSPR) GO TO 30
     IF(IPRO.EQ.OKAL) GO TO 40
     IF(IPRO.EQ.KSRT) GO TO 45
     IF(IPRO.EQ.MCRP) GO TO 50
     IF(IPRO.EQ.GNET) GO TO 55
     WRITE(6,2)

   2 FORMAT('UNIDENTIFIED ALGORITHM*****EXECUTION TERMINATED')
     GO TO 90
  10 CALL DIJKA
     GO TO 60
  15 CALL MULTSH
     GO TO 60
  20 CALL MINSPA
     GO TO 60
  25 CALL MAXFLO
     GO TO 60
  30 CALL TRASAL
     GO TO 60
```

```
   40 CALL OKAIG
      READ(5,5) NPRO
      GO TO 60
   45 CALL KSHORT
      GO TO 60
   50 CALL MAXCAP
      GO TO 60
   55 CALL GENPAK
      READ(5,5) NPRO
      GO TO 60
   90 CONTINUE
      STOP

      END
C
      SUBROUTINE DIJKA
      DIMENSION NODE(50,50),A(50,50),LAB(50),LP(50),EPS(50),PL(50),
     1NICID(50,50),SCAN(50),TL(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      EQUIVALENCE (NODE,A,S),(LAB,LP,ROWUSD),(EPS,PL,COLUSD),(NICID,UP)
     1,(SCAN,TL,NP)
      INTEGER PEND/'PEND'/,NPRO
      LOGICAL LP
      READ(5,100)NODES

  100 FORMAT(2I10,F10.0)
C
C     INITIALIZES ARC DISTANCE MATRIX
C
      DO 1 I=1,NODES
      TL(I)=1.E10
      PL(I)=1.E10
      LP(I) = .FALSE.
      DO 1 J = 1,NODES
    1 D(I,J) = 1.E10
C
C     READ THE INPUT DATA

C
    2 READ(5,120) NPRO,I,J,VAL
  120 FORMAT(A4,I6,I10,F10.2)
      IF(NPRO.EQ.PEND) GO TO 10
    5 D(I,J) = VAL
      GO TO 2
   10 LP(1) = .TRUE.
      WRITE(6,500)
  500 FORMAT(1H1)
      WRITE(6,600)
  600 FORMAT(25X,'SHORTEST ROUTE PROBLEM'//)
      PL(1) =0.

      LASTLP = 1
C
C     PRINTS THE INPUT DATA
C
      WRITE(6,101)
  101 FORMAT(' ',7X,'DIJKSTRA S ALGORITHM TO FIND THE SHORTEST ROUTE'//'
     @        FROM AN ORIGIN TO ALL THE OTHER POINTS'//'        FROM NOD
     @E   TO NODE   IS A DISTANCE OF '//)
      DO 15 I = 1,NODES
      DO 15 J = 1,NODES
      IF(D(I,J).NE.1.E10) PRINT 200,I,J,D(I,J)
  200 FORMAT(5X,2I10,F10.2)
   15 CONTINUE
```

```
      DO 20 I= 2,NODES
C
C STEP A
C
      DO 16 J = 1,NODES
      IF(LP(J)) GO TO 16
      IF(PL(LASTLP) + D(LASTLP,J).LT.TL(J)) TL(J)=PL(LASTLP)+
     * D(LASTLP,J)
   16 CONTINUE
C
C STEP B
C

      MIN = 2
      XMIN = TL(2)
      DO 17 J = 3,NODES
      IF(TL(J).GT.XMIN) GO TO 17
      MIN = J
      XMIN = TL(J)
   17 CONTINUE
      PL(MIN) = XMIN
      TL(MIN) = 1.E10
      LP(MIN) = .TRUE.
   20 LASTLP = MIN
      DO 30 I = 2,NODES

   30 CALL TRACED(I)
   12 CONTINUE
      RETURN
      END
      SUBROUTINE TRACED(IT)
      DIMENSION NODE(50,50),A(50,50),LAB(50),LP(50),EPS(50),PL(50),
     1NICID(50,50),SCAN(50),TL(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      EQUIVALENCE (NODE,A,S),(LAB,LP,ROWUSD),(EPS,PL,COLUSD),(NICID,UP)
     1,(SCAN,TL,NP)
      INTEGER OPTPAT(50)

      EQUIVALENCE (OPTPAT,INFSTC(1,2))
      LOGICAL LP
      IX = IT
      OPTPAT(1) = IX
      K = 2
      PRINT 100, IX, PL(IX)
    1 DO 10 I = 1,NODES
      IF(IX.EQ.1) GO TO 10
      IF(PL(IX) - PL(I).EQ.D(I,IX)) GO TO 20
   10 CONTINUE
   20 OPTPAT(K) = I
      IF(I.EQ.1) GO TO 30

      IX = I
      K = K+1
      GO TO 1
   30 L = K-1
      DO 40 I = 1,L
      M = K-I
      N1 = OPTPAT(M+1)
      N2 = OPTPAT(M)
   40 PRINT 101, N1, N2, D(N1,N2)
      RETURN
  100 FORMAT(////,7X,'THE OPTIMAL ROUTE FROM NODE 1 TO NODE ',I5,
     @'HAS A SOLUTION OF ',F10.2,'AND',//,7X,'THIS IS ATTAINED BY THE
     *FOLLOWING ROUTE'//)
```

```
  101 FORMAT(' ',7X,'FROM NODE',I7,'TO NODE',I7,'IS A DISTANCE OF ',F10.
     @2)
      END
C
      SUBROUTINE MULTSH
C
      DIMENSION NODE(50,50),A(50,50),LAB(50),LP(50),EPS(50),PL(50),
     1NICID(50,50),SCAN(50),TL(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      EQUIVALENCE (NODE,A,S),(LAB,LP,ROWUSD),(EPS,PL,COLUSD),(NICID,UP)
     1,(SCAN,TL,NP)

      INTEGER PEND/'PEND'/,NPRO
      IIN=5
      IOT=6
      N=0
      READ(IIN,400)NODES
  400 FORMAT(I10)
C
C     INITIALIZATION OF THE DISTANCE MATRIX.
C
      DO 1 L=1,NODES
      DO 1 M=1,NODES
      D(L,M)=9999999.99

      IF(L.EQ.M) D(L,M)=0.0
    1 CONTINUE
C
C     DATA READ IN
C
    2 READ(IIN,3) NPRO,I,J,B
      IF(NPRO.EQ.PEND) GO TO 10
    3 FORMAT(A4,I6,I10,F10.2)
      D(I,J)=B
      IF(J.GT.N) N=J
      IF(I.GT.N) N=I
      GO TO 2

   10 CONTINUE
      WRITE(IOT,600)
  600 FORMAT('1',10X,' MULTITERMINAL SHORTEST CHAINS PROBLEM')
      WRITE(IOT,700)
  700 FORMAT('0',10X,'*** ORIGINAL MATRIX***')
      CALL PRNTRX
C
C     INITIALIZATION OF THE NODES ON THE CHAINS MATRIX.
C
      DO 100 I=1,N
      DO 100 K=1,N
      NODE(I,K)=K
  100 CONTINUE
C
C     TRIPLE OPERATION TO UPDATE THE DISTANCE MATRIX.
C
      DO 20 J=1,N
      DO 20 I=1,N
      DO 20 K=1,N
      IF(I.EQ.J.OR.I.EQ.K.OR.J.EQ.K) GO TO 20
      X=D(I,J)+D(J,K)
      Y=D(I,K)
      IF(Y.LE.X) GO TO 20
      D(I,K)=X
C
```

```
C   UPDATING THE NODES ON THE CHAINS MATRIX.
C
      NODE(I,K)=NODE(I,J)
   20 CONTINUE
      WRITE(IOT,800)
  800 FORMAT('1',5X,'** SHORTEST DISTANCE MATRIX **')
      CALL PRNTRX
      WRITE(IOT,900)
  900 FORMAT('1',5X,'#MATRIX REPRESENTING THE NODES ON THE SHORTEST CHAI
     @NS')
      CALL PRNTRY
      RETURN

      END
C
      SUBROUTINE PRNTRX
      DIMENSION NODE(50,50),A(50,50),LAB(50),LP(50),EPS(50),PL(50),
     1NICID(50,50),SCAN(50),TL(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      EQUIVALENCE (NODE,A,S),(LAB,LP,ROWUSD),(EPS,PL,COLUSD),(NICID,UP)
     1,(SCAN,TL,NP)
      N1=1
      N2=5
   43 IF(N2-N) 45,45,44

   44 N2=N
   45 PRINT 911,(I,I=N1,N2)
  911 FORMAT(//,9X,5(I10,2X))
      WRITE(6,912)
  912 FORMAT(/)
      DO 48 L=1,N
   48 PRINT 913,L,(D(L,M),M=N1,N2)
      IF(N2-N) 52,55,55
   52 N1=N1+5
      N2=N2+5
      GO TO 43
  913 FORMAT(' ',5X,I2,2X,5(F10.2,2X))

   55 CONTINUE
      RETURN
      END
      SUBROUTINE PRNTRY
      DIMENSION NODE(50,50),A(50,50),LAB(50),LP(50),EPS(50),PL(50),
     1NICID(50,50),SCAN(50),TL(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      EQUIVALENCE (NODE,A,S),(LAB,LP,ROWUSD),(EPS,PL,COLUSD),(NICID,UP)
     1,(SCAN,TL,NP)
      N1=1
      N2=15

   43 IF(N2-N) 45,45,44
   44 N2=N
   45 PRINT 911,(I,I=N1,N2)
  911 FORMAT(//,10X,15(I3))
      WRITE(6,912)
  912 FORMAT(/)
      DO 48 L=1,N
   48 PRINT 913,L,(NODE(L,M),M=N1,N2)
      IF(N2-N) 52,55,55
   52 N1=N1+15
      N2=N2+15
      GO TO 43
  913 FORMAT(' ',4X,I2,3X,15(I3))
```

```
   55 CONTINUE
      RETURN
      END
      SUBROUTINE MINSPA
      DIMENSION NODE(50,50),A(50,50),LAB(50),LP(50),EPS(50),PL(50),
     1NICID(50,50),SCAN(50),TL(50)
      DIMENSION ICON(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      EQUIVALENCE (NODE,A,S),(LAB,LP,ROWUSD),(EPS,PL,COLUSD),(NICID,UP)
     1,(SCAN,TL,NP)
      EQUIVALENCE (ICON,UP)

      INTEGER PEND/'PEND'/,NPRO
      IIN=5
      READ(5,31)NODES
   31 FORMAT(I10)
C
C     INITIALIZATION AND REINITIALIZATION
C
      DO 1 I=1,NODES
      DO 1 J=1,NODES
      NICID(I,J)=0
    1 D(I,J)=0.0
      NN=0

C
C     DATA READ IN
C
  100 READ(IIN,2) NPRO,I,J,A1
    2 FORMAT(A4,I6,I10,F10.2)
      IF(NPRO.EQ.PEND) GO TO 1000
      NICID(I,J)=1
      D(I,J)=A1
      IF(J.GT.NN)NN=J
      GO TO 100
 1000 IOT=6
C

C     NODE ONE IS INCLUDED IN THE TREE TO START WITH
C
 1060 M=1
      ICON(1)=1
 1071 HOLD=999999.0
      IF(M.GT.NN) GO TO 1081
C
C     SEARCH FOR NEW CANDIDATES AND CONNECT THE BEST CANDIDATE
C
      DO 108 I=1,M
      K=ICON(I)
      DO 108 J=1,NN

      DO 1072 L=1,M
      IF(J.EQ.ICON(L)) GO TO 108
 1072 CONTINUE
      IF(K-J)1073,108,1074
 1073 IF(NICID(K,J).EQ.0) GO TO 108
      IF(D(K,J).GE.HOLD) GO TO 108
      HOLD=D(K,J)
      GO TO 1075
 1074 IF(NICID(J,K).EQ.0) GO TO 108
      IF(D(J,K).GE.HOLD) GO TO 108
      HOLD=D(J,K)
 1075 IO=K
      JO=J
```

```
      GO TO 111
  110 EPS(I)=FLO(J,I)
  111 IF(LAB(NN).NE.0) GO TO 1145
  112 CONTINUE
      SCAN(J)=1
  114 CONTINUE
  113 CONTINUE
C
C     CHECK TO SEE IF ALL NODES ARE SCANNED
C
      DO 1131 I=1,NN
      IF(LAB(I).EQ.0) GO TO 1131

      IF(SCAN(I).NE.0) GO TO 1131
      GO TO 105
 1131 CONTINUE
      GO TO 990
C
C     SINK NODE LABELLED, INCREASE FLOW
C
 1145 ADD=EPS(NN)
  115 JK=LAB(NN)
      IF(LAB(NN).GT.0) GO TO 116
      FLO(JK,NN)=FLO(JK,NN)-ADD
      GO TO 117

  116 FLO(JK,NN)=FLO(JK,NN)+ADD
      NN=LAB(NN)
  117 IF(LAB(NN).EQ.9999) GO TO 1171
      GO TO 115
C
C     BACK AT THE BEGINNING, REINITIALIZE
C
 1171 NN=N
C
C     A FEASIBLE FLOW HAS BEEN FOUND PRINT SO.
C
      K=3

      GO TO 201
 2000 CONTINUE
C
C     REINITIALIZE THE REQUIRED
C
      DO 118 I=2,NN
      LAB(I)=0
  118 SCAN(I)=0
      SCAN(1)=0
      GO TO 105
  990 K=2
      NN=N

      GO TO 201
 3000 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE OKALG
C
      DIMENSION I(500),J(500),HI(500),LO(500),FLOW(500),PI(500),COST
     @(500)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      EQUIVALENCE(I,S),(J,S(501)),(HI,S(1001)),(LO,S(1501)),
```

```
      CAP(I,J)=A1
      FLO(I,J)=A2
      IF(J.GT.NN)NN=J
      GO TO 102
  104 LAB(1)=9999.0
      EPS(1)=9999.0
      N=NN
      K=1
  201 WRITE(IOT,3)
      GO TO (10,20,30)
   10 WRITE(IOT,4)
      GO TO 991

   20 WRITE(IOT,5)
      GO TO 991
   30 WRITE(IOT,8)
  991 FLOMAX=0.0
      WRITE(IOT,6)
      DO 992 I=1,NN
      DO 993 J=1,NN
      IF(NICID(I,J).EQ.0) GO TO 993
      WRITE(IOT,7)I,J,CAP(I,J),FLO(I,J)
      IF(J.EQ.NN) FLOMAX=FLOMAX+FLO(I,J)
  993 CONTINUE
  992 CONTINUE

      WRITE(IOT,1) FLOMAX
    1 FORMAT('0',5X,'MAXIMUM FLOW THROUGH THE ABOVE ARCS IS =',F14.2)
    3 FORMAT('1',30X,'MAX/FLOW PROBLEM')
    4 FORMAT('0',5X,'STATUS OF NETWORK AT START')
    5 FORMAT('0',5X,'STATUS OF NETWORK AT OPTIMAL SOLUTION')
    6 FORMAT('0',5X,'STARTING NODE',2X,'ENDING NODE',2X,'CAPACITY',
     14X,'FLOW IN ARC')
    7 FORMAT(1H0,3X,I10,3X,I10,F10.2,3X,F10.2)
    8 FORMAT('0',5X,'STATUS OF NETWORK AT A FEASIBLE SOLUTION')
      IF(K.EQ.2) GO TO 3000
      IF(K.EQ.3) GO TO 2000
  105 DO 113 J=1,NN

C
C     FIND A LABELED NODE.
C
  106 IF(LAB(J).EQ.0) GO TO 114
C
C     FOUND A LABELED NODE, IS IT SCANNED?, IF 'YES' CONTINUE,IF NO, SCAN IT.
C
      IF(SCAN(J).NE.0) GO TO 114
C     FOUND LABELLED,UNSCANNED NODE, SCAN IT .
  108 DO 112 I=1,NN
      IF(NICID(J,I).EQ.0) GO TO 112
  109 IF(LAB(I).NE.0) GO TO 112

      PS=CAP(J,I)-FLO(J,I)
      IF(PS.GT.0.0) GO TO 1091
      IF(NICID(I,J).EQ.0) GO TO 112
      IF(FLO(I,J).GE.0.0) GO TO 112
      LAB(I)=-J
      IF(FLO(J,I).LT.EPS(J)) GO TO 110
      EPS(I)=EPS(J)
      GO TO 111
 1091 LAB(I)=J
      IF(PS.LT.EPS(J)) GO TO 1092
      EPS(I)=EPS(J)
      GO TO 111
 1092 EPS(I)=PS
```

```
  108 CONTINUE
      M=M+1
      IF(IO.GT.JO) GO TO 1082
C
C     THE CONNECTED ARC IS INDEXED
C
      NICID(IO,JO)=2
 1082 NICID(JO,IO)=2
 1083 ICON(M)=JO
      GO TO 1071
 1081 CONTINUE
C

C     OUTPUT ROUTINE FOR MINIMAL SPANNING TREE
C
  300 WRITE(IOT,331)
  331 FORMAT('1',20X,'MINIMAL SPANNING TREE PROBLEM')
      WRITE(IOT,431)
  431 FORMAT('0',5X,' THE MINIMAL SPANNING TREE CONSISTS OF THE FOLLOWI
     @NG ARCS')
      WRITE(IOT,34)
   34 FORMAT('0',5X,'STARTING NODE',2X,'ENDING NODE',2X,'DISTANCE')
      DO 302 I=1,NN
      DO 302 J=1,NN
      IF(I.GT.J) GO TO 302

      IF(NICID(I,J).EQ.2) WRITE(IOT,7)I,J,  D(I,J)
  302 CONTINUE
    7 FORMAT(1HO,3X,I10,3X,I10,F10.2,3X,F10.2)
      RETURN
      END
.C
      SUBROUTINE MAXFLO
      DIMENSION FLO(50,50),CAP(50,50)
      DIMENSION NODE(50,50),A(50,50),LAB(50),LP(50),EPS(50),PL(50),
     1NICID(50,50),SCAN(50),TL(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX

      EQUIVALENCE (NODE,A,S),(LAB,LP,ROWUSD),(EPS,PL,COLUSD),(NICID,UP)
     1,(SCAN,TL,NP)
      EQUIVALENCE (FLO,S),(CAP,D)
      INTEGER SCAN,NICID
      INTEGER PEND/'PEND'/,NPRO
      IIN=5
      IOT=6
C
C     INITIALIZATION
C
      READ(5,100) N
  100 FORMAT(I10)

      DO 101 I=1,N
      LAB(I)=0
      EPS(I)=0
      SCAN(I)=0
      DO 101 J=1,N
      CAP(I,J)=0.0
      FLO (I,J)=0.0
  101 NICID(I,J)=0
      NN=0
  102 READ(IIN,9) NPRO,I,J,A1,A2
    9 FORMAT(A4,I6,I10,2F10.2)
      IF(NPRO.EQ.PEND) GO TO 104
  103 NICID(I,J)=1
```

```
      @(FLOW,S(2001)),(PI,D),(COST,D(501))
       INTEGER ARCS, FLOW, PL, COST, HI
       LOGICAL INFES
  100 FORMAT(2I10)
C
C      READS THE TOTAL NUMBER OF NODES AND ARCS FOR THE PROBLEM
C
       READ 100,NODES,ARCS
C
C      READS THE UPPER LIMIT, LOWER LIMIT AND COST FOR EACH
C      OF THE ARCS.
C

       DO 1 M=1,ARCS
       READ (5,250)I(M),J(M),T1,T2,T3
       HI(M)=T1+0.00001
       LO(M)=T2+0.00001
       COST(M)=T3+0.00001
    1 CONTINUE
  250 FORMAT(2I5,3F10.2)
C
C      INITIALIZES ARC FLOWS
C
       DO 5 M=1,ARCS
    5  FLOW(M)=0

C
C      INITIALIZES PI VALUES
C
       DO 10 M=1,NODES
   10  PI(M)=0
C
C      CALLS THE SUBROUTINE WHICH EVALUTES THE FEASIBLE NETFLOW
C           FOR EACH ARC
C
       CALL NETFLO(ARCS,INFES)
       IF(.NOT.INFES)WRITE(6,120)
C

C      PRINTS THE FINAL SUMMARY REPORT
C
       WRITE(6,801)
  801  FORMAT('1',12X,'****SOLUTION BY OUT-OF-KILTER ALGORITHM****')
       WRITE(6,112)
  112  FORMAT(//,20X,'FINAL SUMMARY REPORT ',///)
       WRITE(6,115)NODES,ARCS,(M,I(M),J(M),HI(M),LO(M),FLOW(M),COST(M),M=
      @1,ARCS)
  115  FORMAT(' ',8X,'NUMBER OF NODES =',I5,/,' ',8X,'NUMBER OF ARCS  =',
      1I5,///,12X,'M',5X,'I',5X,'J',10X,'HI',11X,'LO',10X,'FLOW',10X,'COS
      2T',///,(7X,3(2X,I4),4(3X,I10)))
       TCOST=0.0

       DO 140 M=1,ARCS
       TCOST=COST(M)*FLOW(M)+TCOST
  140  CONTINUE
       WRITE(6,150)TCOST
  150  FORMAT(//,'          TOTAL PROJECT COST =',F10.2)
       WRITE(6,125)(M,FLOW(M),M=1,ARCS)
       WRITE(6,130 (M,PI(M),M=1,NODES)
  120  FORMAT('  SOLUTION INFEASIBLE')
  125  FORMAT(///,5X,'  ARC          FLOW(ARC)',/,(5X,I4,6X,I10))
  130  FORMAT(///,5X,'  NODE       PI(NODE)',/,(4X,I4,2X,I10))
       WRITE(6,750)
  750  FORMAT(//,5X,'   SENSITIVITY ANALYSIS',///)
       WRITE(6,753)
```

```
753 FORMAT(12X,'M',5X,'I',5X,'J',10X,'HI',10X,'LO',10X,'FLOW',10X,'COS
   @T',//)
     DO 751 M=1,ARCS
     IZ=FLOW(M)
     IF(IZ.EQ.0)GO TO 751
     WRITE(6,752)M,I(M),J(M),HI(M),LO(M),FLOW(M),COST(M)
752  FORMAT(7X,3(2X,I4),4(3X,I10))
751  CONTINUE
     RETURN
     END
     SUBROUTINE NETFLO(ARCS,INFES)
     DIMENSION I(500),J(500),HI(500),LO(500),FLOW(500),PI(500),COST

    @(500),NA(500),NB(500)
     COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
    1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
     EQUIVALENCE(I,S),(J,S(501)),(HI,S(1001)),(LO,S(1501)),
    @(FLOW,S(2001)),(PI,D),(COST,D(501)),(NA,D(1001)),(NB,D(1501))
     LOGICAL INFES
     INTEGER A,AOK,C,COK,DEL,E,EPS,INF,LAB,N,NI,NJ,SRC,SNK,
    1FLOW,PI,NA, NODES,ARCS,I,J,COST,HI,LO,HB
C
C    CHECK FEASIBILITY OF FORMULATION
C
         INFES=.TRUE.

     DO 10 A=1,ARCS
     IF(LO(A).GT.HI(A)) GO TO 39
  10 CONTINUE
C
C    SET INF TO MAX AVAILABLE INTEGER
C
  16     INF=999999
         AOK=0
C
C    FIND OUT OF KILTER ARC
C
  20 DO 21 A=1,ARCS

     IA = I(A)
     JA = J(A)
         C=COST(A)+PI( IA )-PI( JA )
C
C    CHECKS THE CONDITIONS THE INDIVIDUAL ARCS ARE IN.
C
     IF((FLOW(A).LT.LO(A)).OR.(C.LT.0.AND.FLOW(A).LT.HI(A))) GO TO 22
     IF((FLOW(A).GT.HI(A)).OR.(C.GT.0.AND.FLOW(A).GT.LO(A))) GO TO 23
  21 CONTINUE
C
C    NO REMAINING OUT OF KILTER ARCS
C

     GO TO 38
  22     SRC=J(A)
         SNK=I(A)
         E=+1
     GO TO 24
  23     SRC=I(A)
         SNK=J(A)
         E=-1
     GO TO 24
  24 DO 99 N=1,NODES
         NA(N)= 0
         NB(N)=0
  99 CONTINUE
```

```
      IF((A.EQ.AOK).AND.(NA( RC).NE.O)) GO TO 25
C
C     ATTEMPT TO BRING OUT OF KILTER ARCS INTO KILTER (
C
          AOK=A
      DO 26 N=1,NODES
          NA(SRC)=IABS(SNK)*E
   26     NB(SRC)=IABS(AOK)*E
   25     COK=C
   27     LAB=O
      DO 30 A=1,ARCS
      IA = I(A)

      JA = J(A)
      IF((NA(IA).EQ.O.AND.NA(JA).EQ.O).OR.(NA(IA).NE.O..AND.NA(JA).NE.O)
     1)GO TO 30
      C= COST(A)+PI(IA) - PI(JA)
      IF(NA(IA).EQ.O) GO TO 28
      IF(FLOW(A).GE.HI(A).OR.(FLOW(A).GE.LO(A).AND.C.GT.O))GO TO 30
      NA(JA) = I(A)
      NB(JA) = A
      GO TO 29
   28 IF(FLOW(A).LE.LO(A).OR.(FLOW(A).LE.HI(A).AND.C.LT.O))GO TO 30
      IA = I(A)
      NA(IA) = -J(A)

      NB(IA) = -A
   29 LAB = 1
C
C NODE LABELED, TEST FOR BREAKTHRU
C
          IF(NA(SNK).NE.O) GO TO 33
   30 CONTINUE
C
C NO BREAKTHRU
C
          IF(LAB.NE.O) GO TO 27
C DETERMINE CHANGE TO PI VECTOR

          DEL = INF
          DO 31 A=1,ARCS
          IA = I(A)
          JA = J(A)
          IF((NA(IA).EQ.O.AND.NA(JA).EQ.O).OR.(NA(IA).NE.O.AND.NA(JA).NE.O))
         1GO,TO 31
          C=COST(A)+PI(IA)-PI(JA)
          IF(NA(JA).EQ.O.AND.FLOW(A).LT.HI(A)) DEL= MINO(DEL,C)
          IF(NA(JA).NE.O.AND.FLOW(A).GT.LO(A)) DEL= MINO(DEL,-C)
   31 CONTINUE
          IF(DEL.EQ.INF.AND.(FLOW(AOK).EQ.HI(AOK).OR.FLOW(AOK).EQ.LO(AOK)))
         1DEL=IABS(COK)

          IF(DEL.EQ.INF) GO TO 39
C
C EXIT, NO FEASIBLE FLOW PATTERN
C CHANGE PI VECTOR BY COMPUTED DEL
C
          DO 32 N=1,NODES
   32 IF(NA(N).EQ.O) PI(N)=PI(N)+DEL
C
C FIND ANOTHER OUT-OF-KILTER ARC
C
          GO TO 20
C
C BREAKTHRU  COMPUTE INCREMENTAL FLOW
```

```
C
   33 EPS=INF
      NI=SRC
   34 NJ=IABS(NA(NI))
      A=IABS(NB(NI))
      C=COST(A)-ISIGN(IABS(PI(NI) -PI(NJ)),NB(NI))
      IF(NB(NI).LT.0) GO TO 35
      IF(C.GT.0.AND.FLOW(A).LT.LO(A)) EPS=MINO(EPS,LO(A)-FLOW(A))
      IF(C.LE.0.AND.FLOW(A).LT.HI(A)) EPS=MINO(EPS,HI(A)-FLOW(A))
      GO TO 36
   35 IF(C.LT.0.AND.FLOW(A).GT.HI(A)) EPS=MINO(EPS,FLOW(A)-HI(A))
      IF(C.GE.0.AND.FLOW(A).GT.LO(A)) EPS=MINO(EPS,FLOW(A)-LO(A))

   36 NI=NJ
      IF(NI.NE.SRC) GO TO 34
C
C CHANGE FLOW VECTOR BY COMPUTED EPS
C
   37 NJ=IABS(NA(NI))
      A=IABS(NB(NI))
      FLOW(A)=FLOW(A)+ISIGN(EPS,NB(NI))
      NI=NJ
      IF(NI.NE.SRC) GO TO 37
C
C FIND ANOTHER OUT OF KILTER ARC

C
      AOK=0
      GO TO 20
   39 INFES = .FALSE.
   38 CONTINUE
      RETURN
      END
      SUBROUTINE TRASAL
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      INTEGER S,D,TREE,UP
      INTEGER PEND/'PEND'/,NPRO

      LOGICAL ROWUSD,COLUSD,FLAG
C
C     INITIALIZE
C
      INF=99999999
      MING=INF
      READ(5,32)NODES
   32 FORMAT(I10)
      DO 1 I=1,NODES
      INFSTC(I,1)=INF
      ROWUSD(I)=.FALSE.
      COLUSD(I)=.FALSE.

      TREE(I,1)=0
      TREE(I,2)=0
      DO 1 J=1,NODES
      UP(I,J)=0
      S(I,J)=INF
1     D(I,J)=INF
C
C         GET DATA
C
    2 READ(5,104) NPRO,I,J,VAL
      IF(NPRO.EQ.PEND) GO TO 11
    5 D(I,J)=VAL
      S(I,J)=VAL
```

```
      GO TO 2
   11 LEVEL=1
      WRITE(6,106)
  106 FORMAT('1',15X,'TRAVELLING SALESMAN PROBLEM****BY BRANCH AND BOUND
     1',////)
      WRITE(6,99)
C
      CALL PRTRXS
C
C
C           REDUCE THE MATRIX TO OBTAIN A LOWER BOUND ON THE SOLUTION
C
      MIN= MINCOL(I)+MINROW(I)
   13 M=0
      N=0
      MAX=0
C
C           THIS LOOP FINDS THE ARC OF VALUE 0  THAT ALLOWS
C           MAXIMUM SAVINGS
C
      DO 31 I=1,NODES
      IF(ROWUSD(I)) GO TO 31
      DO 20 J=1,NODES
      IF(COLUSD(J)) GO TO 20
      IF(D(I,J).NE.0) GO TO 20
      IF(M.NE.0) GO TO 17
      M=I
      N=J
   17 MT=INF
      DO 18 K=1,NODES
      IF(ROWUSD(K)) GO TO 18
      IF(D(K,J).GE.MT) GO TO 18
      IF(K.EQ.I) GO TO 18
      MT=D(K,J)
   18 CONTINUE
      MAXT=MT
      MT=INF
      DO 19 K=1,NODES
      IF(COLUSD(K)) GO TO 19
      IF(D(I,K).GE.MT) GO TO 19
      IF(K.EQ.J) GO TO 19
      MT=D(I,K)
   19 CONTINUE
      MAXT=MAXT+MT
      IF(MAXT.LE.MAX) GO TO 20
      MAX= MAXT
      M=I
      N=J
   20 CONTINUE
   31 CONTINUE
C
C           INDICATE THAT THE OPTIMUM ARC FROM M TO N IS IN THE SOLUTION
C
      ROWUSD(M)=.TRUE.
      COLUSD(N)=.TRUE.
      UP(M,N)= LEVEL
C
C           IF THE MATRIX HAS BEEN REDUCED TO A 2X2, NO MODIFICATION IS
C           NEEDED. HOWEVER, IF NOT, THE MATRIX MUST BE ALTERED SO THAT
C           CYCLES WILL NOT OCCUR, CAUSING A PREMATURE RETURN TO THE
C           ORIGINAL NODE
```

```
C
      IF(LEVEL.GE.(NODES-1)) GO TO 30
C
C           FIRST, THE CHAIN CURRENTLY BEING ADDED TO THE SOLUTION MUST BE
C           EXTENDED FORWARD AS FAR AS POSSIBLE. VIA THE ARCS  ALREADY IN
C           THE SOLUTION.
C
      L=N
   21 DO 22 I=1,NODES
      IF(UP(L,I).EQ.0) GO TO 22
      GO TO 23
   22 CONTINUE

      GO TO 25
   23 L=I
      GO TO 21
C
C           NEXT THE ARC CURRENTLY BEING ADDED  TO THE SOLUTION MUST BE
C           EXTENDED BACKWARD AS FAR AS POSSIBLE VIA THE ARCS ALREADY IN
C           THE SOLUTION.
C
   25 K=M
   26 DO 27 I=1,NODES
      IF(UP(I,K).EQ.0) GO TO 27
      GO TO 28

   27 CONTINUE
      GO TO 29
   28 K=I
      GO TO 26
C
C           AT THIS POINT , IF AN ARC FROM L TO K IS ADDED TO THE SOLUTION
C           A CYCLE WILL OCCUR, SO WE MUST BLOCK THAT ARC FROM ENTERING.
C           THE SOLUTION
C
   29 D(L,K)=INF
C
C          WE MUST SET UP A TREE, THE LEET HAND BRANCH (1) HAS THE VALUE

C          OF THE LOWER BOUND ON THE SOLUTION IF ARC (M,N) IS NOT USED
C          THE RIGHT HAND BRANCH (2) HAS THE VALUE OF THE LOWER BOUND
C          ON THE SOLUTION IF ARC (M,N) IS IN THE SOLUTION.
C
C
C            INDICATE THAT THE LEFT HAND BRANCH OF THE TREE HAS A VALUE
C            OF MIN+MAX, THAT IS, THE CURRENT LOWER BOUND PLUS THE
C            SAVINGS FROM USING ARC(M,N).
C
   30 TREE(LEVEL,1)=MIN+MAX
C
C          REDUCE THE MATRIX FURTHER(IF POSSIBLE) AND COMPUTE THE NEW

C          LOWER BOUND.
C
      MIN=MIN+MINCOL(I)+MINROW(I)

C
C           SET THE RIGHT HAND SIDE OF THE TREE TO THE NEW LOWER BOUND
C
      TREE(LEVEL,2)=MIN
      LEVEL=LEVEL+1
C
C           IF THE LOWER BOUND CURRENTLY IN EFFECT IS ALREADY GREATER
C           THAN THE GLOBAL MINIMUM ALREADY FOUND , THEN WE DO NOT
C           NEED TO PROCEED ANY FURTHER.
C
```

```
      IF(MIN.GE.MING) GO TO 60
C
C          IF THE MATRIX IS NOT FULLY ELIMINATED, GO BACK AND DO MORE
C          WORK
C
      IF(LEVEL.LE.NODES) GO TO 13
C
C        THE SOLUTION JUST FOUND IS BETTER THAN OUR PREVIOUS GLOBAL
C        MINIMUM. SO WE RESET MING AND SAVE THE ROUTE THAT WE TOOK.
C
C
      MING=MIN

      IS=1
      DO 50 I=1,NODES
      NP(I)=IS
      DO 45 J=1,NODES
      IF(UP(IS,J).EQ.0) GO TO 45
      GO TO 50
   45 CONTINUE
      PRINT 102,I,J,IS
   50 IS=J
      NP(NODES+1)=IS
      PRINT 105,MIN
      PRINT 103,(NP(I),I=1,NODES)

      GOTO 91
C
C          NOW WE MUST CHECK THE LEFT HAND BRANCHES OF THE TREE TO SEE
C          IF ANY OTHER ROUTE MIGHT FEASIBLY LEAD TO A BETTER SOLUTION.
C
   60 LEVEL1=LEVEL-1
      DO 65 I=1,LEVEL1
      IF(TREE(LEVEL-I,1).LT.MING) GO TO 70
   65 CONTINUE
C
C          SINCE ALL LEFT HAND BRANCHES OF THE TREE ARE GREATER THAN
C          THE GLOBAL MINIMUM, NO OTHER ROUTE IS FEASIBLE, AND WE ARE

C          FINISHED.
C
      GO TO 91
C
C        IT IS POSSIBLE THAT THIS BRANCH OF THE TREE WILL OFFER A
C        BETTER SOLUTION , SO WE MUST BLOCK THE ARC THAT WAS USED
C        AT THIS LEVEL FROM BEING USED IN THE NEW SOLUTION BY SETTING
C        THE DISTANCE TO INFINITY. RESET ALL INITIAL VALUES, AND GO
C        BACK TO THE START.
C
   70 LEVEL=LEVEL-I
      DO 75 I=1,NODES

      ROWUSD(I)=.FALSE.
      COLUSD(I)=.FALSE.
      TREE(I,1)=0
      TREE(I,2)=0
      DO 75 J=1,NODES
      D(I,J)=S(I,J)
      IF(UP(I,J).NE.LEVEL) GO TO 75
      M=I
      N=J
   75 UP(I,J)=0
C
C
C
```

```
C        IT IS IMPORTANT TO OBSERVE THAT IF ARC(I,J) HAS BEEN ELIMINATED
C        FROM THE SOLUTION, IT IS SET TO INFINITY. THEN AFTER PROCEEDING
C        DOWN, THE MIN WILL PERHAPS EXCEED MING AND IT IS TIME TO STOP
C        THAT TREE BUT AS YOU GO BACK AND CHECK THE LEFT HAND BRANCHES
C        PERHAPS AN ARC AFTER THE POINT IN THE TREE WHERE ARC (I,J) WAS
C        PREVIOUSLY SET TO INFINITY SHOULD NOW BE ELIMINATED. IN THIS
C        CASE ONE MUST BE CAREFUL TO LEAVE THE VALUE OF ARC(I,J) AT
C        INFINITY.
C
C
C
      PASS=0.

      DO 90  I=1,NODES
      IF(INFSTC(I,1).LE.LEVEL) GO TO 80
      IF(PASS.NE.0.) GO TO 85
      PASS=1.
      INFSTC(I,1)=LEVEL
      INFSTC(I,2)=M
      INFSTC(I,3)=N
   80 II=INFSTC(I,2)
      IJ=INFSTC(I,3)
      D(II,JJ)=INF
      GO TO 90
   85 INFSTC(I,1)=INF

   90 CONTINUE
      GO TO 11
   91 PRINT 100, MING
      DO 95 I=1,NODES
      NY=NP(I)
      NZ=NP(I+1)
   95 PRINT 101, NY, NZ, S(NY,NZ)
   99 FORMAT(' ',8X,'***ORIGINAL MATRIX***')
  100 FORMAT('1',8X,'WE HAVE AN OPTIMAL SOLUTION OF ',I14,//40X,'BY',/)
  101 FORMAT(' ',8X,'GO FROM NODE',I3,5X,'TO NODE',I3,5X,'AT A DISTANCE
     @OF',I10)
  102 FORMAT(' ','ROUTE UNDEFINED',3I10)

  103 FORMAT(' ',2X,10(3X,I4)/10(3X,I4)/10(3X,I4)/10(3X,I4)/10(3X,I4)/
     15(3X,I4)/)
  104 FORMAT(A4,I6,I10,F10.2)
  105 FORMAT(///,8X,'**WE HAVE A FEASIBLE SOLUTION **',I14,///)
      RETURN
      END

          COLUMN REDUCTION

      FUNCTION MINCOL(KKK)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX

      INTEGER S,D,TREE,UP
      LOGICAL ROWUSD, COLUSD,FLAG
      MINCOL=0
      DO 50 I=1,NODES
      MIN=0
      IF(COLUSD(I)) GO TO 50
      MIN=INF
      DO 10 J=1,NODES
      IF(ROWUSD(J)) GO TO 10
      IF(D(J,I).GT.MIN) GO TO 10
      MIN=D(J,I)
   10 CONTINUE
      IF(MIN.EQ.0) GO TO 50
```

```
      DO 20 J=1,NODES
   20 D(J,I)=D(J,I)-MIN
   50 MINCOL=MINCOL+MIN
      RETURN
      END
C
C          ROW REDUCTION
C
      FUNCTION MINROW(KKK)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      INTEGER S,D,TREE,UP

      LOGICAL ROWUSD,COLUSD,FLAG
      MINROW=0
      DO 50 I=1,NODES
      MIN=0
      IF(ROWUSD(I)) GO TO 50
      MIN=INF
      DO 10 J=1,NODES
      IF(COLUSD(J)) GO TO 10
      IF(D(I,J).GT.MIN) GO TO 10
      MIN=D(I,J)
   10 CONTINUE
      IF(MIN.EQ.0) GO TO 50

      DO 20 J=1,NODES
   20 D(I,J)=D(I,J)-MIN
   50 MINROW=MINROW+MIN
      RETURN
      END
      SUBROUTINE PRTRXS
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      INTEGER S,D,TREE,UP
      N=NODES
      N1=1
      N2=5

   43 IF(N2-N) 45,45,44
   44 N2=N
   45 PRINT 911,(I,I=N1,N2)
  911 FORMAT(//,9X,5(I10,2X))
      WRITE(6,912)
  912 FORMAT(/)
      DO 48 L=1,N
   48 PRINT 913,L,(D(L,M),M=N1,N2)
      IF(N2-N) 52,55,55
   52 N1=N1+5
      N2=N2+5
      GO TO 43

  913 FORMAT(' ',6X,I2,5(I10,2X))
   55 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE MAXCAP
      DIMENSION NODE(50,50),A(50,50),LAB(50),LP(50),EPS(50),PL(50),
     1NICID(50,50),SCAN(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      EQUIVALENCE (NODE,A,S),(LAB,LP,ROWUSD),(EPS,PL,COLUSD),(NICID,UP)
     1,(SCAN,TL,NP)
```

```
      INTEGER PEND/'PEND'/,EXIT/'EXIT'/,IWRD
      TL=9999999.00
    4 N=0
      READ(5,400)NODES,NARCS
  400 FORMAT(2I10)
C
C     INITIALIZATION OF THE DISTANCE MATRIX
C
      DO 1 L=1,NODES
      DO 1 M=1,NODES
      D(L,M)=9999999.99
      IF(L.EQ.M)D(L,M)=0.0

    1 CONTINUE
C
C     DATA READ IN
C
    2 READ(5,3)IWRD,I,J,B
    3 FORMAT(A4,I6,I10,F10.2)
      IF(IWRD.EQ.PEND)GO TO 10
      D(I,J)=B
      IF(J.GT.N)N=J
      IF(I.GT.J)N=I
      GO TO 2
   10 WRITE(6,700)

  700 FORMAT('1',5X,' ** ORIGINAL MATRIX **')
      CALL PRNTRX
C
C     INITIALIZATION OF THE NODES ON THE CHAINS MATRIX
C
      DO 100 I=1,N
      DO 100 K=1,N
      NODE(I,K)=K
  100 CONTINUE
C
C     TRIPLE OPERATION TO UPDATE THE DISTANCE MATRIX
C

      DO 20 J=1,N
      DO 20 I=1,N
      DO 20 K=1,N
      IF(I.EQ.J.OR.I.EQ.K.OR.J.EQ.K)GO TO 20
      IF (D(I,J).EQ.TL.OR.D(J,K).EQ.TL)GO TO 20
      X=AMIN1(D(I,J),D(J,K))
      IF (D(I,K).EQ.TL)GO TO 19
      IF (X.LE.D(I,K))GO TO 20
   19 D(I,K)=X
C
C     UPDATING THE NODES ON THE CHAINS MATRIX
C

      NODE(I,K)=J
   20 CONTINUE
      WRITE(6,800)
  800 FORMAT('1',5X,'** ARC CAPACITY MATRIX **')
      CALL PRNTRX
      WRITE(6,900)
  900 FORMAT('1',5X,'#MATRIX REPRESENTING THE NODES ON THE MAX-CAP'
     1' PATH')
      CALL PRNTRY
  200 CONTINUE
      RETURN
      END
      SUBROUTINE KSHORT
```

```
      DIMENSION LLEN(40),LINC(50),LVAL(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      INTEGER ULEN(40),UINC(50),UVAL(50),START,VAL
      INTEGER PEND/'PEND'/,NPRO
      COMMON /BLK1/ MU,ML,LLEN,LINC,LVAL,ULEN,UINC,UVAL
      COMMON /BLK2/ K
      COMMON /BLK4/ START(41),INC(40),VAL(40)
C
C INF IS DEFINED.
C
      INF=99999999

C
C AS THE INPUT NETWORK IS READ IN, THE VARIABLES AND THE ARRAYS NEEDED
C BY DSWP AND TRACE ARE CREATED.
C
  110 FORMAT(2I10)
      J=0
      MU=0
      ML=0
      NPREV=0
      N=0
    1 READ(5,110) NODES,NARCS
      DO 30 I=1,NARCS

      READ (5,800) NB,NA,LEN
  800 FORMAT(3I10)
      IF(NA.GT.N) N=NA
      IF(NB.GT.N) N=NB
      IF(NA.EQ.NPREV)  GO TO 10
      IF(NA.EQ.NPREV+1) GO TO 3
      L1=NPREV+1
      L2=NA-1
      DO 2 L=L1,L2
      START(L)=0
      ULEN(L)=0
    2 LLEN(L)=0

    3 IF(J.EQ.0) GO TO 5
      ULEN(NPREV)=JU
      LLEN(NPREV)=JL
    5 START(NA)=J+1
      JU=0
      JL=0
      NPREV=NA
   10 J=J+1
      INC(J)=NB
      VAL(J)=LEN
      IF(NB.GT.NA) GO TO 20
      MU=MU+1

      UINC(MU)=NB
      UVAL(MU)=LEN
      JU=JU+1
      GO TO 30
   20 ML=ML+1
      LINC(ML)=NB
      LVAL(ML)=LEN
      JL=JL+1
   30 CONTINUE
      START(NPREV+1)=J+1
      ULEN(NPREV)=JU
      LLEN(NPREV)=JL
C
```

```
C THE (K,NS,IMAX) AND (NF,PMAX) DATA RECORDS ARE SUCCESSIVELY READ.
C
   40 READ (5,801) NPRO,I1,I2,I3
  801 FORMAT(A4,3I5)
      IF(NPRO.EQ.PEND) GO TO 100
      IF(I3.EQ.0) GO TO 50
      K=I1
      NS=I2
C
C THE K SHORTEST DISTINCT PATH LENGTHS FROM NODE NS TO ALL NODES OF
C THE NETWORK ARE CALCULATED.
C
      CALL DSWP(NS,I3)
      GO TO 40
C
C UP TO PMAX OF THE PATHS HAVING THE K SHORTEST PATH LENGTHS FROM NODE
C NS TO NODE NF ARE DETERMINED.
C
   50 CALL TRACE(NS,I1,I2)
      GO TO 40
  100 RETURN
      END
C
C
      SUBROUTINE DSWP(NS,IMAX)
      DIMENSION LLEN(40),LINC(50),LVAL(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      INTEGER ULEN(40),UINC(50),UVAL(50),X
      COMMON /BLK1/ MU,ML,LLEN,LINC,LVAL,ULEN,UINC,UVAL
      COMMON /BLK2/ K
      COMMON /BLK3/ X(40,5)
      N1=N-1
C
C THE INITIAL APPROXIMATION MATRIX X IS FORMED.
C
      DO 20 I=1,N
      DO 20 J=1,K
   20 X(I,J)=INF
      X(NS,1)=0
      ITNS=1
C
C THE CURRENT X IS MODIFIED THROUGH MATRIX MULTIPLICATION WITH THE
C LOWER TRIANGULAR PORTION OF THE ARC LENGTH MATRIX.
C
   30 IFIN=ML
      INDX=1
      DO 40 III=1,N1
      I=-III+N1+1
      IF(LLEN(I).EQ.0) GO TO 40
      IS=IFIN-LLEN(I)+1
      CALL XMULT(I,IS,IFIN,LINC,LVAL,INDX)
      IFIN=IS-1
   40 CONTINUE
      IF(ITNS.EQ.1) GO TO 50
C
C TEST FOR CONVERGENCE.
C
      IF(INDX.EQ.1) GO TO 100
C
C THE CURRENT X IS MODIFIED THROUGH MATRIX MULTIPLICATION WITH THE
```

```
C UPPER TRIANGULAR PORTION OF THE ARC LENGTH MATRIX.
C
   50 ITNS=ITNS+1
      IS=1
      INDX=1
      DO 60 I=2,N
      IF(ULEN(I).EQ.0) GO TO 60
      IFIN=IS+ULEN(I)-1
      CALL XMULT(I,IS,IFIN,UINC,UVAL,INDX)
      IS=IFIN+1
   60 CONTINUE
C
C TEST FOR CONVERGENCE.
C
      IF(INDX.EQ.1) GO TO 100
      ITNS=ITNS+1
C
C A TEST IS MADE TO SEE IF TOO MANY ITERATIONS HAVE BEEN PERFORMED.
C
      IF(ITNS.LT.IMAX) GO TO 30
      WRITE(6,900) IMAX
  900 FORMAT('NUMBER OF ITERATIONS EXCEEDS',I5)
      GO TO 200
C
C THE SOLUTION MATRIX X IS PRINTED OUT ON UNIT 6, TOGETHER WITH THE
C VALUES FOR K,NS AND ITNS.
C
  100 WRITE (6,901), K, NS
  901 FORMAT(1H1,10X,'K=',I3,1X,'SHORTEST PATH LENGTHS FROM NODE',I4//
     12X,'TO'/1X,'NODE'//)
      DO 130 I=1,N
  130 WRITE (6,902) I,(X(I,J),J=1,K)
  902 FORMAT(' ',I3,6X,(10I9))
      WRITE (6,903) ITNS
  903 FORMAT(//1HO,'NUMBER OF ITERATIONS REQUIRED FOR CONVERGENCE =',I5)
  200 RETURN

      END
C
C
      SUBROUTINE XMULT(I,IS,IFIN,INC,VAL,INDX)
      DIMENSION INC(50)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      INTEGER VAL(50),A(5),X
      COMMON /BLK2/ K
      COMMON /BLK3/ X(40,5)
C
C INITIALIZE  TO CURRENT K SHORTEST PATH LENGTHS FOR NODE I, IN

C STRICTLY INCREASING ORDER.
C
      DO 10 J=1,K
   10 A(J)=X(I,J)
      MAX=A(K)
C
C EACH NODE OF INC INCIDENT TO NODE I IS EXAMINED.
C
      DO 100  L=IS,IFIN
      II=INC(L)
      IV=VAL(L)
C
C TEST TO SEE WHETHER IXV IS TOO LARGE TO BE INSERTED INTO A.
```

```
C
      DO 90 M=1,K
      IX=X(II,M)
      IF(IX.GE.INF) GO TO 100
      IXV=IX+IV
      IF(IXV.GE.MAX) GO TO 100
C
C IDENTIFY THE POSITION INTO WHICH IXV CAN BE INSERTED.
C
      DO 30 JJJ=2,K
      J=-JJJ+K+2
      IF(IXV-A(J-1)) 30,90,50

   30 CONTINUE
      J=1
   50 JJ=K
   70 IF(JJ.LE.J) GO TO 80
      A(JJ)=A(JJ-1)
      JJ=JJ-1
      GO TO 70
   80 A(J)=IXV
C
C IF AN INSERTION HAS BEEN MADE IN A, SET INDX = 0.
C
      INDX=0

      MAX=A(K)
   90 CONTINUE
  100 CONTINUE
      IF(INDX.EQ.1) GO TO 120
C
C  UPDATE THE K SHORTEST PATH LENGTHS TO NODE I.
C
      DO 110 J=1,K
  110 X(I,J)=A(J)
  120 RETURN
      END
C

C
      SUBROUTINE TRACE(NS,NF,PMAX)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      INTEGER P(50),Q(50),PV(50),START,VAL,X,PMAX
      COMMON /BLK2/ K
      COMMON /BLK3/ X(40,5)
      COMMON /BLK4/ START(41),INC(40),VAL(40)
C
C INITIALIZATION PHASE.
C
      DO 10 I=1,50

      P(I)=0
      Q(I)=0
   10 PV(I)=0
      JJ=1
      IF(NS.EQ.NF)JJ=2
      NPH=0
      IF(X(NF,JJ).LT.INF) GO TO 15
      WRITE (6,909) NS,NF
  909 FORMAT(1H1,'THERE ARE NO PATHS FROM NODE',I4,'TO NODE',I4)
      GO TO 200
   15 WRITE(6,901) NS,NF
  901 FORMAT(1H1,'THE K SHORTEST PATHS FROM NODE',I3,1X, 'TO NODE',I4//
     11HO,'PATH   LENGTH   NODE SEQUENCE'//)
```

```
C
C THE JJ-TH DISTINCT PATH IS BEING EXPLORED.
C
   20 KK=1
      LAB=X(NF,JJ)
      IF(LAB.EQ.INF) GO TO 200
      LL=LAB
      P(1)=NF
   30 LAST=0
C
C NOTES INCIDENT TO NODE P(KK) ARE SCANNED.
C

   40 NT=P(KK)
      IS=START(NT)
      DO 45 ND=NT,40
      IF(START(ND+1).NE.0) GO TO 48
   45 CONTINUE
   48 IF=START(ND+1)-1
      II=IS+LAST
   50 IF(II.GT.IF) GO TO 90
      NI=INC(II)
      NV=VAL(II)
      LT=LAB-NV
C

C A TEST IS MADE TO SEE IF THE CURRENT PATH CAN BE EXTENDED BACK TO
C NODE NI.
C
      DO 60 J=1,K
      IF(X(NI,J)-LT)60,80,70
   60 CONTINUE
   70 II=II+1
      GO TO 50
   80 KK=KK+1
      IF(KK.GT.50) GO TO 190
      P(KK)=NI
      Q(KK)=II-IS+1

      PV(KK)=NV
      LAB=LT
C
C TESTS ARE MADE TO SEE IF THE CURRENT PATH CAN BE EXTENDED FURTHER.
C
      IF(LAB.NE.0) GO TO 30
      IF(NI.NE.NS) GO TO 30
C
C A COMPLETE PATH FROM NS TO NF HAS BEEN GENERATED AND IS PRINTED
C OUT ON UNIT 6.
C
      NPH=NPH+1

      WRITE (6,902) NPH,LL,(P(J),J=1,KK)
  902 FORMAT(1X,I4,I8,5X,(20I5))
      IF(NPH.GE.PMAX) GO TO 200
   90 LAST=Q(KK)
      P(KK)=0
      LAB=LAB+PV(KK)
      KK=KK-1
      IF(KK.GT.0) GO TO 40
C
C THE EXPLORATION OF THE CURRENT JJ=TH DISTINCT PATH LENGTH IS ENDED.
C
      JJ=JJ+1
      IF(JJ.GT.K) GO TO 200
```

```
      GO TO 20
  190 WRITE(6,903)
  903 FORMAT(1HO,'NUMBER OF ARCS IN PATH EXCEEDS 50')
  200 RETURN
      END
C
      SUBROUTINE GENPAK
      DIMENSION IARC(500),JARC(500),COST(500),FLOW(500)
     1,AMP(500),LOWER(500),UPPER(500),TITLE(20)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT

      COMMON /G2/ COST,FLOW,AMP
      COMMON /G3/ LOWER,UPPER,BIG
      COMMON /G5/ FLONET,OUTFLO,TOTCST,CSTNOW
      COMMON /G6/ TITLE,NDEG,NLOP,ITER,SINK
      COMMON /G7/ SOURCE,EPS,SICH
      INTEGER  TITLE,SINK,SOURCE,FINISH
      INTEGER PEND/'PEND'/,NPRO
      REAL LOWER
      DATA FINISH/4H    /
COMPUTE THE MACHINE EPSILON.
      EPS = 1.0
    5 EPS = EPS/2.0/

      TOL1 = 1.0   + EPS
      IF (TOL1 .GT. 1.0  ) GO TO 5
      EPS=SQRT(EPS)
COMPUTE THE MACHINE INFINITY.
      BIG=1.E+6
    8 BIG=BIG*BIG
      BIG1=1.+BIG
      IF(BIG1 .GT. BIG) GO TO 8
      BIG=BIG*BIG
C
      WRITE(6,195)
      READ(5,90) NODES,NARCS

      WRITE(6,130)
      WRITE(6,140)
      WRITE(6,150)
C   READ THE ARCS AND INITIALIZE THEIR FLOWS.
      DO 10 I=1,NARCS
      READ (5,160) IARC(I),JARC(I),LOWER(I),UPPER(I),COST(I),AMP(I)
      FLOW(I)=0.
      WRITE(6,170) I,IARC(I),JARC(I),LOWER(I),UPPER(I),COST(I),AMP(I)
 10   CONTINUE
      READ(5,110) SOURCE,SINK,IPRINT,OUTFLO
      WRITE(6,100) SOURCE,SINK
      WRITE(6,120) OUTFLO

   20 NDEG=0
      NLOP=0
      ITER=0
      FLONET=0.0
      TOTCST=0.0
      IFS=0
      NARC = NARCS + 1
CREATE A DUMMY ARC TO PROVIDE FEASIBLE OUTPUT FLOW IN
CASE THE DESIRED OUTPUT FLOW IS NOT FEASIBLE.
      IARC(NARC)=SOURCE
      JARC(NARC)=SINK
      LOWER(NARC)=0.
      UPPER(NARC)=BIG
```

```
      GOST(NARC)=BIG
      AMP(NARC)=1.
      FLOW(NARC)=0.
C
C   DEFINE THE INVERSE NETWORK (OR THE MIRROR ARCS)
C
      DO 30 I=1,NARC
      NN = NARC + I
      IARC(NN) = JARC(I)
      JARC(NN) = IARC(I)
      AMP(NN)=1./AMP(I)
      LOWER(NN)=LOWER(I)*AMP(I)

      UPPER(NN)=UPPER(I)*AMP(I)
      COST(NN)=-COST(I)/AMP(I)
      FLOW(NN)=0.
 30   CONTINUE
      IF(IPRINT .GT. 0) WRITE(6,200)
 40   CALL SHORT (IENTER,ILEAV)
        IF (IENTER.EQ.0) GO TO 60
      CALL MAXFLW
      TOTCST=TOTCST+CSTNOW
      ITER=ITER+1
        IF (IPRINT.EQ.0) GO TO 50
      WRITE(6,180) ITER,FLONET,TOTCST

       WRITE(6,190) ILEAV,IENTER,SICH
 50   CONTINUE
        IF (ABS(FLONET-OUTFLO) .LE. EPS) GO TO 60
        GO TO 40
 60   CONTINUE
      CALL PROUT(IENTER)
 90   FORMAT(2I10)
 110  FORMAT(3I5,F10.2)
 100  FORMAT (' SOURCE NODE=',I5,5X,'SINK NODE =',I5,///)
 120  FORMAT (' OUTPUT FLOW REQUIREMENT',F10.2,///)
 130  FORMAT (21H *****INPUT DATA*****,//)
 140  FORMAT ('          ARC      START       END      LOWER      UPPER
     1 COST      AMPLIFICATION')
 150  FORMAT ('          NO.       NODE       NODE      BOUND      BOUND',//)
 160  FORMAT(2I10,4F10.2)
 170  FORMAT (3I10,6F11.2)
 180  FORMAT (' ITERATION ',15,5X,' FLOW = ',F10.2,5X,' GOST = ',F10.2)
  195 FORMAT(1H1,///,5X,'GENERALIZED NETWORK MINIMIZATION PROBLEM',//
     1//)
 190  FORMAT(15X,' REMOVE',I4,5X,'ENTER',I4,5X,'DELTA =',F10.3,//)
  200 FORMAT(1H1,'*****  INTERMEDIATE RESULTS  *****',///)
      RETURN
      END
C

      SUBROUTINE SHORT (IENTER,ILEAV)
C     SUBROUTINE TO FIND THE INITIAL AND SUBSEQUENT FLOW AUGMENTING TREE
      DIMENSION DISSET(300),BARC(300),FARC(300),RARC(300),ICHK(300)
     1 ,GAN(300),V(300),LOWER(500),UPPER(500)
     2  ,IARC(500),JARC(500),COST(500),FLOW(500),AMP(500)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
      COMMON /G2/ COST,FLOW,AMP
      COMMON /G3/ LOWER,UPPER,BIG
      COMMON /G5/ FLONET,OUTFLO,TOTCST,CSTNOW
      COMMON /G7/ SOURCE,EPS,SICH

      COMMON /G8/ BARC,FARC,RARC,GAN,V,ICHK
```

```
      INTEGER DISSET,BARC,FARC,RARC,SOURCE
      REAL LOWER
         IF (IFS.NE.0) GO TO 150
      SET UP POINTERS TO FIND INITIAL TREE.
      DO 10 I=1,NODES
      BARC(I)=0
      FARC(I)=0
      RARC(I)=0
      DISSET(I)=1
      GAN(I)=1.
      V(I) =BIG

   10 CONTINUE
      V(SOURCE)=0.
      ICHANG=0
      IENTER=1
      ITF=0
      IFS=1
C     SET UP SHORTEST PATH TREE FOR FIRST ITERATION.
   20 CONTINUE
      DO 70 I=1,NARC
      JJ=JARC(I)
      II=IARC(I)
         IF( (LOWER(I)-FLOW(I)) .GT. EPS ) GO TO 40

      POT=(V(II)+COST(I)/AMP(I)
      GO TO 50
   40 POT=(V(II)-BIG)/AMP(I)
   50 CONTINUE
         IF ((V(JJ)-POT) .LT. EPS) GO TO 70
      V(JJ)=POT
      BARC(JJ)=I
         IF (ITF.EQ.0) GO TO 60
      CALL LOOP (I,JJ)
   60 CONTINUE
      ICHANG=1
   70 CONTINUE

         IF (ICHANG.EQ.0) GO TO 80
      ICHANG=0
      ITF=1
         GO TO 20
   80 CONTINUE
C     CALCULATE FORWARD POINTERS FOR FIRST ITERATION.
      DO 120 I=1,NODES
         IF (DISSET(I).EQ.0) GO TO 120
      KK=BARC(I)
         IF (KK.EQ.0) GO TO 120
      LL=IARC(KK)
         IF (FARC(LL).NE.0) GO TO 90

      FARC(LL)=KK
         GO TO 120
   90 CONTINUE
      MM=FARC(LL)
  100 CONTINUE
      MN=JARC(MM)
         IF (RARC(MN).NE.0) GO TO 110
      RARC(MN)=KK
         GO TO 120
  110 CONTINUE
      MM=RARC(MN)
         GO TO 100

  120 CONTINUE
  130 CONTINUE
```

```
      IF ( IPRINT .LT. 2) RETURN
      WRITE(6,320) (BARC(I),I=1,NODES)
      WRITE(6,330) (FARC(I),I=1,NODES)
      WRITE(6,340) (RARC(I),I=1,NODES)
      WRITE(6,300) (DISSET(I),I=1,NODES)
      WRITE(6,290) (V(I),I=1,NODES)
      WRITE(6,310) (GAN(I),I=1,NODES)
      RETURN
C     FIND NEW FLOW AUGMENTING TREE AFTER THE FIRST ITERATION.
  150 CONTINUE

      DO 160 I=1,NODES
      DISSET(I)=0
  160 CONTINUE
C     DELETE BRANCH FROM BASIS AFTER THE FIRST ITERATION.
      II=IROOT
      I=BARC(II)
      ILEAV=I
      IA=IARC(I)
      CALL DESUB (I,IA)
      RARC(II)=0
      BARC(II)=0
C     SET NEW NODE GAINS ON DISCONNECTED NODES.

  170 CONTINUE
      I=FARC(II)
         IF (I.EQ.0) GO TO 180
      JJ=JARC(I)
      GAN(JJ)=GAN(II)/AMP(I)
      II=JJ
         IF (II.EQ.IROOT) GO TO 180
         GO TO 170
  180 CONTINUE
      J=RARC(II)
      DISSET(II)=1
         IF (J.EQ.0) GO TO 190

      II=JARC(J)
      K=BARC(II)
      JJ=JARC(K)
      GAN(II)=GAN(JJ)/AMP(K)
         GO TO 170
  190 CONTINUE
      K=BARC(II)
         IF (II.EQ.IROOT) GO TO 200
      II=IARC(K)
         GO TO 180
  200 CONTINUE
C     DETERMINE THE NEW BRANCH TO ENTER THE BASIS.

      IENTER=0
      SICH=BIG
      DO 250 K=1,NARC
         IF ((UPPER(K)-FLOW(K)).GT.EPS) GO TO 210
C     LOOKING AT A BACKWARD BRANCH
         IF((FLOW(K)-LOWER(K)) .LT. EPS) GO TO 250
      JJ=IARC(K)
         IF (DISSET(JJ).EQ.0) GO TO 250
      II=JARC(K)
      I=K+NARC
         IF (DISSET(II).EQ.0) GO TO 240
C     NEW BRANCH FORMS A LOOP

         GO TO 220
  210 CONTINUE
      T=K
```

```
      JJ=JARC(I)
         IF (DISSET(JJ).EQ.0) GO TO 250
      II IARC(I)
         IF (DISSET(II).EQ.0) GO TO 240
C     NEW BRANCH FORMS A LOOP.
  220 CONTINUE
      GALPIV=GAN(II)/(GAN(JJ)*AMP(I))
      IF ( GALPIV .GE. (1.-EPS)) GO TO 250
      POTCH=(((V(II)+COSTF(I))/AMP(I))-V(JJ))/((1.-GALPIV)*GAN(JJ))
  230 CONTINUE
         IF (POTCH.GE.SICH) GO TO 250
      SICH=POTCH
      IENTER=I
         GO TO 250
C     NEW BRANCH DOES NOT FORM A LOOP.
  240 CONTINUE
      POTCH=(((V(II)+COSTF(I))/AMP(I))-V(JJ))/GAN(JJ)
         GO TO 230
  250 CONTINUE
         IF (IENTER.EQ.0) GO TO 270
C     CHANGE NODE LABELS AND POINTERS TO REFLECT ENTERING BRANCH.
C     CHANGE POINTERS.
      CALL TRECHG (IENTER, ILEAV)
C     CHANGE NODE POTENTIALS.
      DO 260 II=1,NODES
         IF (DISSET(II).EQ.0) GO TO 260
      V(II)=SICH*GAN(II)+V(II)
  260 CONTINUE
         GO TO 130
  270 CONTINUE
      IF (IPRINT .NE. 0) WRITE(6,280) FLONET
                         II=IARC(ILEAV)
      CALL ADSUB (ILEAV,II)
                      JJ=JARC(ILEAV)
      BARC(JJ)=ILEAV
         GO TO 130
  280 FORMAT (' MAXIMUM FLOW FOUND ',F20.10)
  290 FORMAT ( ' V       ',(21(F5.2,1X)))
  300 FORMAT ( ' DISSET  ',(21I5))
  310 FORMAT ( ' GAIN    ',(21(F5.2,1X)))
  320 FORMAT ( ' BARC    ',(21I5))
  330 FORMAT ( ' FARC    ',(21I5))
  340 FORMAT ( ' RARC    ',(21I5))
      END
C
      SUBROUTINE TRECHG (IENTER,ILEAV)
C     SUBROUTINE TO FORM THE FLOW AUGMENTING TREE BY DELETING ONE ARC
C     FROM THE TREE AND INSERTING A NEW ARC INTO THE TREE.
      DIMENSION IARC(500),JARC(500),BARC(300),FARC(300),RARC(300)
     1,GAN(300) , V(300),ICHK(300)
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
      COMMON /G8/ BARC,FARC,RARC,GAN,V,ICHK
      INTEGER   BARC,   RARC
      IROOT=JARC(ILEAV)
      NLIS=0
      JJ=JARC(IENTER)
      DELETE PATH FROM JARC(IENTER) TO IROOT

   10 CONTINUE
      JB=BARC(JJ)
         IF (JJ.EQ.IROOT) GO TO 20
      NLIS=NLIS+1
```

```
      II = IARC(JB)
      CALL DESUB (JB,II)
      IF(JB.LE.NARC) I=JB+NARC
      IF(JB.GT.NARC) I=JB-NARC
      ICHK(NLIS)=I
      BARC(JJ)=0
      RARC(JJ)=0
      JJ=II
         GO TO 10
 20   CONTINUE
C     ADD IN THE REVERSE OF THE PATH JUST DELETED.
         IF (NLIS.EQ.0) GO TO 30
      I=ICHK(NLIS)
      NLIS=NLIS-1
      II=IARC(I)
      JJ=JARC(I)
      CALL ADSUB (I,II)
      BARC(JJ)=I
         GO TO 20
 30   CONTINUE

      II = IARC(IENTER)
      CALL ADSUB (IENTER,II)
      JJ=JARC(IENTER)
      BARC(JJ)=IENTER
      RARC(JJ)=0
      RETURN
      END
C
      SUBROUTINE FLCHG (II,FLONOW)
C     SUBROUTINE TO INCREASE THE FLOW IN AN ARC BY A GIVEN AMOUNT.
      DIMENSION IARC(500),JARC(500),COST(500),FLOW(500),AMP(500)
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT

      COMMON /G2/ COST,FLOW,AMP
      COMMON /G5/ FLONET,OUTFLO,TOTCST,CSTNOW
      IF(II.GT.NARC) GO TO 10
C     CHANGE FLOW IN A FORWARD ARC.
      FLOW(II)=FLOW(II)+FLONOW
      CSTNOW=CSTNOW+FLONOW*COST(II)
         GO TO 20
C     CHANGE FLOW IN A MARROR ARC.
 10   CONTINUE
      KK=II-NARC
      FLOW(KK)=FLOW(KK)-FLONOW/AMP(KK)
      CSTNOW=CSTNOW-FLONOW*COST(KK)/AMP(KK)

 20   CONTINUE
      RETURN
      END
C
      SUBROUTINE FLOP (JJ,FLMAX,GN)
C     SUBROUTINE TO DETERMINE THE GAIN, MAXIMUM FLOW CHANGE, AND ROOT OF
C     A FLOW GENERATING CYCLE IN THE FLOW AUGMENTING TREE.
      DIMENSION IARC(500),JARC(500),BARC(300),FARC(300),RARC(300)
     1,GAN(300) , V(300),ICHK(300) ,COST(500),FLOW(500),AMP(500)
     2,LOWER(500),UPPER(500)
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
      COMMON /G2/ COST,FLOW,AMP

      COMMON /G3/ LOWER,UPPER,BIG
      COMMON /G8/ BARC,FARC,RARC,GAN,V,ICHK
      INTEGER   BARC
      FLMAX=BIG
      GN=1.
```

```
      IJ=JJ
 10   CONTINUE
      IJK=BARC(IJ)
      GN=GN*AMP(IJK)
      FLMXT=FLMXC(IJK)*GN
         IF (FLMXT.GT.FLMAX) GO TO 20
      FLMAX=FLMXT

      IROOT=JARC(IJK)
 20   CONTINUE
       IF (IARC(IJK) .EQ. JJ ) GO TO 30
      IJ=IARC(IJK)
      GAN(IJ)=GAN(JJ)*GN
         GO TO 10
 30   CONTINUE
      FLMAX=FLMAX*(1.-1./GN)
      RETURN
      END
C
      SUBROUTINE ADSUB (I,II)

C     SUBROUTINE TO ADD AN ARC TO THE TRIPLE LABEL REPRESENTATION OF THE
C     FLOW AUGMENTING TREE.
      DIMENSION IARC(500),JARC(500),BARC(300),FARC(300),RARC(300)
     1,GAN(300) , V(300),ICHK(300)
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
      COMMON /G8/ BARC,FARC,RARC,GAN,V,ICHK
      INTEGER    FARC, RARC
C     ADDS ARC I TO THE LIST OF SUBSEQUENT ARCS TO NODE II.
         IF (FARC(II).NE.0) GO TO 10
      FARC(II)=I
         GO TO 40
 10   CONTINUE

      MM=FARC(II)
 20   CONTINUE
      MN=JARC(MM)
         IF (RARC(MN(.NE.0) GO TO 30
      RARC(MN)=I
         GO TO 40
 30   CONTINUE
      MM=RARC(MN)
      GO TO 20
 40   CONTINUE
      RETURN
      END

C
      SUBROUTINE DESUB (I,II)
C     SUBROUTINE TO DELETE AN ARC FROM THE TRIPLE LABEL REPRESENTATION
C     OF THE FLOW AUGMENTING TREE.
      DIMENSION IARC(500),JARC(500),BARC(300),FARC(300),RARC(300)
     1,GAN(300) , V(300),ICHK(300)
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
      COMMON /G8/ BARC,FARC,RARC,GAN,V,ICHK
      INTEGER    FARC, RARC
C     DELETES ARC I FROM THE LIST OF SUBSEQUENT ARCS TO NODE II.
      JJ = JARC(I)
         IF (FARC(II).NE.I) GO TO 10

      FARC(II)=RARC(JJ)
      RETURN
 10   CONTINUE
      MM=FARC(II)
 20   CONTINUE
      MN=JARC(MM)
```

```
          IF (RARC(MN).NE.I) GO TO 30
         RARC(MN)=RARC(JJ)
         RETURN
   30    CONTINUE
         MM=RARC(MN)
            GO TO 20
         END
         FUNCTION FLMXC (I)
C        FUNCTION TO DETERMINE MAXIMUM FLOW CHANGE IN AN ARC
         DIMENSION IARC(500),JARC(500),BARC(300),FARC(300),RARC(300)
        1 ,V(300),ICHK(300),COST(500),FLOW(500),AMP(500),LOWER(500)
        2 ,GAN(300), UPPER(500)

         COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
         COMMON /G2/ COST,FLOW,AMP
         COMMON /G3/ LOWER,UPPER,BIG
         COMMON /G7/ SOURCE,EPS,SICH
         COMMON /G8/ BARC,FARC,RARC,GAN,V,ICHK
         REAL      LOWER
         II=IARC(I)
         JJ=JARC(I)
         IF(I.GT.NARC) GO TO 80
            IF (FLOW(I).GE.UPPER(I)) GO TO 110
            IF (FLOW(I).LT.LOWER(I)) GO TO 70
         FLMXC=UPPER(I)-FLOW(I)

         RETURN
   70    CONTINUE
         FLMXC=LOWER(I)-FLOW(I)
         RETURN
   80    CONTINUE
         K=I-NARC
            IF (FLOW(K).GT.UPPER(K)) GO TO 100
            IF (FLOW(K).LE.LOWER(K)) GO TO 110
         FLMXC=(FLOW(K)-LOWER(K))*AMP(K)
         RETURN
  100    CONTINUE
         FLMXC=(FLOW(K)-UPPER(K))*AMP(K)

         RETURN
  110    CONTINUE
         FLMXC=0.0
         RETURN
         END
         FUNCTION COSTF(I)
C    FUNCTION TO CALCULATE THE COSTS ON AN ARC CONSIDERING
C    UPPER AND LOWER BOUNDS.
         DIMENSION IARC(500),JARC(500),COST(500),FLOW(500)
        1,AMP(500),LOWER(500),UPPER(500)
         COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
         COMMON /G2/ COST,FLOW,AMP

         COMMON /G3/ LOWER,UPPER,BIG
         REAL LOWER
         IF(I .GT. NARC) GO TO 50
         IF(FLOW(I)-LOWER(I)) 10,30,30
    10   COSTF=-BIG
         RETURN
    20   COSTF=COST(I)
         RETURN
    30   IF(UPPER(I)-FLOW(I)) 40,40,20
    40   COSTF=BIG
         RETURN
    50   K=I-NARC

        IF(FLOW(K)-LOWER(K)) 80,80,70
```

```
   60 COSTF=-COST(K)/AMP(K)
      RETURN
   70 IF(UPPER(K)-FLOW(K)) 90,60,60
   80 COSTF=BIG/AMP(K)
      RETURN
   90 COSTF=-BIG/AMP(K)
      RETURN
      END
C
C
      SUBROUTINE LOOP (I,JJ)
C      SUBROUTINE TO DETERMINE IF THE FLOW AUGMENTING TREE INCLUDES A
C     FLOW GENERATING CYCLE.  IF SO NODE POTENTIALS ARE ADJUSTED
C     ACCORDINGLY. USED ONLY IN THE FIRST ITERATION.
      DIMENSION IARC(500),JARC(500),BARC(300),FARC(300),RARC(300)
     1,GAN(300) , V(300),ICHK(300) ,COST(500),FLOW(500),AMP(500)
      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
      COMMON /G2/ COST,FLOW,AMP
      COMMON /G8/ BARC,FARC,RARC,GAN,V,ICHK
      INTEGER   BARC, FARC, RARC
      DETERMINE IF POINTERS INDICATE A LOOP.

      DO 10 K=1,NODES
      ICHK(K)=0
  10  CONTINUE
      ICHK(JJ)=1
      IJ=JJ
  20  CONTINUE
      IJK=BARC(IJ)
      IF (IJK.EQ.0) RETURN
      IA=IARC(IJK)
         IF (IA.EQ.JJ) GO TO 30
      IF (ICHK(IA).EQ.1) RETURN
      ICHK(IA)=1

      IJ=IA
         GO TO 20
  30  CONTINUE
      TEMP=0.
      GN=1.
      IJ=JJ
C     CALCULATE THE COST TO OBTAIN ONE UNIT OF FLOW INTO JJ
  40  CONTINUE
      IJK=BARC(IJ)
      GN=GN*AMP(IJK)
      TEMP = TEMP + COSTF(IJK) / GN
      IA=IARC(IJK)

         IF (IA.EQ.JJ) GO TO 50
      IJ=IA
         GO TO 40
  50  CONTINUE
C     CALCULATE COST TO OBTAIN ONE UNIT OF FLOW OUT OF LOOP AT JJ
      V(JJ)=TEMP/(1.-1./GN)
  60  CONTINUE
      IJ=IA
      IJK=BARC(IJ)
      IA=IARC(IJK)
      IF ( IA. EQ. JJ) GO TO 70
      V(IA) = V(IJ) * AMP(IJK) - COSTF(IJK)

         GO TO 60
   70 CONTINUE
```

```
      RETURN
      END
C
      SUBROUTINE MAXFLW
C     SUBROUTINE TO CALCULATE THE MAXIMUM FLOW INCREASE INTO THE SINK.
C     ARC TO LEAVE THE TREE IS ALSO DETERMINED.  THE FLOW IS CHANGED IN
C     THE AUGMENTING PATH.
      DIMENSION IARC(500),JARC(500),BARC(300),FARC(300),RARC(300)
     1,GAN(300) , V(300),ICHK(300) ,TITLE(20)
     2,COST(500),FLOW(500),AMP(500),LOWER(500),UPPER(500)

      COMMON S(50,50),D(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     1UP(50,50),NP(50),INFSTC(50,3),NODES,INF,LEVEL,M,N,MAX
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
      COMMON /G2/ COST,FLOW,AMP
      COMMON /G3/ LOWER,UPPER,BIG
      COMMON /G5/ FLONET,OUTFLO,TOTCST,CSTNOW
      COMMON /G6/ TITLE,NDEG,NLOP,ITER,SINK
      COMMON /G7/ SOURCE,EPS,SICH
      COMMON /G8/ BARC,FARC,RARC,GAN,V,ICHK
      INTEGER   SOURCE, SINK, BARC, TITLE
      FIND OUT IF THERE IS A LOOP. IF SO JJ IS THE JUNCTION OF THE LOOP.
      DO 10 I=1,NODES

      ICHK(I)=0
 10   CONTINUE
      JJ=SOURCE
      I=SINK
 20   CONTINUE
      ICHK(I)=1
         IF (I.EQ.SOURCE) GO TO 40
      II=BARC(I)
         IF (II.EQ.0) GO TO 130
      I=IARC(II)
         IF (ICHK(I).EQ.1) GO TO 30
         GO TO 20

 30   CONTINUE
      JJ=I
C    INCREASE THE NUMBER OF LOOP ITERATIONS.
      NLOP=NLOP+1
C     FIND MAXIMUM FLOW CHANGE POSSIBLE.
 40   CONTINUE
      FLMX=BIG
      GN=1.
      I=SINK
      GAN(I)=1.
 50   CONTINUE
         IF (I.EQ.JJ) GO TO 60

      KK=BARC(I)
      GN=GN*AMP(KK)
      FLMXT=FLMXC(KK)*GN
      I=IARC(KK)
      GAN(I)=GN
         IF (FLMXT.GT.FLMX) GO TO 50
      FLMX=FLMXT
      IROOT=JARC(KK)
         GO TO 50
 60   CONTINUE
         IF (JJ.EQ.SOURCE) GO TO 70
      CALL FLOP (JJ,FLMAX,GLOOP)

      FLMXT=FLMAX*GN
         IF (FLMXT.GT.FLMX) GO TO 70
      FLMX=FLMXT
```

```
 70   CONTINUE
C     INCREASE TOTAL FLOW BY THE MAXIMUM FLOW CHANGE.
      FLO=OUTFLO-FLONET
      IF (FLO.GT.FLMX) FLO=FLMX
      CSTNOW=0.
      IF (FLO .GT. EPS) GO TO 75
C   INCREASE THE NUMBER OF DEGENERATE ITERATIONS.
      NDEG=NDEG+1
      GO TO 120

   75 FLONET=FLONET+FLO
C     CALCULATE FLOW CHANGE ON EACH ARC.
      I=SINK
 80   CONTINUE
         IF (I.EQ.JJ) GO TO 90
      II=BARC(I)
         IF (II.EQ.0) GO TO 130
      I=IARC(II)
      FLONOW=FLO/GAN(I)
      CALL FLCHG (II,FLONOW)
         GO TO 80
 90   CONTINUE

         IF (JJ.EQ.SOURCE) GO TO 120
      FLOOP=FLO/(GAN(I)*(1-(1/GLOOP)))
      I=JJ
      FLGA=FLOOP*GAN(JJ)
 100  CONTINUE
      II=BARC(I)
      I=IARC(II)
      FLONOW=FLGA/GAN(I)
         IF (I.NE.JJ) GO TO 110
      J=JARC(II)
      FLONOW=FLGA/(GAN(J)*AMP(II))
 110  CONTINUE

      CALL FLCHG (II,FLONOW)
         IF (I.EQ.JJ) GO TO 120
         GO TO 100
 120  CONTINUE
      RETURN
 130  CONTINUE
      WRITE(6,140) FLONET,TOTCST
      CALL EXIT
      RETURN
 140  FORMAT (///,' PROBLEM IS INFEASIBLE. THE MAXIMUM FLOW IS  ',F10.2 ,
     1'   AT A TOTAL COST OF  ',F10.2)
      END

C
      SUBROUTINE PROUT(IENTER)
C SUBROUTINE TO PRINT OUT THE OPTIMAL SOLUTION.
      DIMENSION IARC(500),JARC(500),BARC(300),FARC(300),RARC(300)
     1  ,ICHK(300),COST(500),FLOW(500),AMP(500),LOWER(500)
     2 ,GAN(300),V(300),UPPER(500),TITLE(20)
      COMMON /G1/ IARC,JARC,NARC,IFS,IROOT,IPRINT
      COMMON /G2/ COST,FLOW,AMP
      COMMON /G3/ LOWER,UPPER,BIG
      COMMON /G5/ FLONET,OUTFLO,TOTCST,CSTNOW
      COMMON /G6/ TITLE,NDEG,NLOP,ITER,SINK
      COMMON /G8/ BARC,FARC,RARC,GAN,V,ICHK

      INTEGER TITLE
      REAL LOWER
      WRITE(6,800)
      IF(FLOW(NARC)) 30,30,10
```

```
   10 WRITE(6,110)
      WRITE(6,120)
      NARC=NARC-1
      GO 20 I=1,NARC
      IF(FLOW(I) .LT. UPPER(I)) GO TO 20
      WRITE(6,130) IARC(I),JARC(I)
   20 CONTINUE
      WRITE(6,170)

      GO TO 60
   30 DO 35 I=1,NARC
      IF(FLOW(I) .GE. LOWER(I)) GO TO 35
      IENTER=1
      GO TO 38
   35 CONTINUE
      IF(IENTER .NE. 0 .AND. FLONET .EQ. OUTFLO) GO TO 50
   38 WRITE(6,110)
      WRITE(6,140)
      NARC=NARC-1
      DO 40 I=1,NARC
      IF(FLOW(I) .GE. LOWER(I)) GO TO 40

      WRITE(6,130) IARC(I),JARC(I)
   40 CONTINUE
      WRITE(6,170)
      GO TO 60
   50 NARC=NARC-1
      WRITE(6,160)
   60 WRITE(6,400)
      DO 70 I=1,NARC
      ACOST = COST(I)*FLOW(I)
      WRITE(6,500)I,IARC(I),JARC(I),LOWER(I),UPPER(I),COST(I),AMP(I)
     1        , FLOW(I), ACOST
   70 CONTINUE

      WRITE(6,600) FLONET,TOTCST
      WRITE(6,700) ITER,NDEG,NLOP
      WRITE(6,800)
      RETURN
  110 FORMAT(///,'********** THE OUTPUT FLOW REQUIREMENT IS NOT FEASIBLE'
     1 , '**********',//)
  120 FORMAT(/,5X,' THE FOLLOWING ARCS ARE SATURATED :')
  130 FORMAT(10X,I3,2X,I3)
  140 FORMAT(/,5X,'THE LOWER BOUNDS CONSTRAINTS ARE VIOLATED FOR THE'
     1 ,'FOLLOWING ARCS :')
  160 FORMAT(///, '********** OPTIMAL FLOW PATTERN **********',//)
  170 FORMAT(///,'********** FLOW PATTERN OBTAINED AT TERMINATION **'
     1 ,'**********',//)
  400 FORMAT(/,'   ARC    START END     LOWER      UPPER       COST
     1,'GAIN       FLOW          ARC COST',/)
  500 FORMAT (I5,2X,2I5,5F10.2,F15.2)
  600 FORMAT(///,' TOTAL OUTPUT FLOW = ',F20.4,/,' TOTAL COST       =
     1      F20.4,/)
  700 FORMAT (' NUMBER OF ITERATIONS ',I10,/,
     1        ' NUMBER OF DEGENERATE ITERATIONS ',I10,/,
     2        ' NUMBER OF LOOP ITERATIONS ',I10,/)
  800 FORMAT(1H1)
      END
```

ОГЛАВЛЕНИЕ