

А. А. ГРЕШИЛОВ

Прикладные задачи
математического
программирования

ИЗДАТЕЛЬСТВО МГТУ

А. А. ГРЕШИЛОВ

ПРИКЛАДНЫЕ ЗАДАЧИ
МАТЕМАТИЧЕСКОГО
ПРОГРАММИРОВАНИЯ

Допущено Государственным комитетом СССР по народному образованию в качестве учебного пособия для студентов высших технических учебных заведений

ИЗДАТЕЛЬСТВО МГТУ
1990

ББК 22.18
Г81

Грешилов А. А.

Прикладные задачи математического программирования:
Учебное пособие. — М.: Изд-во МГТУ, 1990. — 189 с., ил.

ISBN 5-7038-0103-6

Рассматривается в популярной форме широкий круг задач математического программирования, возникающих в повседневной жизни (формирование семейного бюджета, организация досуга, составление диет, покупка автомобиля и т. д.), при разработке занимательных игр на компьютерах («военные» задачи), а также в производственной деятельности. Излагаются особенности этих задач и методы их решения, подробно описываются алгоритмы решения каждой задачи.

Для студентов технических и экономических специальностей вузов, изучающих методы оптимизации, исследования операций и системного анализа.

Рис. 70. Табл. 48. Библиогр. 22 назв.

Рецензенты:

канд. техн. наук Л. В. Маркин, канд. физ.-мат. наук А. В. Тимохов

ББК 22.18

Анатолий Антонович Грешилов

ПРИКЛАДНЫЕ ЗАДАЧИ МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Редактор *С. Н. Удалова*

Художественный редактор *Л. М. Толмачева*

Технический редактор *О. В. Рыбина*

Корректор *Л. И. Малютина*

Подписано в печать 26.11.90. Формат 60×90¹/₁₆. Бумага тип. № 2. Усл. печ. л. 12.
Уч.-изд. л. 11,34. Тираж 4200 экз. Заказ 1290. Цена 40 коп. Изд. № 242.

Издательство МГТУ

107005, Москва, Б-5, 2-я Бауманская, 5.

Типография Минстанкопрома СССР

Москва, 142002, г. Щербинка, ул. Типографская, д. 10

ISBN 5-7038-0103-6

© А. А. Грешилов, 1990.

Г $\frac{1602110000-242}{095(02)-90}$ 3-90

ВВЕДЕНИЕ

Развитие вычислительной техники и широкое распространение персональных компьютеров дает возможность многие решения принимать не «на веру», а построив соответствующую модель явления, на основе соответствующих расчетов. В свою очередь, такой подход требует алгоритмов и методов решения производственных и повседневных задач, а также алгоритмов составления занимательных игр, доступных самому широкому кругу пользователей.

Как правило, в задачах можно выделить цель, во имя которой их решают, и указать условия, т. е. ограничения, при которых они должны быть решены. Такими задачами занимается раздел математики — математическое программирование. По математическому программированию существует достаточно много литературы, однако все книги часто ориентированы на математиков, в крайнем случае на студентов экономических вузов, изучивших дифференциальное и интегральное исчисления. Книги, которая была бы доступна инженеру, владеющему математикой в объеме технического вуза, нет. Восполнить этот пробел предназначена данная книга. Она рассчитана на самый широкий круг пользователей персональных компьютеров и микрокалькуляторов, на любителей занимательных задач по математике. Для понимания всех разделов книги достаточно знаний математики в объеме первых двух курсов вузов, а для понимания алгоритмов решения задач и этого не требуется. По ходу изложения теоретических основ решения задач математического программирования даны определения ряда понятий, необходимых для понимания излагаемых алгоритмов: квадратичные формы, скалярное произведение векторов, частные производные, градиент, производная по направлению, линейная зависимость векторов и т. п.

В гл. 1 рассказано об истории появления математической дисциплины — математического программирования, а также обсуждены основные особенности задач на условный экстремум, решаемых в этом разделе математики. Приведены простейшая оптимизационная задача и графические методы решения задач математического программирования.

В гл. 2 рассмотрен наиболее простой и изученный раздел математического программирования — линейное программирование.

Здесь же рассмотрены и частные случаи линейного программирования — транспортные задачи и задачи целочисленного линейного программирования. Приведены примеры решения задач линейного программирования из различных областей знания.

В гл. 3 описаны сетевые задачи. На конкретных примерах рассмотрены сетевые задачи о минимальной стоимости (о кратчайшей сети) и о максимальном потоке, показаны их особенности и преимущества по сравнению с задачами линейного программирования.

В гл. 4 рассмотрены задачи, решаемые с помощью динамического программирования. Показана связь динамического программирования с другими разделами математического программирования.

В гл. 5 рассказано о некоторых направлениях развития методов решения задач математического программирования: о параметрическом программировании; о решении многопродуктовых сетевых задач; об одном из эвристических методов решения сетевых задач — о методе штрафных (барьерных) функций.

В книге не излагаются численные методы безусловной и условной оптимизации, применяемые в математическом программировании, несмотря на большую их важность в изучении этой математической дисциплины. Читатель может самостоятельно познакомиться с численными методами, поскольку они хорошо описаны в литературе.

В задачах математического программирования всегда рассматривают функции многих переменных. Поэтому решение задачи об экстремуме имеет несколько координат, а рассчитываемые величины являются векторами. Как правило, векторный смысл рассматриваемых величин ясен из контекста задач. Но в ряде случаев, когда, по нашему мнению, следовало подчеркнуть, что мы имеем дело с векторами, последние выделены полужирным шрифтом.

Математические символы объяснены по тексту и дан их список.

Чтобы познакомить читателя с разнообразными формами записи решения в линейном программировании, в процессе изложения материала приведены различные формы симплекс-таблиц.

Глава 1. ВВЕДЕНИЕ В МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

1.1. Общие положения математического программирования

Деятельность отдельных людей и коллективов, как правило, связана с выбором таких решений, которые позволили бы получить некие оптимальные результаты — затратить минимум средств на питание семьи, достичь максимальной прибыли предприятия, добиться наилучших показателей в спорте и т. д. Но в каждой конкретной ситуации надо считаться с реальными условиями, накладываемыми на решение данной задачи. При расчете затрат на питание следует приобретать те продукты и в таком количестве, чтобы организм получил необходимые ему жиры, белки, углеводы и т. п.; достичь максимальной прибыли предприятия нельзя, не учитывая реальных запасов сырья, его стоимости и целого ряда других факторов; для достижения наилучших показателей в спорте необходимо правильно организовать тренировку спортсменов, оптимально использовать имеющиеся технические средства и площадки, правильно сформировать команду. Но чтобы что-то рассчитать, надо формализовать задачу, т. е. составить математическую модель изучаемого явления, поскольку математические методы можно применять не непосредственно к изучаемой деятельности, а лишь к математическим моделям того или иного круга явлений. Результаты исследований математических моделей представляют практический интерес только тогда, когда модели адекватно отображают реальные ситуации, достаточно совершенны.

Приведенные примеры позволяют выделить в наших моделях цель и сформулировать *целевую функцию* (*оптимизационный критерий*): минимум затрат, максимум прибыли, наилучшие спортивные достижения — и *условия-ограничения*: необходимое количество жиров, белков и углеводов; запасы сырья и его стоимость; возможности спортивных площадок и различные варианты состава команд. Задача отыскания минимума некоторой функции $f(x)$ эквивалентна задаче отыскания максимума той же функции, взятой со знаком минус (рис. 1.1) и наоборот. Поскольку принципиальных отличий в отыскании минимума и максимума не существует,

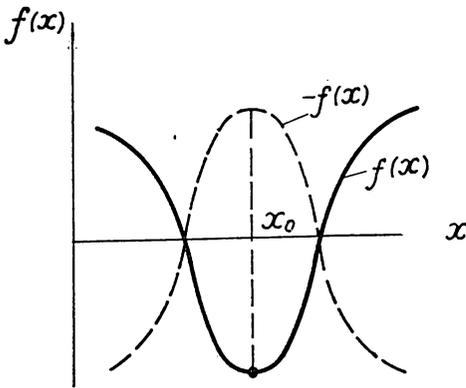


Рис. 1.1. Положения максимума и минимума функций $f(x)$ и $-f(x)$

вует, будем говорить об оптимальном (optimum — наилучший) значении целевых функций. Итак, сформулированные в реальных задачах требования могут быть выражены количественными критериями и записаны в виде математических выражений, т. е. мы можем перевести условие задачи на математический язык и получить так называемую математическую формулировку задачи. Процесс формирования реальной математической задачи и последующего

ее решения достаточно сложен. Его можно представить в виде следующих этапов.

1. *Изучение объекта.* Анализ особенностей функционирования объекта; определение факторов, оказывающих влияние (их числа и степени влияния); получение характеристик объекта в различных условиях; выбор оптимизируемого критерия.

2. *Описательное моделирование.* Установление и словесная фиксация основных связей и зависимостей между характеристиками процесса или явления с точки зрения оптимизируемого критерия.

3. *Математическое моделирование.* Перевод описательной модели на формальный математический язык. Все условия записывают в виде соответствующей системы равенств и неравенств, а критерий оптимизации — в виде функции. После того как задача записана в математической форме, ее конкретное содержание перестает нас интересовать до проведения содержательного анализа получаемого решения. Дело в том, что различные по своему содержанию задачи часто можно свести к одной и той же формальной математической записи.

4. *Выбор или создание метода решения.* Исходя из полученной математической записи задачи выбирают, либо известный метод решения, либо некую модификацию известного метода, либо разрабатывают новый метод решения. Под *допустимым решением* понимают такой набор значений искомых величин (переменных), который удовлетворяет поставленным условиям-ограничениям задачи. *Решением задачи* будет то решение из множества допустимых решений, при котором целевая функция достигает своего наибольшего (наименьшего) значения.

5. *Выбор или написание программы для решения задачи на ЭВМ.* Задачи, содержащие целевую функцию и условия-ограничения и описывающие поведение реальных объектов, как правило, имеют большое число переменных и большое число зависимостей

(уравнений связи) между ними. Поэтому в разумные сроки они могут быть решены только с помощью ЭВМ. Программа для ЭВМ реализует выбранный метод решения задачи.

6. *Решение задачи на ЭВМ.* Необходимую информацию для решения задачи вводят в память ЭВМ вместе с программой. В соответствии с программой ЭВМ обрабатывает введенную числовую информацию, получает решение и выдает его пользователю в заданной им форме.

7. *Анализ полученного решения.* Анализ решения бывает *формальным* и *содержательным*. При формальном (математическом) анализе проверяют соответствие полученного решения построенной математической модели, т. е. проверяют, правильно ли введены исходные данные, правильно ли функционируют программа, ЭВМ и т. д. При содержательном анализе проверяют соответствие полученного решения тому реальному объекту, который моделировали. В результате содержательного анализа в модель (словесную и математическую) могут быть внесены изменения, затем весь рассмотренный процесс повторяют.

Только после полного завершения анализа модель можно использовать для расчета. Чтобы подчеркнуть важность содержательного анализа, приведем следующий пример. Когда впервые решили задачу о питании, то в качестве фактора оптимизации взяли минимум затрат, а в условии-ограничение включили только требование по калорийности пищи. Решение задачи было таким: питаться следует *уксусом*, который входит в состав всевозможных продуктов питания — и калорийность обеспечена, и стоимость минимальна.

Построим математическую модель *задачи о питании*.

Пусть хозяйка собирается в магазин за продуктами. Предположим, что в рацион семьи входят три различных питательных вещества и требуется их соответственно не менее b_1, b_2, b_3 единиц. В магазине продается пять различных продуктов по цене c_1, c_2, \dots, c_5 . Единица продукта i -го вида содержит a_{ij} j -го питательного вещества, т. е., например, a_{23} показывает, что в единице второго продукта третьего питательного вещества будет a_{23} единиц. Какое количество продуктов каждого вида x_1, x_2, x_3, x_4, x_5 следует купить хозяйке, чтобы стоимость продуктов была минимальна, но в то же время рацион семьи содержал все необходимые питательные вещества в нужном количестве?

Целевая функция этой задачи — минимизировать по x_1, x_2, x_3, x_4, x_5 стоимость продуктов:

$$c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 = \sum_{i=1}^5 c_i x_i \rightarrow \min_{x_1, \dots, x_5}.$$

Условия-ограничения задачи: количество первого питательного вещества должно быть не менее b_1 , т. е.

$$a_{11}x_1 + a_{21}x_2 + a_{31}x_3 + a_{41}x_4 + a_{51}x_5 = \sum_{i=1}^5 a_{i1}x_i \geq b_1.$$

Аналогично для других питательных веществ получим неравенства

$$a_{12}x_1 + a_{22}x_2 + a_{32}x_3 + a_{42}x_4 + a_{52}x_5 \equiv \sum_{i=1}^5 a_{i2}x_i \geq b_2;$$

$$a_{13}x_1 + a_{23}x_2 + a_{33}x_3 + a_{43}x_4 + a_{53}x_5 \equiv \sum_{i=1}^5 a_{i3}x_i \geq b_3.$$

Очевидно, что количество продуктов

$$x_1 \geq 0; x_2 \geq 0; x_3 \geq 0; x_4 \geq 0; x_5 \geq 0.$$

Разработкой методов решения задач, содержащих целевую функцию и условия ограничения (задачи на условный экстремум), занимается раздел математики — *математическое программирование*.

Математическое программирование — математическая дисциплина, посвященная теории и методам решения задач о нахождении экстремумов функций на множествах, определяемых линейными и нелинейными ограничениями в виде равенств и неравенств.

Название «математическое программирование» связано с тем, что целью решения задач является выбор программы действий.

Отдельные задачи математического программирования и методы их решения известны давно. Так в числе *старинных русских задач* по математике есть следующая:

сколько надо взять бабе на базар для продажи *живых* гусей, уток и кур, чтобы выручить как можно больше денег, если можно взять товар весом не более P кг; причем известно, что a_1 — вес одной курицы, a_2 — вес одной утки, a_3 — вес одного гуся, c_1 — стоимость одной курицы, c_2 — стоимость одной утки, c_3 — стоимость одного гуся. Пусть x_1, x_2, x_3 — число соответственно кур, уток и гусей, взятом бабой для продажи.

Целевая функция в этой задаче — максимальная стоимость птицы:

$$c_1x_1 + c_2x_2 + c_3x_3 \equiv \sum_{i=1}^3 c_i x_i \rightarrow \max_{x_1, x_2, x_3},$$

Условие-ограничение — вес товара, который может взять баба:

$$a_1x_1 + a_2x_2 + a_3x_3 \equiv \sum_{i=1}^3 a_i x_i \leq P.$$

Среди старинных задач по математике встречается *задача о Кенигсбергских мостах*, сформулированная и решенная известным математиком Леонардом Эйлером в 1736 г.: можно ли поочередно обойти все семь мостов города Кенигсберга, соединяющих районы этого города с островом на реке Прегель, пройдя по каждому мосту только один раз. Как мы увидим дальше, это тоже задача математического программирования.

Принципиальные результаты теории оптимизации, явившейся основой математического программирования, были получены еще в период становления математического анализа. В этой связи следует отметить теорему французского математика П. Ферма (1601—1665 гг.) о необходимом условии локального экстремума в *безусловной задаче оптимизации* и исследования другого французского математика Ж. Лагранжа (1736—1815 гг.) в теории *условных экстремумов*, указывающие необходимые условия экстремума в задаче оптимизации при наличии ограничений в виде равенств. Ж. Лагранж предложил метод решения задач на условный экстремум (1797 г.), который заключается в сведении этих задач к задачам на безусловный экстремум вспомогательной функции — *функций Лагранжа*. Сам метод получил название — *метод (неопределенных) множителей Лагранжа*. Функцию Лагранжа применяют как при исследовании задач *вариационного исчисления*, так и задач математического программирования.

Естественное развитие методов математического анализа, посвященных определению точек экстремумов функций, привело к таким математическим дисциплинам, как вариационное исчисление и математическое программирование. *Вариационное исчисление* — математическая дисциплина, занимающаяся отысканием экстремальных (наибольших и наименьших) значений *функционалов* — переменных величин, зависящих от выбора одной или нескольких функций. Одной из первых задач вариационного исчисления была знаменитая *задача о брахистохроне* И. Бернулли (1696 г.): определить форму кривой, лежащей в вертикальной плоскости, по которой тяжелая материальная точка, двигаясь под действием одной только силы тяжести и не имеющая начальной скорости, перейдет из верхнего положения в нижнее за минимум времени. Эта задача сводится к отысканию функции $y(x)$, представляющей минимум функционалу

$$T(y) = \int_a^b \sqrt{1 + (dy/dx)^2} dx$$

где a и b — абсциссы верхней и нижней точек.

Несмотря на столь ранние истоки математического программирования, его развитие относится к концу 30-х годов нашего столетия. Математическое программирование развивалось как метод решения задач управления и планирования, а также в связи с возникшими в 50-е годы разделом математики «*исследование операций*» и совокупностью методологических средств, называемых *системным анализом*.

Наиболее разработанным разделом математического программирования является *линейное программирование*, содержащее теорию и методы решения условных экстремальных задач, в которых критерии оптимальности *линейно* зависят от неизвестных, а ограничения — *линейные равенства и неравенства*. Развитие линейного программирования тесно связано с задачами управления и планирования. Первые публикации по линейному программиро-

ванию принадлежат советскому ученому Л. В. Канторовичу («Математические методы в организации и планировании производства». ЛГУ, 1939, с. 67), удостоенному в 1975 г. совместно с американским ученым Т. Купмансом Нобелевской премии за вклад в теорию оптимизации распределения ресурсов.

Математическое программирование стало бурно развиваться наряду с развитием вычислительной техники и применением ЭВМ в научных исследованиях. Появление быстродействующих вычислительных машин создало мощные предпосылки для автоматизации многочисленных задач управления и стимулировало разработку специальных новых математических методов, позволяющих сводить решение задач управления и планирования к последовательности автоматически выполняемых операций в соответствии с исходной информацией,— математического, в основном линейного программирования.

Методы математического программирования применялись и одновременно развивались во время второй мировой войны для планирования военных операций. Еще до начала второй мировой войны методы анализа военных систем с использованием математического программирования стали применяться военными специалистами в Великобритании, а затем и в других странах. В США и Канаде были созданы специальные подразделения, занимавшиеся анализом военных операций. В 1938 г. в США был введен термин «*исследование операций*» для характеристики рода деятельности необычной исследовательской группы, созданной по инициативе Air Ministry Research Station и выполнявшей работы по анализу военных систем, в частности решавшей задачи оптимального использования радиолокационных установок в общей системе обороны страны. Этот анализ являлся основой для принятия командованием соответствующих решений.

Впоследствии исследование операций сформировалось в научное направление.

Исследование операций — научный метод выработки количественно обоснованных рекомендаций по принятию решений.

Описание всякой задачи исследования операций включает задание компонентов (факторов) решения, налагаемых на них ограничений и системы целей. Каждой из целей соответствует целевая функция, заданная на множестве допустимых решений, значения которой выражают меру осуществления цели.

Среди задач исследования операций выделяются те, в которых имеется *одна* целевая функция, принимающая численные значения. Это и есть задачи математического оптимального программирования, т. е. математическое программирование — раздел науки об исследовании операций. Задачи с *несколькими* целевыми функциями или с *одной* целевой функцией, но *принимаящей векторные значения* или *значения еще более сложной природы*, называются *многокритериальными*. Они решаются путем сведения к задачам с единственной целевой функцией либо на основе использования *теории игр*.

Задачи исследования операций классифицируют и по их *теоретико-информационным свойствам*. Если субъект в ходе принятия решения сохраняет свое информационное состояние, т. е. никакой информации не приобретает и не утрачивает, то принятие решения можно рассматривать как мгновенный акт. Соответствующие задачи называются *статическими*. Напротив, если субъект в ходе принятия решения изменяет свое информационное состояние, получая или теряя информацию, то в такой *динамической задаче* обычно целесообразно принимать решение *постанно* (многошаговые решения). Значительная часть теории динамических задач исследования операций входит в *динамическое программирование*.

С конца 40-х годов сфера приложения исследования операций стала охватывать разнообразные стороны человеческой деятельности. Исследование операций используется для решения как чисто технических (особенно технологических), так и технико-экономических задач, а также задач управления на различных уровнях.

Лишь отдельные задачи исследования операций поддаются аналитическому решению и сравнительно немногие — численному решению вручную. Поэтому рост возможностей исследования операций тесно связан с прогрессом вычислительной техники.

В настоящее время издается большое число научных журналов по исследованию операций, первый из которых был издан в 1950 г. В 1957 г. в Лондоне был созван первый конгресс Международной Федерации обществ исследования операций (International Federation of Operations Research Societies — **IFORS**); эти конгрессы проводят каждые три года.

С 50-х годов нашего столетия для обоснований решений по сложным проблемам в системах политического, социального, военного, экономического, научного и технического характера стала применяться совокупность методологических средств, получившая название *системный анализ*. Системный анализ используют для решения таких задач, как распределение производств, мощностей между различными видами изделий, определение будущей потребности в новом оборудовании и в рабочей силе той или иной квалификации, прогнозирование спроса на различные виды продукции, а также при решении проблем, связанных с развитием и техническим оснащением вооруженных сил, освоением космоса и т. д.

Системный анализ опирается на ряд прикладных математических дисциплин и разделов, в частности на исследования операций. Когда в задаче системного анализа имеется *одна* четко выраженная цель, степень достижения которой можно оценить на основе *одного* критерия, используют методы математического программирования. Первое применение системного анализа в военном деле относят к истории древних веков, когда правитель Сиракуз обратился к Архимеду с просьбой помочь осажденному городу прорвать осаду римлян.

Отдельные исследования по системному анализу проводились в конце XIX и начале XX веков, а также в первую мировую войну. Так, в 1886 г. военное командование прибегло к системному анализу, чтобы принять решение относительно производства 12-дюймовых орудий (1 дюйм — 2,54 см), заряжающихся с казенной части и предназначенных для использования в береговой артиллерии. Необходимо было сделать выбор между орудием, выпускаемым фирмой *Krupp*, и орудием нового образца американского производства. Во время первой мировой войны с помощью системного анализа разрабатывались, например, стратегические планы борьбы с подводными лодками.

Но эти работы не имели практического выхода и были неизвестны. Поэтому во время второй мировой войны работы пришлось начинать заново. Системный анализ (и его часть — исследование операций) во время второй мировой войны применяли в основном в области тактики. Например, для того, чтобы решить, что использовать в первую очередь в качестве радиолокационных помех — пассивные или активные помехи; как определить наиболее эффективные цели для бомбометания; какие из способов обнаружения подводных лодок являются наилучшими и т. п.

После войны центр тяжести исследований сместился с тактических задач на задачи планирования, так как возникла необходимость создания новых видов оружия. Затем область применения методов исследования операций расширилась настолько, что охватила и военно-политические доктрины государств.

Примерами задач, решаемых с помощью методов исследования операций и использующих математическое программирование, могут быть следующие:

1) разработка методов управления людьми и техникой, обеспечивающих достаточно высокий уровень эффективности деятельности людей;

2) разработка методов использования имеющихся в распоряжении людей техники, обеспечивающей выполнение поставленной задачи с минимальными затратами или с максимальным эффектом;

3) определение техники, материалов, которые необходимо разработать, создать, приобрести и развернуть в рамках общей стратегии деятельности людей.

Первые две задачи наиболее полно допускают применение формальных методов, которые являются наиболее продуктивными. Методы, используемые при принятии решений и распределении ресурсов в различных областях науки и техники (экономике, торговле и промышленности — управление запасами, назначение персонала, составление маршрутов и т. п.) — практически не отличаются и интенсивно развиваются в последние десятилетия.

Важным классом задач математического программирования являются так называемые *сетевые (потоковые) задачи*, в терминах которых могут быть сформулированы задачи линейного программирования.

Рассмотрим в качестве примера так называемую транспортную задачу, являющуюся одной из первых потоковых задач, решенную в 1941 г. Хитчкоком Ф. Л.

Пусть имеется два завода и три склада. Заводы производят соответственно s_1 и s_2 единиц продукции, возможности складов — d_1 ; d_2 ; d_3 единиц; $s_1 + s_2 = d_1 + d_2 + d_3$. Задача состоит в том, чтобы минимизировать затраты на перевозку продукции с заводов на склады.

Пусть x_{ij} — объем продукции, который необходимо перевезти с i -го завода на j -й склад и c_{ij} — стоимость перевозки единицы продукции с i -го завода на j -й склад. Тогда целевая функция задачи — стоимость перевозки:

$$c_{11}x_{11} + c_{12}x_{12} + c_{13}x_{13} + c_{21}x_{21} + c_{22}x_{22} + c_{23}x_{23} \equiv \sum_{i=1}^2 \sum_{j=1}^3 c_{ij}x_{ij} \rightarrow \min.$$

Условие, что вся продукция будет увезена с каждого завода:

$$x_{11} + x_{12} + x_{13} \equiv \sum_{j=1}^3 x_{1j} = s_1;$$

$$x_{21} + x_{22} + x_{23} \equiv \sum_{j=1}^3 x_{2j} = s_2.$$

Эти два равенства можно записать кратко:

$$\sum_{j=1}^3 x_{ij} = s_i; \quad i = 1, 2.$$

Условие заполнения складов

$$\sum_{i=1}^2 x_{ij} = d_j; \quad j = \overline{1, 3}$$

Причем $x_{ij} \geq 0$; $i = \overline{1, 2}$; $j = \overline{1, 3}$.

Эта модель может быть описана с помощью сети, если предположить, что узлами сети являются заводы и склады, а дугами — имеющиеся для перевозки груза дороги (рис. 1.2). Сформулированная транспортная задача является частным случаем задачи поиска потока минимальной стоимости на сети. Сетевые задачи применяют при проектировании и совершенствовании больших и сложных систем, а также при поиске путей их наиболее рационального использования. 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) строить модель сложной системы как совокупность простых систем;

2) составлять формальные процедуры для определения качественных характеристик системы;

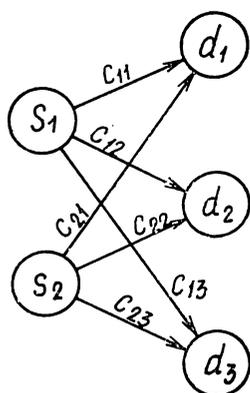


Рис. 1.2. Сеть для транспортной задачи

1) сетевые модели могут точно описывать 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. Пользователю легче понять сетевую диаграмму, чем абстрактные формулы;

3) сетевые алгоритмы позволяют находить наиболее эффективные решения при изучении некоторых больших систем;

4) по сравнению с другими сетевые алгоритмы нередко позволяют решать задачи со значительно большим числом переменных и ограничений. Это становится возможным из-за того, что при этом часто удается ограничиться изучением лишь части рассматриваемой системы.

Сетевой анализ берет начало с упоминавшейся задачи Эйлера о мостах Кенигсберга. Спустя более века Джеймс Клерк Максвелл и Густав Роберт Кирхгофф, исследуя электрические цепи, сформулировали некоторые основные принципы сетевого анализа. В начале XX века европейскими и американскими инженерами были разработаны методы расчета наибольшей пропускной способности телефонных линий и коммутаторов, позволяющие обеспечить гарантированное обслуживание определенного числа абонентов. В связи с развитием вычислительной техники стали проводиться более глубокие исследования построенных моделей и поиск путей реализации алгоритмов на ЭВМ.

В настоящее время трудно назвать область практической и научной деятельности, где бы не применялись методы математического программирования: планирование производства; управление запасами полезных ископаемых и трудовыми ресурсами; планирование и размещение объектов; техническое обслуживание оборудования; планирование работ над проектами и календарное планирование; построение систем — вычислительных, информационных, городской сферы обслуживания, здравоохранения, электро-

3) указывать механизм взаимодействия компонентов управляющей системы с целью описания последней с помощью ее основных характеристик;

4) определять, какие данные необходимы для исследования системы;

5) проводить исследование управляющей системы и составлять предварительное расписание работы ее компонентов.

Основное достоинство *сетевого подхода* заключается в том, что он может быть успешно применен к решению практически любых задач, когда исследователь может точно построить сетевую модель. Преимущество использования *сетевых моделей* можно сформулировать следующим образом:

элементами столбца, и наоборот. Обозначается эта операция индексом «Т»:

$$(x_1, x_2, \dots, x_n)^T \equiv \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Общая задача математического программирования разбивается на задачи, названия которых определяются видом функций, которые необходимо оптимизировать и которые входят в условия-ограничения, типом переменных задач, алгоритмом решения. Если функции $f(x)$, $g_i(x)$, $h_j(x)$ в выражениях (1.1)—(1.3) *линейны*, то полученную задачу называют задачей *линейного программирования* (например, рассмотренные ранее задача о питании и транспортная задача).

Если хотя бы одна из функций $f(x)$, $g_i(x)$, $h_j(x)$ *нелинейна*, то (1.1)—(1.3) называют задачей *нелинейного программирования*. Многие задачи, в свою очередь, разбивают на подмножества. Так, если $f(x)$ является *квадратичной функцией*, а ограничения *линейны*, то получаем задачу *квадратичного программирования* (более точно $f(x)$ должна быть квазиопределенной квадратичной формой)*.

В *сепарабельном программировании* целевая функция $f(x)$ представляет собой сумму функций, различных для каждой переменной. Условия-ограничения здесь могут быть как линейными, так и нелинейными, но все недиагональные элементы матрицы, состоящей из вторых частных производных любой функции задачи, равны нулю.

Если координаты искомого вектора x являются только целыми числами, то получаем задачу *целочисленного программирования* (линейного или нелинейного).

1.3. Некоторые сведения об экстремуме функции, частных производных, градиенте и производной по направлению

Из курса математики читателю известны простейшие задачи на отыскание точек *максимума или минимума функции* одной переменной. Функция $y=f(x)$, определенная в точке x_0 , достигает максимума (минимума) в окрестности точки x_0 , если для всех точек x этой окрестности удовлетворяется неравенство $f(x) \leq f(x_0)$.

Максимум и минимум функции объединяют одним названием *экстремум*. Как правило, точка x_0 — внутренняя точка естественной области определения функции $f(x)$, и экстремум называют *внутренним*. Если существует производная $f'(x)$ в точке x_0 , то функция $f(x)$ может иметь в точке x_0 внутренний экстремум лишь в том случае, когда при $x=x_0$ производная $f'(x_0)=0$ (*необходимое условие экстремума*). Экстремум может быть и в тех точ-

* Часть определений дана в § 1.3.

ках x_0 , где производная $f'(x_0)$ не существует. Но выполнение необходимого условия еще не означает, что в точке x_0 будет экстремум. Для того, чтобы в точке x_0 был экстремум, производная $f'(x)$ в окрестности точки x_0 при переходе через $x=x_0$ должна менять свой знак с плюса на минус в точке максимума и с минуса на плюс в точке минимума. Можно применить и другой признак: если в точке x_0 первая производная $f'(x_0)=0$ и существует вторая производная $f''(x_0)$, то в точке x_0 будет максимум при $f''(x_0)<0$ и минимум при $f''(x_0)>0$.

В общем случае, если существуют производные от $f(x)$ до n -го порядка включительно и, если $f'(x_0)=\dots=f^{(n-1)}(x_0)=0$ и $f^{(n)}(x_0)\neq 0$, то функция $f(x)$ имеет в точке x_0 максимум при n четном и $f^{(n)}(x_0)<0$ и минимум при n нечетном и $f^{(n)}(x_0)>0$. Если n нечетно, то функция $f(x)$ в точке x_0 не имеет ни максимума, ни минимума, а имеет точку *перегиба*.

Дадим несколько определений, которые потребуются в дальнейшем.

Действительная функция $f(x)$, определенная при $x=x_0$, имеет в точке x_0 (*локальный*) максимум или (*локальный*) минимум $f(x_0)$, если существует такое положительное число δ , что при всех приращениях Δx независимого переменного x , равных $x-x_0$, для которых выполняется неравенство $0<|\Delta x|<\delta$ и существует значение $f(x_0+\Delta x)$, приращение данной функции соответственно

$$\Delta f = f(x_0 + \Delta x) - f(x_0) < 0$$

или

$$\Delta f \equiv f(x_0 + \Delta x) - f(x_0) > 0.$$

Если в каждом из этих случаев выполняются нестрогие неравенства, то говорят, что функция $f(x)$ имеет в точке x_0 нестрогий максимум (минимум).

Локальный (максимум) минимум называют *внутренним (максимумом) минимумом* или *граничным (максимумом) минимумом*, если соответственно точка x_0 является внутренней или граничной точкой области определения функции $f(x)$.

В формулировке задачи должна быть точно указана область определения функции $f(x)$. Например, функция $f_1(x)=x$ при $-\infty < x < \infty$ не имеет максимума, а функция $f_2(x)=x$ при $x \leq 1$ имеет при $x=1$ граничный максимум.

Если неравенства $f(x) < f(x_0)$ [$f(x) > f(x_0)$] выполняются для любой точки x , принадлежащей области определения функции $f(x)$, то говорят о *глобальном максимуме (минимуме)* функции $f(x)$ в точке x_0 .

Аналогичные определения справедливы для функции многих переменных.

Функцию $f(x)$, имеющую в данной точке $x=x_0$ производную, называют дифференцируемой в этой точке; функцию, имеющую производную во всех точках некоторого промежутка (a, b) , называют *дифференцируемой* в этом промежутке.

Функцию многих переменных, имеющую полный дифференциал (в данной точке, области), называют дифференцируемой (в

этой точке, области). *Необходимое условие* дифференцируемости функции многих переменных — наличие частных производных первого порядка (в точке, в области). *Достаточные условия* дифференцируемости функции многих переменных — существование и непрерывность всех частных производных первого порядка (в точке, в области).

Числовую функцию $f(x)$ одного векторного аргумента $x = \{x_1, x_2, \dots, x_n\}$ вида

$$f(x) = \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i x_j \equiv \sum_{i=1}^n a_{ii} x_i^2 + 2 \sum_{1 < i < j < n} a_{ij} x_i x_j,$$

где a_{ij} — элементы симметричной матрицы (квадратной таблицы чисел) $A = \|a_{ij}\|_{n \times n}$ порядка n , $a_{ij} = a_{ji}$, называют *квадратичной формой* n переменных.

Квадратичную форму $f(x)$ называют *положительно (отрицательно) определенной*, если для любого ненулевого вектора x выполняется неравенство

$$f(x) > 0 \quad (f(x) < 0).$$

Такие формы объединяют общим названием — *знакоопределенные*. Если же имеется ненулевой вектор x , для которого $f(x) = 0$, форму называют *квазизнакоопределенной*.

Квадратичную форму называют *знакопеременной*, если существуют такие векторы x_1 и x_2 , что $f(x_1) > 0$, $f(x_2) < 0$.

Для наглядного представления поведения функции $y = f(x)$ строят *график функции*. Если независимую переменную x (аргумент) и зависимую переменную y рассматривать как декартовы координаты на плоскости, то действительная функция $y = f(x)$ действительного переменного x изобразится кривой — графиком функции y от x .

Для функции многих переменных $y = f(x_1, x_2, \dots, x_n)$ упорядоченному множеству значений независимых переменных x_1, x_2, \dots, x_n ставят в соответствие значения переменного y . Множество значений x_1, x_2, \dots, x_n , для которых определено соотношение $y = f(x_1, x_2, \dots, x_n)$, есть *область определения функции* $f(x_1, x_2, \dots, x_n)$.

Графиком функции многих переменных является *поверхность* для функций двух переменных и *гиперповерхность* — для большего числа переменных. Чтобы представить функцию n переменных, вводятся понятия *линий* и *поверхностей уровня*. Это геометрическое место точек, в которых функция принимает одно и то же значение. Уравнение поверхности уровня имеет вид $f(x_1, x_2, \dots, x_n) = C$. Давая константе C различные значения, получаем семейство поверхностей уровня, определяющих поведение функции. Линии уровня вводятся для функции двух переменных: $f(x_1, x_2) = C$. Семейство линий уровня дает возможность представить функцию двух переменных $y = f(x_1, x_2)$ на плоскости. Например, семейство линий уровня на географических картах дает представление и о морских глубинах, и о высоте горных хребтов.

Для характеристики скорости изменения функции многих переменных относительно одной из переменных, например x_i , при фиксированных значениях остальных независимых переменных вводится понятие *частных производных* $\partial f/\partial x_i$. Частная производная $\partial f/\partial x_i$ ($i=1, n$) может быть найдена посредством дифференцирования функции $f(x_1, x_2, \dots, x_n)$ по x_i , если остальные $n-1$ независимых переменных рассматривать как постоянные параметры.

Направление, в котором скорость возрастания функции многих переменных наибольшая, определяется вектором, называемым *градиентом*. Противоположное направление называют *антиградиентом*. Градиент скалярной функции $f(x_1, x_2, \dots, x_n)$ есть *векторная функция* точки и определяется как

$$\text{grad } f \equiv \nabla f(x_1, x_2, \dots, x_n) = \frac{\partial f}{\partial x_1} \mathbf{i} + \frac{\partial f}{\partial x_2} \mathbf{j} + \dots + \frac{\partial f}{\partial x_n} \mathbf{k},$$

где ∇ — знак градиента, набла; $\mathbf{i}, \mathbf{j}, \dots, \mathbf{k}$ — единичные векторы (орты), направленные по координатным осям: $\mathbf{i}=\{1, 0, \dots, 0\}$; $\mathbf{j}=\{0, 1, \dots, 0\}$; $\mathbf{k}=\{0, 0, \dots, 1\}$. Иногда применяется обозначение градиента в виде $\nabla_x f$; индекс x у оператора набла показывает переменные, по которым определяется градиент. Другими словами, градиент скалярной функции — это вектор, координатами которого являются частные производные заданной функции.

Скорость изменения скалярной функции $f(x_1, x_2, \dots, x_n)$ в произвольном направлении, задаваемом единичным вектором $\mathbf{u} = \cos\alpha \cdot \mathbf{i} + \cos\beta \cdot \mathbf{j} + \dots + \cos\gamma \cdot \mathbf{k}$ с направляющими косинусами $\cos\alpha, \cos\beta, \dots, \cos\gamma$ определяется *производной по направлению* (действительное число)

$$\frac{df}{du} = \frac{\partial f}{\partial x_1} \cos\alpha + \frac{\partial f}{\partial x_2} \cos\beta + \dots + \frac{\partial f}{\partial x_n} \cos\gamma.$$

Производная по направлению с градиентом скалярной функции ∇f связана *скалярным произведением* $df/du = (\nabla f, \mathbf{u})$.

Скалярным произведением двух векторов $\mathbf{a}=\{a_1, a_2, \dots, a_n\}$ и $\mathbf{b}=\{b_1, b_2, \dots, b_n\}$ называют действительное число, равное сумме произведений соответствующих координат векторов; $(\mathbf{a}, \mathbf{b}) = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$ или произведению длин этих векторов на косинус угла между ними: $(\mathbf{a}, \mathbf{b}) = |\mathbf{a}| \cdot |\mathbf{b}| \cos(\mathbf{a}, \mathbf{b})$. Градиент ∇f всегда ортогонален поверхности (линии) уровня функции $f(x_1, x_2, \dots, x_n)$.

Действительно $df/du = |\nabla f| \cdot |\mathbf{u}| \cos(\nabla f, \mathbf{u})$. Производная по направлению касательной к поверхности (линии) уровня df/du равна нулю, $|\nabla f| \neq 0$; $|\mathbf{u}| \neq 0$. Поэтому $\cos(\nabla f, \mathbf{u}) = 0$;

$$(\nabla f, \mathbf{u}) = \frac{\pi}{2}; \quad \nabla f \perp \mathbf{u}.$$

Нам потребуются понятия *линейной зависимости* и *независимости векторов*. Векторы $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m$ называют линейно зависимыми, если найдутся такие действительные числа $\alpha_1, \alpha_2, \dots, \alpha_m$, не все равные нулю, что линейная комбинация векторов $\mathbf{a}_1, \mathbf{a}_2,$

..., a_m равна нулю: $\alpha_1 a_1 + \alpha_2 a_2 + \dots + \alpha_m a_m = 0$. Если же это равенство выполняется только тогда, когда все числа $\alpha_1, \alpha_2, \dots, \alpha_m$ равны нулю, то векторы a_1, a_2, \dots, a_m называют линейно независимыми.

Из определения линейной зависимости векторов следует, что если векторы линейно зависимы, то один из них может быть представлен в виде линейной комбинации остальных, и, наоборот, если один из векторов есть линейная комбинация остальных, то векторы линейно зависимы.

1.4. Особенности нахождения оптимальных решений в задачах математического программирования

В задачах математического программирования требуется найти так называемый условный экстремум (максимум или минимум) функции при наличии ограничений. Рассмотрим задачу математического программирования, в которой есть только ограничения в виде равенств. Пусть целевая функция задачи является функцией двух переменных: $z = f(x) = f(x_1, x_2)$. Ее аргументы связаны уравнением $\varphi(x_1, x_2) = 0$ (ограничения в виде неравенств отсутствуют). Если функции $z = f(x_1, x_2)$ поставить в соответствие некоторую поверхность, то в данной задаче необходимо найти следующие точки: 1) принадлежащие линии пересечения поверхности $z = f(x_1, x_2)$ и цилиндра с образующей, параллельной оси OZ , и с направляющей $\varphi(x, x_2) = 0$; 2) в которых функция $z = f(x_1, x_2)$ принимает экстремальные значения (рис. 1.3). Как видно из рис. 1.3, точки условного экстремума A и B не совпадают с наибольшим или наименьшим значением функции $z = f(x_1, x_2)$ — с безусловным экстремумом функции $f(x_1, x_2)$. Если из уравнения связи $\varphi(x_1, x_2) = 0$ можно выразить в явном виде одну переменную через другую, например $x_2 = \psi(x_1)$, то $z = f(x_1, x_2) = f[x_1, \psi(x_1)]$ становится функцией одной переменной x_1 и ее безусловный экстремум отыскивается традиционными методами (приравняем первую производную от $f[x_1, \psi(x_1)]$ по x_1 нулю). Безусловный экстремум функции $f(x_1, \psi(x_1))$ является условным экстремумом для функции $f(x_1, x_2)$ при ограничении $\varphi(x_1, x_2) = 0$.

Однако выразить в явном виде из условий-ограничений необходимую часть переменных, как правило, не удастся.

Лагранж предложил оригинальный метод нахождения условного экстремума функции. Метод носит его имя. Пусть требуется решить следующую задачу: минимизировать $f(x_1, x_2, \dots, x_n)$ при ограничениях $h_j(x_1, x_2, \dots, x_n) = 0$; $j = \overline{1, p}$. По условию задачи составляется функция Лагранжа

$$F(x_1, x_2, \dots, x_n) = f(x_1, x_2, \dots, x_n) + \sum_{j=1}^p \lambda_j h_j(x_1, x_2, \dots, x_n).$$

Здесь λ_j — неизвестные постоянные множители, подлежащие определению (множители Лагранжа), т. е. требуется найти n неизвестных x_1, x_2, \dots, x_n и p множителей Лагранжа $\lambda_1, \lambda_2, \dots, \lambda_p$.

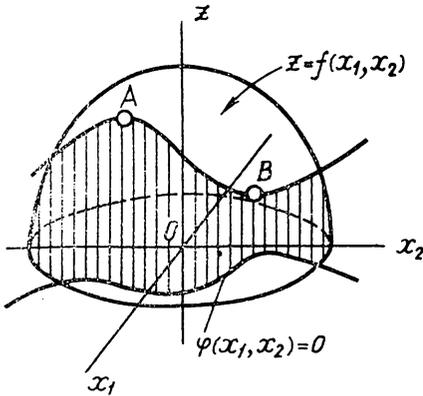


Рис. 1.3. Геометрическая интерпретация метода Лагранжа

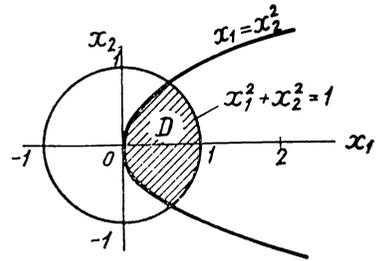


Рис. 1.4. Область допустимых значений x_1 и x_2

Для рассматриваемого в начале параграфа примера

$$F(x_1, x_2) = f(x_1, x_2) + \lambda \varphi(x_1, x_2).$$

Точки, в которых возможен экстремум, находятся как решения системы алгебраических уравнений, полученной приравняв нулю частные производные от функции Лагранжа по искомым переменным (n уравнений) и включением в эту систему p ограничений-равенств. Метод Лагранжа сводит задачу отыскания *условного экстремума* функции $f(x)$ к задаче отыскания *безусловного экстремума* функции $F(x, \lambda)$.

Ограничения-неравенства еще более усложняют задачу. Дело в том, что ограничения-неравенства задают область допустимых значений переменных. Например, пусть требуется оптимизировать некоторую функцию $f(x)$ при ограничениях

$$g_1(x) = x_1 - x_2^2 \geq 0;$$

$$g_2(x) = 1 - x_1^2 - x_2^2 \geq 0.$$

Область допустимых значений переменных x_1 и x_2 в этой задаче есть пересечение области, лежащей «внутри» параболы $x_1 = x_2^2$ с кругом единичного радиуса, уравнение окружности которого имеет вид $x_1^2 + x_2^2 = 1$ (рис. 1.4).

Пересечение цилиндра, направляющей которого является граница полученной области D , с поверхностью $z = f(x)$ может давать самые разнообразные варианты. На рис. 1.5а—в показаны поверхности, полученные в результате пересечения цилиндра, направляющей которого служит граница области допустимых значений переменных x_1 и x_2 , и поверхности, соответствующей целевой функции $z = f(x_1, x_2)$. На рис. 1.5а точка M безусловного экстремума функции $z = f(x_1, x_2)$ является и точкой условного экстремума задачи. На рис. 1.5б точка M является уже граничной и

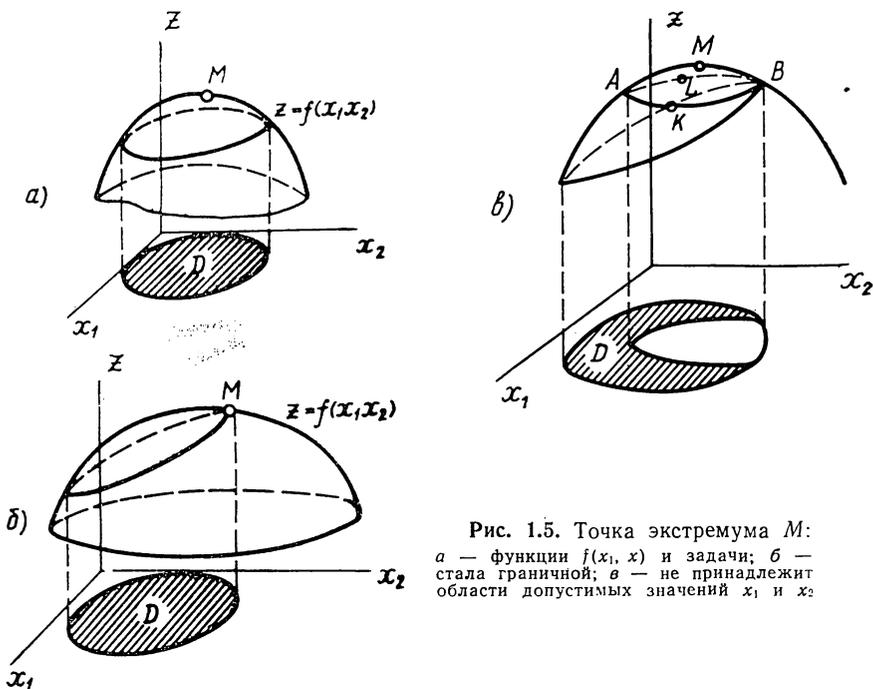


Рис. 1.5. Точка экстремума M :
 а — функции $f(x_1, x_2)$ и задачи; б — стала граничной; в — не принадлежит области допустимых значений x_1 и x_2

в ней целевая функция достигает своего наибольшего значения. На рис. 1.5в точка M не принадлежит области допустимых значений переменных, а целевая функция имеет равные наибольшие значения по линиям ALB и AKB (т. е. неясно, что же брать за решение задачи). Эти неоднозначные результаты получены даже в случае, когда поверхность целевой функции $z=f(x)$ достаточно проста и обладает единственным (глобальным) максимумом.

Наиболее полные результаты в задачах математического программирования получены для *выпуклых* целевых функций, когда область допустимых значений является *выпуклым множеством*. Множество точек D называют *выпуклым*, если для любых точек M_1 и M_2 , принадлежащих области D , отрезок M_1M_2 принадлежит множеству (области) D (рис. 1.6а). Другими словами, любая точка $[\lambda M_1 + (1-\lambda)M_2]$ принадлежит области D для любого λ , $0 \leq \lambda \leq 1$, и для любых точек M_1 и M_2 , принадлежащих области D . Причем пересечение конечного числа выпуклых множеств выпукло.

На рис. 1.6б показаны невыпуклые множества. Функцию $f(M)$ называют *выпуклой* на непустом выпуклом множестве D , если для любых двух точек M_1 и M_2 , принадлежащих области D , и любого числа λ , $0 \leq \lambda \leq 1$, справедливо неравенство

$$f[\lambda M_1 + (1 - \lambda) M_2] \leq \lambda f(M_1) + (1 - \lambda) f(M_2) .$$

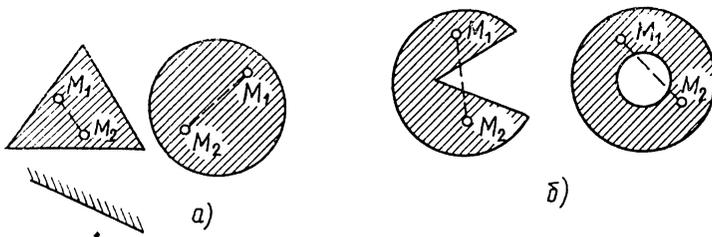


Рис. 1.6. Выпуклые (а) и невыпуклые (б) области (множества)

Функцию $f(M)$ называют строго выпуклой, если для $0 < \lambda < 1$ и $M_1 \neq M_2$ выполняется строгое неравенство

$$f[\lambda M_1 + (1 - \lambda) M_2] < \lambda f(M_1) + (1 - \lambda) f(M_2).$$

Геометрически выпуклая функция лежит над своими касательными. Примером выпуклой функции является парабола.

Сумма выпуклых на множестве D функций есть также выпуклая на D функция.

Функцию $f(x)$ называют *вогнутой* на выпуклом множестве D , если функция $-f(x)$ выпукла на D .

Ограничения $g_i(x) \geq 0$; $i = \overline{1, m}$ образует *выпуклое множество* D (выпуклую область D), если все функции $g_i(x)$ *вогнуты*.

В математическом программировании выделяется важный класс задач — *задачи выпуклого программирования*:

минимизировать $f(x)$

при ограничениях $g_i(x) \geq 0$, $i = \overline{1, m}$,

где $f(x)$ — выпуклая функция, а все функции $g_i(x)$ — вогнуты, т. е. рассматривают выпуклые функции на выпуклых множествах.

Задачи выпуклого программирования обладают важным положительным свойством: *локальные минимумы* целевых функций являются одновременно *глобальными* (единственными).

Очевидно, что решить подобную задачу проще (но не просто!), чем в случае, когда целевая функция $f(x)$ и область D будут общего вида.

1.5. Необходимые и достаточные условия оптимума в задачах математического программирования

В общем случае в задачах математического программирования ставится вопрос об отыскании локального минимума (максимума) целевой функции, т. е. такого значения x^* , что для значений x , принадлежащих некоторой окрестности этого значения x^* , выполняются неравенства $f(x^*) \leq f(x)$ для строго минимума (максимума) и $f(x^*) \leq f(x)$ для нестрогого минимума (максимума). Как и для функций одной переменной, в задачах математического программирования требуется сформулировать необходимые и достаточные условия существования оптимума. Если в задаче матема-

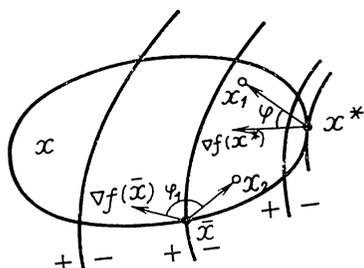


Рис. 1.7. Положение градиента $\nabla f(x)$ в точке решения x^* и в точке x , не являющейся решением задачи математического программирования (знаками «+» и «-» указано направление возрастания значений линий уровня)

тического программирования множество D выпукло, а функция $f(x)$ дифференцируема в точке $x^* \in D$, то градиент $\nabla f(x^*)$, если он отличен от нуля, составляет нетупой угол φ с вектором, направленным из x^* в любую точку $x_1 \in D$. Другими словами, скалярное произведение $(\nabla f(x^*), x_1 - x^*) \geq 0$ (рис. 1.7). Для точки x , не являющейся решением задачи, всегда найдется такая точка x_2 , что φ_1 будет больше $\pi/2$.

В тех случаях, когда решение x^* принадлежит внутренней области D , градиент $\nabla f(x^*) = 0$.

Сформулированное условие является *необходимым условием локальной оптимальности* в задаче минимизации дифференцируемой функции на выпуклом множестве (для выпуклой задачи оно является и *достаточным условием глобальной оптимальности*).

Для области D в виде параллелепипеда, когда $a_i \leq x_i \leq b_i$, $-\infty < a_i < b_i < +\infty$, $i = \overline{1, n}$, данное (необходимое) условие следует понимать как

$$\frac{\partial f(x^*)}{\partial x_i} \begin{cases} = 0, & \text{если } a_i < x_i^* < b_i, \\ \geq 0, & \text{если } x_i^* = a_i \neq -\infty; \\ \leq 0, & \text{если } x_i^* = b_i \neq +\infty. \end{cases}$$

Здесь градиент «смотрит» внутрь области D .

Чтобы определить координаты возможной оптимальной точки, надо из необходимых условий составить соответствующие системы уравнений и решить их.

В общем случае для задачи вида

$$f(x) \rightarrow \min; \quad (1.4)$$

$$g_i(x) \geq 0, \quad i = \overline{1, m}; \quad (1.5)$$

$$h_j(x) = 0, \quad j = \overline{1, p}, \quad (1.6)$$

вводится функция Лагранжа

$$L(x, u, \lambda) = f(x) - \sum_{i=1}^m u_i g_i(x) + \sum_{j=1}^p \lambda_j h_j(x), \quad (1.7)$$

где $u_i (i = \overline{1, m})$, $\lambda_j (j = \overline{1, p})$ — множители Лагранжа, подлежащие определению наряду с координатами вектора x . Множители Лаг-

ранжа λ_j для ограничений-равенств могут иметь любой знак, множители Лагранжа u_i для ограничений-неравенств — неотрицательны. Если в задаче математического программирования (1.4) — (1.6) множество P , $x \in P \subset R^n$, выпукло, функции $f(x)$, $g_i(x)$, $i = \overline{1, m}$ выпуклы на P и дифференцируемы в точке $x^* \in D$, $D = \{x \in P \mid g_i(x) \geq 0, i = \overline{1, m}, h_j(x) = 0, j = \overline{1, p}\}$, функции $h_j(x)$, $j = \overline{1, p}$ линейны и при некоторых u_i^* , λ_j^* выполняются условия $(\nabla_x L(x, u^*, \lambda^*), x - x^*) \geq 0$ при всех $x \in P$ и $u_i^* g_i(x^*) = 0, i = \overline{1, m}$, то x^* — (глобальное) решение этой задачи.

Соотношения

$$(\nabla_x L(x^*, u^*, \lambda^*), x - x^*) \geq 0 \quad (1.8)$$

• при всех $x \in P$,

$$u_i^* g_i(x^*) = 0, i = \overline{1, m}, \quad (1.9)$$

для задачи выпуклого программирования являются не только необходимыми, но и достаточными условиями существования решения (условия Куна-Таккера):

а) если x^* является внутренней точкой области P ; $x^* \in \text{int} P$, то условие (1.8) эквивалентно $\nabla_x L(x^*, u^*, \lambda^*) = 0$;

б) если область P имеет вид параллелепипеда

$$P = \{x \in R^n \mid a_k \leq x_k \leq b_k, k = \overline{1, n}\},$$

где $-\infty < a_k < b_k < +\infty$, то соотношение (1.8) эквивалентно следующему условию:

для любого $k = \overline{1, n}$

$$\frac{\partial L}{\partial x_k}(x^*, u^*, \lambda^*) \begin{cases} = 0, & \text{если } a_k < x_k < b_k; \\ \geq 0, & \text{если } x_k^* = a_k \neq -\infty; \\ \leq 0, & \text{если } x_k^* = b_k \neq +\infty. \end{cases}$$

в) если P учитывает условие неотрицательности части (s) переменных и имеет вид

$$P = \{x \in R^n \mid x_k \geq 0, k = \overline{1, s}\},$$

где $0 \leq s \leq n$, то условие (1.8) эквивалентно совокупности условий:

$$\begin{aligned} \frac{\partial L}{\partial x_k}(x^*, u^*, \lambda^*) &\geq 0, \\ x_k^* \frac{\partial L}{\partial x_k}(x^*, u^*, \lambda^*) &= 0, k = \overline{1, s}; \\ \frac{\partial L}{\partial x_k}(x^*, u^*, \lambda^*) &= 0, k = \overline{s+1, n}. \end{aligned} \quad (1.10)$$

В отличие от методов отыскания оптимальных решений в задачах без ограничений-неравенств здесь появляется дополнитель-

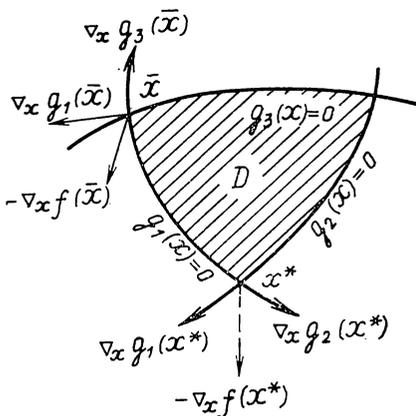


Рис. 1.8. Связь направлений градиентов активных ограничений и антиградиента целевой функции в точке решения x^* и в точке \bar{x} , не являющейся точкой решения

ное условие (1.9), которое называют *условием дополняющей нежесткости*:

$$u_i^* g_i(x^*) = 0, \quad i = \overline{1, m}.$$

Это условие разделяет ограничения-неравенства на активные, которые в точке оптимума обращаются в нуль ($g_i(x^*) = 0, i = \overline{1, l_1}; l_1 \leq m$), и пассивные ($g_i(x^*) \neq 0, i = \overline{1, l_2}, l_1 + l_2 = m$). Для пассивных ограничений коэффициенты Лагранжа u_i^* должны быть равны нулю, при этом пассивные ограничения не оказывают своего влияния на решение x^* .

Рассмотрим случай, когда в функции Лагранжа присутствуют только ограничения-неравенства:

$f(x) \rightarrow \min, g_i(x) \leq 0, i = \overline{1, m}$. Тогда в точке минимума

$$\nabla_x f(x^*) + \sum_{i=1}^m u_i^* \nabla_x g_i(x^*) = 0,$$

т. е. антиградиент целевой функции является неотрицательной линейной комбинацией градиентов функций, образующих активные ограничения в точке x^* (рис. 1.8).

На рис. 1.8 показано множество, образованное неравенствами $g_1(x) \leq 0, g_2(x) \leq 0, g_3(x) \leq 0$. Здесь же в точках x^* и \bar{x} указаны направления градиентов активных ограничений и антиградиента целевой функции. Отсюда следует, что точкой оптимума не может быть точка \bar{x} , так как в ней не выполняется условие того, что антиградиент $f(x)$ есть положительная линейная комбинация градиентов активных ограничений. Решением является точка x^* , где данное условие выполняется.

Мы рассмотрели некоторые условия существования решения, учтя производные первого порядка. Как и для функции одной переменной, при анализе условий оптимальности можно рассматривать производные высших порядков (в частности второго). В задачах математического программирования сформулированы и доказаны условия оптимальности второго порядка, в которых оперируют вторыми частными производными от функции Лагранжа. Здесь мы их рассматривать не будем.

В общем случае задачу математического программирования можно было бы решать по следующей схеме.

1. Запись задачи в канонической форме вида (1.4)—(1.6) и составление функции Лагранжа (1.7).

2. Составление системы условий, которые характеризуют решение (определяют точки, где возможно существование оптимального решения — стационарные точки): в развернутой форме записывают условия (1.8), (1.9), а также условия, накладываемые задачей на допустимые значения x и на множители Лагранжа. Например, для условия (1.10) полная система для определения стационарных точек имеет вид

$$\begin{aligned} x_k &\geq 0; \quad \frac{\partial L}{\partial x_k}(x, u, \lambda) \geq 0; \quad x_k \frac{\partial L}{\partial x_k}(x, u, \lambda) = 0, \quad \bar{k} = \overline{1, s}; \\ \frac{\partial L}{\partial x_k}(x, u, \lambda) &= 0, \quad k = \overline{s+1, n}; \\ u_i &\geq 0, \quad g_i(x) \geq 0; \quad u_i g_i(x) = 0, \quad i = \overline{1, l}; \\ g_i(x) &= 0, \quad i = \overline{l+1, m}. \end{aligned}$$

3. Решение полученной системы необходимых условий. Это удается сделать в аналитическом виде лишь в редких случаях.

4. Если удалось получить решение системы необходимых условий — стационарные точки, надо провести исследование стационарных точек для отбора среди них решений. Это тоже сделать непросто. Иногда проще провести непосредственное исследование поведения целевой функции в стационарной точке.

На последних двух этапах полезно привлечение физических и геометрических соображений о возможном решении задачи математического программирования.

1.6. Теория двойственности и недифференциальные условия оптимальности в задаче выпуклого программирования

В задачах математического программирования (1.4) — (1.6) можно указать условия оптимальности, не прибегая к понятиям производных и градиентов, с помощью так называемой теории *двойственности*. Особенно плодотворен этот подход в задачах выпуклого программирования.

Будем рассматривать функцию Лагранжа (1.7). Обозначим через f^* точную нижнюю грань целевой функции задачи (1.4) — (1.6) на ее допустимом множестве D : $f^* = \inf_{x \in D} f(x)$. Точка $x^* \in D$ является решением задачи (1.4) — (1.6) в том и только в том случае, если $f^* = f(x^*)$.

Введем вектор y с координатами $u_i (i = \overline{1, m})$ и $\lambda_j (j = \overline{1, p})$. Вектор y^* называется вектором Куна-Таккера задачи (1.4) — (1.6), если при всех $x \in D$

$$f^* \leq f(x) - \sum_{i=1}^m u_i^* g_i(x) + \sum_{j=1}^p \lambda_j^* h_j(x) = L(x, y^*).$$

Любой задаче математического программирования можно поставить в соответствие так называемую *двойственную задачу* оп-

тимизации. Между *прямой* и *двойственной* задачами имеются полезные связи.

Двойственной к задаче (1.4) — (1.6) называют задачу

$$\varphi(y) \rightarrow \max; y \in Y,$$

$$\varphi(y) = \inf_{x \in D} L(x, u, \lambda) = \inf_{x \in D} [f(x) - \sum_{i=1}^m u_i g_i(x) + \sum_{j=1}^p \lambda_j h_j(x)];$$

$$Y = \{y \in Q \mid \varphi(y) > -\infty\}.$$

Исходную задачу называют *прямой*. Если целевую функцию в двойственной задаче $\varphi(y) \rightarrow \max$ заменить на $-\varphi(y) \rightarrow \min$, то можно утверждать, что задача, двойственная к произвольной задаче математического программирования, всегда выпукла. Если в задаче математического программирования множество замкнуто и выпукло, функции $f(x)$, $g_i(x)$, $i=1, m$ непрерывны и выпуклы на D , функции $h_j(x)$, $j=1, p$, линейны или отсутствуют и решение прямой задачи конечно ($f^* > -\infty$), в частности она имеет решение, то множество решений двойственной задачи непусто и совпадает с множеством векторов Куна-Таккера прямой задачи. При этом справедливо соотношение двойственности $f^* = \varphi^*$, т. е. *минимум целевой функции прямой задачи совпадает с максимумом целевой функции двойственной задачи*. Учитывая, что число переменных в двойственной задаче равно числу условий-ограничений в прямой задаче, в ряде случаев двойственную задачу решить проще.

Получим необходимые и достаточные условия оптимальности в задаче выпуклого программирования на основе теории двойственности. В этом случае изменится только форма необходимых и достаточных условий (не будут участвовать производные), но предпосылки в обоих случаях одинаковы.

Пара $(x^*, y^*) \in P \times Q$ называется *седловой точкой* функции $L(x, y)$ на $P \times Q$, если

$$L(x^*, y^*) = \min_{x \in P} L(x, y^*);$$

$$L(x^*, y^*) = \max_{y \in Q} L(x^*, y),$$

т. е. $L(x, y^*) \geq L(x^*, y^*) \geq L(x^*, y)$ при всех $x \in P$, $y \in Q$.

Тогда точка $x^* \in P$ является решением прямой задачи в том и только в том случае, если существует вектор $y^* \in Q$ такой, что пара (x^*, y^*) — седловая точка функции Лагранжа $L(x, y)$ на $P \times Q$. Таким образом, если одновременно решать и прямую и двойственную задачи, то к точке минимума (к решению) мы можем приближаться с «двух» сторон.

Вектору Куна-Таккера часто придают экономические интерпретации. Рассмотрим две из них. Пусть предприятие выпускает некую продукцию и стремится получить максимальный доход. Условия-ограничения в виде неравенств характеризуют затраты ресурсов при выпуске продукции. Очевидно, что в процессе вы-

пуска продукции один или несколько ресурсов будут исчерпаны полностью. Для нас это активные ограничения. Другая часть ресурсов будет не использована (пассивные ограничения). В двойственной задаче (согласно условию дополняющей нежесткости) для активных ограничений множители $u_i^* \neq 0, i = \overline{1, k}$, а для пассивных ограничений множители $u_i^* = 0, i = \overline{k+1, m}$ (эти ресурсы недефицитны). То есть предприятию следует закупать в первую очередь те ресурсы, для которых $u_i^* (i = \overline{1, k})$ имеют наибольшее значение. Если же есть возможность увеличивать количество всех ресурсов сразу, то их желательно приобретать в пропорциях, описываемых вектором Куна-Таккера.

В другой интерпретации предприятие хочет продать «ненужную» часть сырья и установить за него такую цену, чтобы максимизировать общий доход. Пусть вектор c — заданный вектор цен на сырье. Предприятие стремится продать сырье по таким ценам, чтобы получить ту же прибыль, как в случае, когда из проданного сырья изготовлена продукция, а для этого цены c должны быть назначены равными координатам вектора Куна-Таккера y^* .

1.7. Графическое решение задач математического программирования

Самыми наглядными методами решения задач математического программирования являются графические. Но они приемлемы только для функций двух и иногда трех переменных. Рассмотрим пример:

$$\text{минимизировать } f(x) = |x_1 - 2| + |x_2 - 2|$$

$$\text{при ограничениях } g_1(x) = x_1 - x_2^2 \geq 0;$$

$$h_1(x) = x_1^2 + x_2^2 - 1 = 0.$$

Получили задачу *нелинейного* математического программирования. Прежде всего построим по условиям-ограничениям допустимую область D — множество точек (x_1, x_2) , удовлетворяющих ограничениям задачи. Ограничение $g_1(x)$ определяет область «внутри» параболы $x_1 = x_2^2$ (рис. 1.9); ограничение $h_1(x)$ — окружность единичного радиуса с центром в начале координат (рис. 1.9).

Допустимая область D этой задачи — дуга окружности \overline{ABC} . Чтобы найти точку, в которой функция $f(x)$ принимает минимальное значение на допустимой области D , построим линии уровня $f(x)$ — штриховые линии. В точке $(2, 2)$ $f(x) = 0$; при $f(x) = 1$ и $f(x) = 2$ линии уровня образуют квадраты. Градиент функции $\nabla f(x)$ направлен в сторону дуги \overline{ABC} , и функция $f(x)$ будет иметь минимальное значение в точке касания линии уровня к дуге \overline{ABC} . Так как линии уровня отсекают от осей x_1 и x_2 равные от-

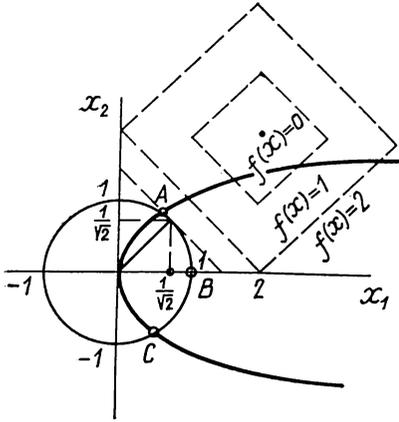


Рис. 1.9. Графическое решение задачи математического программирования

$f(x)$ в области D будет достигнуто в точках $x_1^* = 1/\sqrt{2}$; $x_2^* = 1/\sqrt{2}$.

Рассмотрим второй пример. Пусть в задаче о питании (§ 1.1) $c_1=2$; $c_2=3$; $a_{11}=1$; $a_{12}=5$; $a_{21}=3$; $a_{22}=2$; $a_{31}=2$; $a_{32}=4$; $a_{41}=2$; $a_{42}=2$; $a_{51}=1$; $a_{52}=0$; $b_1=10$; $b_2=12$; $b_3=16$; $b_4=6$; $b_5=1$.

Получили задачу линейного программирования:
минимизировать $f(x) = 2x_1 + 3x_2$
при ограничениях

$$\begin{aligned} x_1 + 5x_2 &\geq 10; & 1 \\ 3x_1 + 2x_2 &\geq 12; & 2 \\ 2x_1 + 4x_2 &\geq 16; & 3 \\ 2x_1 + 2x_2 &\geq 6; & 4 \\ x_1 &\geq 1; & 5 \\ x_1 &\geq 0; \quad x_2 &\geq 0. \end{aligned}$$

Построим области, определяемые неравенствами в ограничениях задачи (рис. 1.10). Строим сначала прямую $x_1 + 5x_2 = 10$ по двум точкам: пусть $x_1=0$, тогда $x_2=2$; при $x_2=0$ $x_1=10$. Нанесем точки (0,2) и (10,0) на график и проведем прямую AB . Чтобы установить, какая часть плоскости определяется неравенством $x_1 + 5x_2 \geq 10$, подставим в него координаты точки (0,0). Получим противоречие, т. е. неравенство определяет полуплоскость, не содержащую точку (0,0). Стрелки на прямой AB указывают эту по-

резки, то координаты точки касания равны $1 \cdot \cos \pi/4 = 1 \cdot \sin \times \pi/4 = 1/\sqrt{2}$. Решение задачи:

$$\begin{aligned} x_1^* &= \frac{1}{\sqrt{2}}, \quad x_2^* = \frac{1}{\sqrt{2}}; \quad f_{\min}(x) = \\ &= f\left(\frac{1}{\sqrt{2}}; \frac{1}{\sqrt{2}}\right) = 2 \left| \frac{1}{\sqrt{2}} \right| - \\ &= 2 \approx 2,6. \end{aligned}$$

Нетрудно видеть, что то же решение будет и в том случае, если вместо $h_1(x)$ взять $g_2(x) = x_1^2 + x_2^2 - 1 \leq 0$. Тогда допустимая область D будет заключена между дугами \widehat{ABC} и \widehat{AOC} (рис. 1.9). Но минимальное значение функции

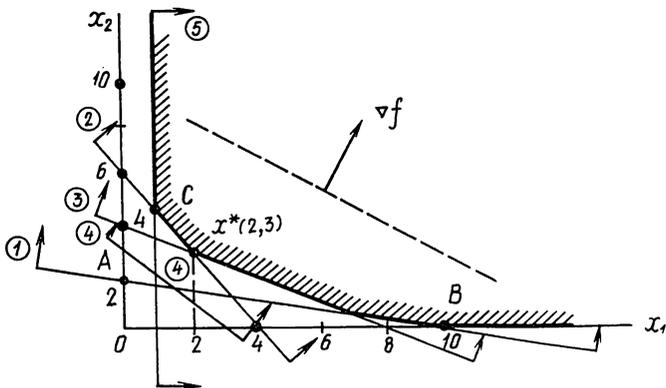


Рис. 1.10. Графическое решение задачи линейного программирования

луплоскость. Аналогично строим области, соответствующие другим неравенствам. Неравенство $x_1 \geq 0$ можно сразу исключить, так как оно «поглощается» неравенством $x_1 \geq 1$. Допустимая область D — заштрихованная выпуклая неограниченная многоугольная область. Чтобы найти оптимальную точку, построим одну из линий уровня целевой функции и ее градиент. Пусть $c=30$, т. е. $2x_1 + 3x_2 = 30$ (штриховая линия), а градиент ∇f имеет координаты $(2, 3)$. Так как требуется определить минимальное значение $f(x)$ на области допустимых значений, то перемещаем линию уровня параллельно себе в направлении антиградиента до тех пор, пока она будет находиться в допустимой области. Точка «выхода» линии уровня из допустимой области и будет точкой, где $f(x)$ примет минимальное значение: $x^* = (2, 3)$; $f_{\min}(x^*) = = 2 \cdot 2 + 3 \cdot 3 = 13$.

На рис. 1.10 видно, что задачи математического программирования могут не иметь решения и могут иметь бесчисленное множество решений. Если бы при тех же ограничениях, какие были заданы в условии задачи, потребовалось максимизировать целевую функцию $f(x)$, то линию уровня пришлось бы перемещать в направлении градиента ∇f . Очевидно, в этом случае решения не существует — множество D не ограничено.

Теперь заменим целевую функцию задачи. Пусть требуется минимизировать функцию $f(x) = 3x_1 + 2x_2$. Очевидно, что линии уровня будут параллельны прямой 2, т. е. линия уровня «выйдет» из допустимой области по отрезку прямой CD : все точки отрезка будут являться решениями задачи (бесчисленное множество решений).

При решении задачи линейного программирования может оказаться, что ограничения противоречивы. Например, если ограничение 4 записать в виде $2x_1 + 2x_2 \leq 6$, то это неравенство будет описывать полуплоскость, включающую точку $(0, 0)$, не имеющую общих точек с другими полуплоскостями (с решением неравенств 1, 2, 3). Решение в данном случае также не существует.

1.8. Простейшая оптимизационная задача

Напомним общие методы решения простейших оптимизационных задач с ограничениями-равенствами. В этих методах обычно с помощью ограничений переменные выражают друг через друга и подставляют в целевую функцию, после чего требуется найти безусловный экстремум целевой функции, который и является решением задачи.

Задача. База берет на себя обязательство хранить товар и выдавать его потребителю в объеме r тонн ежедневно. Стоимость хранения товара h рублей за 1 тонну в сутки. База может получать товар только равными партиями q тонн и через равные промежутки времени T . Стоимость хранения запаса товара q в течение времени T равна hqT . Загрузка базы товаром и подготовка к его приему обходится базе независимо от количества товара в p рублей. Очередной завоз товара производится в момент выдачи предыдущего. Определить оптимальный объем порции товара q и интервала его поставки T , чтобы суточные затраты базы были минимальными.

Решение. Суммарные суточные затраты базы

$c = (hqT + p)/T$ — функция двух переменных q и T . Дополнительное условие $T = q/r$. Минимизируемая функция $c = hq + (p/T) = hq + pr/q$ — функция одной переменной q . Чтобы найти значение q , при котором c будет иметь минимальное значение, найдем производную c по q и приравняем ее нулю:

$$c'_q = h - \frac{pr}{q^2} = 0$$

Отсюда $q^2 = pr/h$, $q = \sqrt{pr/h}$; q не может быть по смыслу задачи отрицательно. Чтобы установить вид экстремума функции c при $q = \sqrt{pr/h}$, найдем вторую производную c'' по q и определим ее знак при $q = \sqrt{pr/h}$:

$$c'' = \frac{pr}{q^3}; \quad c''(q = \sqrt{\frac{pr}{h}}) = \frac{h^{3/2}}{\sqrt{pr}} > 0.$$

Отсюда следует, что функция c при $q = \sqrt{pr/h}$ достигает наименьшего значения. Оптимальный интервал поставки товара

$$T = q/r = \sqrt{\frac{p}{rh}}$$

Суточные затраты при этом

$$c = h \sqrt{\frac{pr}{h}} + \frac{pr \sqrt{h}}{\sqrt{pc}} = 2 \sqrt{phr}.$$

Глава 2. ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

2.1. Математическая постановка задачи линейного программирования

Из рассмотрения задач математического программирования следует, что в общем виде решить их практически невозможно. Целесообразно рассматривать отдельные классы (виды) задач. Для каждого такого класса удается сформулировать алгоритм решения, приемлемый только для данного класса задач. Наиболее разработанными в математическом программировании являются задачи линейного программирования.

В задачах линейного программирования целевая функция линейна, а условия-ограничения содержат линейные равенства и линейные неравенства. Переменные могут быть подчинены или не подчинены требованию неотрицательности. Примерами задач линейного программирования являются уже сформулированные в гл. 1 задача о питании, транспортная задача и т. п. Одна и та же задача линейного программирования может быть записана в различной форме. Говорят, что задача линейного программирования записана в *канонической форме*, если все ее ограничения, кроме $x_j \geq 0, j=1, n$, представляют собой *равенства*. Если все ограничения имеют вид неравенств, то задача записана в *стандартной форме*.

Для записи задачи линейного программирования в различной форме применяются следующие приемы.

1. Точка минимума функции $f(x)$ совпадает с точкой максимума функции $-f(x)$.

2. Ограничения в виде неравенств

$$\sum_{j=1}^n a_{ij}x_j \geq b_i \quad (i = \overline{1, m})$$

можно представить в виде равенств, используя новые переменные $x_i (i = n+1, n+2, \dots, m)$, $x_i \geq 0$, называемые *слабыми*:

$$\sum_{j=1}^n a_{ij}x_j - x_i = b_i.$$

Для неравенства $\sum_{j=1}^n a_{ij}x_j \leq b_i$ можно взять $x_i \geq 0$ и получить равенство $\sum_{j=1}^n a_{ij}x_j + x_i = b_i$.

3. Ограничение в виде равенства $\sum_{j=1}^n a_{ij}x_j = b_i$ можно заменить двумя неравенствами

$$\sum_{j=1}^n a_{ij}x_j \geq b_i, \quad \sum_{j=1}^n a_{ij}x_j \leq b_i.$$

Если имеется m равенств $\sum_{j=1}^n a_{ij}x_j = b_i (i = \overline{1, m})$, их можно заменить $(m+1)$ неравенствами

$$\sum_{j=1}^n a_{ij}x_j \geq b_i (i = \overline{1, m}) \quad \text{и} \quad \sum_{i=1}^m \left(\sum_{j=1}^n a_{ij}x_j - b_i \right) \leq 0.$$

4. Если на переменную $x_j (j = \overline{1, n})$ не наложено условие неотрицательности, ее можно заменить двумя неотрицательными переменными x_j^+ и x_j^- , положив

$$x_j = x_j^+ - x_j^-; \quad x_j^+ \geq 0; \quad x_j^- \geq 0.$$

Если имеется n таких переменных $x_j (j = \overline{1, n})$, то их можно заменить $(n+1)$ неотрицательными переменными x_j^+ и x_0 , положив $x_j = x_j^+ - x_0$.

Система ограничений в виде равенств и неравенств образует выпуклое множество — выпуклый многогранник. Это множество может быть ограниченным и неограниченным. Целевая функция задачи линейного программирования — тоже выпуклая функция. Таким образом, задача линейного программирования является частным случаем задачи выпуклого программирования.

Рассмотрим систему ограничений задачи линейного программирования в виде равенств

$$\sum_{j=1}^n a_{ij}x_j = b_i, \quad i = \overline{1, m}, \quad n > m. \quad (2.1)$$

Говорят, что система (2.1) линейных уравнений *совместна*, если она имеет по крайней мере одно решение. Система (2.1) называется *избыточной*, если одно из уравнений можно выразить в виде линейной комбинации остальных. Система (2.1) *несовместна*, если ранг матрицы $\|a_{ij}\|; i = \overline{1, m}, j = \overline{1, n}$, равен r , ранг *расширенной матрицы* этой системы (с присоединенным столбцом b_i) больше r .

В системе (2.1) число переменных (неизвестных x) n больше, чем число уравнений m . Будем считать, что ранг этой системы равен m (система неизбыточна) и что система (2.1) совместна. Тогда m переменных из общего их числа n образуют *базисные переменные*, а остальные $(n-m)$ переменных называют *свободными*. Система (2.1) в этом случае будет иметь бесчисленное множество решений, так как свободным переменным можно давать любые значения, для которых находят значения базисных переменных. Решение системы (2.1) называют *базисным*, если все свободные переменные равны нулю. Если система уравнений имеет решение, то она имеет и базисное решение. Решение системы уравнений (2.1) называют *допустимым*, если все его компоненты неотрицательны. Если система линейных уравнений обладает допустимым решением, то она имеет и базисное допустимое решение. Совокупность всех допустимых решений системы (2.1) есть выпуклое множество или, другими словами, множество решений задачи линейного программирования выпукло. Так как это множество образовано плоскостями (гиперплоскостями), то оно имеет вид выпуклого многогранника. *Базисное допустимое решение* соответствует крайней точке выпуклого многогранника (его грани или вершине).

Если существует оптимальное решение задачи линейного программирования, то существует *базисное оптимальное решение*.

Целевая функция задачи линейного программирования есть уравнение плоскости (или гиперплоскости для числа переменных больше трех). Пусть в вершинах выпуклого многоугольника мы установили «столбы», высота которых определяет значения целевой функции в данной вершине. На эти «столбы» наложим плоскость (графическое представление целевой функции). Очевидно, что максимальное и минимальное значение целевая функция задачи линейного программирования достигает либо в вершине выпуклого многогранника, либо на одной из его граней. Таким образом, решение (решения) задачи линейного программирования лежит в вершинах выпуклого многогранника и для его нахождения надо вычислить значения целевой функции в вершинах выпуклого многогранника, определяемого условиями-ограничениями задачи.

2.2. Симплекс-метод — основной метод решения задач линейного программирования

Рассмотрим задачу линейного программирования в канонической форме:

$$\text{найти минимум функции } f(x) = \sum_{j=1}^n c_j x_j$$

при условиях $\sum_{j=1}^n a_{ij} x_j = b_i$, $i = \overline{1, m}$, $m \leq n$, $x_j \geq 0$, $j = \overline{1, n}$.

Предполагается, что решение этой задачи существует. Чтобы най-

ти оптимальное решение, надо найти допустимые базисные решения, а из них выбрать оптимальное базисное решение. Для чего мы должны поочередно из столбцов матрицы $\|a_{ij}\|$, $i=1, m$, $j=1, n$ выбирать m столбцов и решать систему из m уравнений с m неизвестными. Такой метод требует решения $C_m^n = n!/[m!(n-m)!]$ уравнений, что практически невозможно, даже для небольших значений m .

Для решения задач линейного программирования в 1949 г. американским математиком Дж. Данцигом разработан симплекс-метод, ставший основным для решения задач линейного программирования.

Разберем главные моменты симплекс-метода на небольшом числовом примере:

$$\text{минимизировать } f(x) = 3 - x_4 + x_5$$

$$\text{при ограничениях } x_2 + 2x_4 + 3x_5 - 7 = 0$$

$$x_3 - x_4 - 3x_5 - 2 = 0$$

$$x_1 + x_4 + x_5 - 2 = 0$$

$$x_1 \geq 0; x_2 \geq 0; x_3 \geq 0;$$

$$x_4 \geq 0; x_5 \geq 0.$$

Определитель, составленный из коэффициентов при неизвестных x_1, x_2, x_3 , имеет вид $\begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{vmatrix}$ и не равен нулю. Поэтому ранг матрицы ограничений равен 3, базисные переменные — x_1, x_2, x_3 , а свободные переменные — x_4, x_5 . Выразим базисные переменные через свободные:

$$x_1 = 2 - x_4 - x_5;$$

$$x_2 = 7 - 2x_4 - 3x_5;$$

$$x_3 = 2 + x_4 + 3x_5.$$

Базисное решение (при нулевых значениях свободных переменных x_4 и x_5) в данном случае $x_1=2$, $x_2=7$, $x_3=2$ и является допустимым (значения x_1, x_2, x_3 положительны). Значение целевой функции при таких значениях переменных $f(x)=3$. Но оно может быть уменьшено, если увеличить значение переменной x_4 , входящей с отрицательным коэффициентом. Очевидно, что увеличивать x_4 можно до тех пор, пока не будут нарушены условия-ограничения задачи, в частности, пока переменные x_1, x_2 и x_3 будут неотрицательны. Например, если $x_4=2$, $x_5=0$, то $x_1=0$, $x_2=3$, $x_3=4$ — новое допустимое решение. При $x_5=0$ переменная $x_1=0$, если $x_4=2/1=2$; $x_2=0$, если $x_4=7/2=3,5$; $x_3=0$, если $x_4=2/(-1)=-2$. Чтобы ни одна из переменных x_1, x_2, x_3 не стала отрицательной, надо выбрать наименьшее положительное отношение элементов столбца свободных членов к соответствующим коэффициентам при x_4 . Берем $x_4=2$, x_1 становится равной нулю, т. е. x_1 переводим в

свободные переменные, а $x_4=2$ становится базисной переменной. Ограничения и целевую функцию надо выразить теперь через x_1 и x_5 : из первого уравнения имеем $x_4=2-x_1-x_5$, из второго $-x_2=3+2x_1-x_5$; из третьего $x_3=4-x_1+2x_5$ и $f(x)=3-2+x_1+x_5+x_5=1+x_1+2x_5$. В данном случае любое увеличение значений свободных переменных x_1 и x_5 ведет к увеличению (но не к уменьшению) значений целевой функции, т. е. получили оптимальное решение:

$$x_1 = 0, x_5 = 0, x_2 = 3, x_3 = 4, x_4 = 2,$$

$$f_{\min}(x) = 1.$$

Какие выводы можно сделать из этого примера? Во-первых, надо так разделить базисные и свободные переменные, чтобы получить допустимое базисное решение, а затем выразить базисные переменные и целевую функцию через свободные переменные. Во-вторых, по знаку коэффициентов при неизвестных в целевой функции следует определить: а) не достигли ли мы уже оптимального решения (нет отрицательных коэффициентов); б) значение какой переменной лучше увеличить, т. е. какую переменную следует перевести в свободные. Другими словами, определяя минимальное положительное отношение элементов столбца свободных членов к коэффициентам при новой свободной переменной, находим переменную, которую необходимо перевести из базисных в свободные. После чего выражаем условия-ограничения и целевую функцию через новые свободные переменные.

Процесс повторяют до тех пор, пока не будет получено оптимальное решение. Если среди коэффициентов при неизвестных в целевой функции есть положительный, а все коэффициенты в условиях-ограничениях при нем неположительны, то задача линейного программирования не имеет оптимального решения, минимальное значение целевой функции равно $-\infty$.

Рассмотрим геометрическую интерпретацию симплекс-метода, давшую название методу.

В условия одной из первых задач линейного программирования, для которых Данциг разработал вычислительный метод, входили ограничения вида

$$\sum_{j=1}^n x_j = 1, x_j \geq 0, j = \overline{1, n}.$$

Эти ограничения в n -мерном пространстве определяют симплекс. Симплекс трехмерного пространства изображен на рис. 2.1. Рассмотрим неравенство $x_1+x_2 \leq b_1$ при условиях $x_1 \geq 0, x_2 \geq 0$. Область решения этого неравенства показана на рис. 2.2. Данное неравенство можно преобразовать в уравнение введением слабой переменной x_3 . Тогда получим систему $x_1+x_2+x_3=b_1, x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$.

Областью решений этой системы является треугольник $\triangle ABC$, показанный на рис. 2.1, если принять, что $A=B=C=b_1$. Каждой точке треугольной области рис. 2.1 соответствует точка обла-

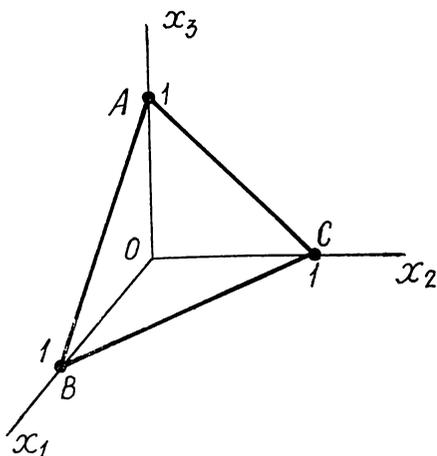


Рис. 2.1. Симплекс трехмерного пространства

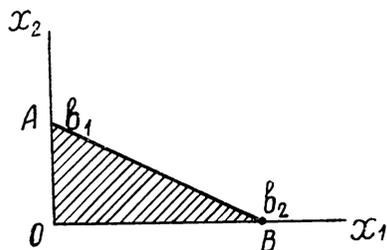


Рис. 2.2. Область решения неравенства

сти на рис. 2.2. Соответствие можно устанавливать, проектируя эту треугольную область на плоскость x_1x_2 . Если придать слабой переменной x_3 постоянное значение c , то x_1 и x_2 должны удовлетворять уравнению $x_1+x_2=b_1-c$, которое является уравнением прямой, параллельной $x_1+x_2=b_1$. Если слабая переменная равна нулю, то $x_1+x_2=b_1$. Таким образом, значение слабой переменной может служить мерой близости точки из треугольной области к границе $x_1+x_2=b_1$ полупространства, определяемого исходным неравенством.

В общем случае в симплекс-методе процедуру поиска начинают с допустимой вершины, а затем переходят в соседнюю вершину так, чтобы значение целевой функции «улучшилось». В пространстве векторов свободных переменных возрастание значения одной из свободных переменных от нуля, при котором остальные свободные переменные остаются равными нулю, соответствует движению из начала системы координат, образованной свободными переменными, по одной из координатных осей. При этом, поскольку $(n-1)$ свободных переменных равны нулю, $(n-1)$ ограничений задачи выполняются как равенства. Другими словами, все соседние с началом координат вершины (которые соответствуют текущему решению) связаны с началом координат $(n-1)$ ребрами выпуклого многогранника. Возрастание от нуля значения некоторой свободной переменной может привести к тому, что эта переменная станет базисной. Для того, чтобы получить в качестве решения вершину, необходимо заменить одну из базисных переменных на свободную, т.е. произвести соответствующее перемещение вдоль одной из координатных осей до тех пор, пока не будет достигнута другая вершина. Если двигаться дальше, то

будет нарушено условие неотрицательности переменных. Таким образом, в симплекс-методе начинают с локальной координатной системы с началом координат, соответствующим текущему решению, и перемещаются вдоль ребра к соседней вершине, в которой значение целевой функции «улучшается». После перехода в новую вершину рассматривают новую систему координат с началом в этой вершине. Если движение осуществляют согласно критерию (выбирают минимальный отрицательный коэффициент при неизвестных в целевой функции), то это соответствует спуску по самому крутому ребру из всех пересекающихся в начале координат. Величину изменения целевой функции за одну итерацию определяют как углом наклона (крутизной) ребра, так и длиной ребра. Более точно она равна минимальному значению по i величины $|c_j b_i / a_{ij}|$ для данного j , где a_{ij} — соответствующий элемент вектора-столбца a_j для j -й свободной переменной.

Замечание. Итак, в симплекс-методе всегда считают, что в первую таблицу внесено допустимое базисное решение. В задачах, описывающих реальные системы, допустимое базисное решение подобрать трудно. Для этого решают вспомогательную задачу линейного программирования, которая позволяет не только найти допустимое базисное решение, но и установить, совместна ли система ограничений исходной задачи.

Пусть система ограничений исходной задачи записана в следующем виде:

$$b_i - \sum_{j=1}^n a_{ij} x_j = 0, \quad i = \overline{1, m},$$

где $b_i \geq 0$, $i = \overline{1, m}$. Этого нетрудно добиться, умножив при необходимости уравнения на -1 .

Введем новые переменные

$$\xi_i = b_i - \sum_{j=1}^n a_{ij} x_j \quad (a)$$

и рассмотрим новую целевую функцию

$$f(\xi) = \sum_{i=1}^m \xi_i \rightarrow \min. \quad (б)$$

Допустимое решение для задачи (а), (б) сразу задано. В процессе решения задачи возможны следующие случаи:

- 1) $\min f(\xi) = 0$, $\xi_i = 0$, $i = \overline{1, m}$ (все ξ_i стали свободными переменными) — полученное решение x_j , $j = \overline{1, n}$, является допустимым решением исходной задачи линейного программирования;
- 2) $\min f(\xi) > 0$ — система ограничений исходной задачи несовместна.

В первом случае можно отметить две особенности:

- а. Целевая функция $f(\xi)$ достигла своего минимума, равного нулю, а некоторые из переменных ξ_i находятся среди базисных,

хотя и равны нулю. При этом нет необходимости обращать внимание на знаки в строке для целевой функции (можно любую свободную переменную выводить в базисные), так как значение целевой функции не изменится, но следует выполнить условия, обеспечивающие допустимость нового базисного решения (рассмотреть минимальное положительное отношение).

б. Даже после выполнения предыдущего пункта в строке для базисной переменной ξ_i нет положительных элементов (нельзя получить положительное отношение). Это означает, что переменные, входящие в уравнение для ξ_i с ненулевыми коэффициентами, должны быть равными нулю в данной задаче. В процессе дальнейшего решения их надо исключить из рассмотрения.

Наряду с решением вспомогательной задачи линейного программирования (а), (б) преобразуется и целевая функция исходной задачи, которая приписывается в задаче (а), (б) в виде дополнительной строки.

2.3. Метод полного исключения Жордана для решения систем линейных алгебраических уравнений

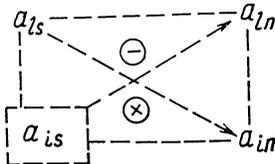
На каждом шаге симплекс-метода требуется определять новые «наборы» базисных и свободных переменных, т. е. решать системы линейных алгебраических уравнений. Задачи линейного программирования решают с помощью стандартных симплекс-таблиц, формализующих алгоритм перевода базисных переменных в свободные. Этот алгоритм и определяет конкретный вид симплекс-таблиц. Рассмотрим симплекс-таблицы, преобразуемые с помощью метода полного исключения Жордана, получившего наибольшее распространение в линейном программировании.

Рассмотрим систему m линейных алгебраических уравнений с n неизвестными:

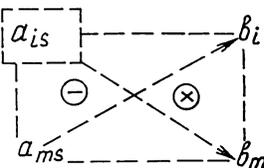
$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1s}x_s + \dots + a_{1n}x_n = b_1; \\ a_{i1}x_1 + a_{i2}x_2 + \dots + a_{is}x_s + \dots + a_{in}x_n = b_i; \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{ms}x_s + \dots + a_{mn}x_n = b_m. \end{cases}$$

В методе полного исключения Жордана делают такие преобразования, в результате которых в каждой строке и в каждом столбце матрицы системы линейных алгебраических уравнений остаются по одному неизвестному с коэффициентами, равными единице. Например, мы хотим исключить переменную x_s из всех строк за исключением i -й строки. Элемент a_{is} — коэффициент, стоящий перед переменной x_s , называют *генеральным элементом*, i -я строка и s -й столбец — *разрешающими*. Прежде всего разрешающую строку делят на a_{is} и она остается неизменной. Чтобы исключить x_s из первого уравнения, умножим разрешающую строку на $(-a_{is})$ и сложим с первой строкой. В результате получим первую строку с нулевым элементом на месте a_{is} . Аналогично исключаем x_s в остальных строках. Получим новую эквивалентную

запись системы алгебраических уравнений. В ней i -я строка имеет прежний вид, но все коэффициенты у нее поделены на a_{is} ; s -й столбец состоит из нулевых элементов (кроме единицы, стоящей в i -й строке). Остальные элементы матрицы системы и столбец свободных переменных пересчитываются по *правилу прямоугольника*. Например, новое значение элемента a_{in} будет равно

$$a_{in}^H = \frac{a_{is} a_{in} - a_{is} a_{in}}{a_{is}},$$


а новое значение b_m столбца свободных членов —

$$b_m^H = \frac{a_{is} b_m - a_{ms} b_i}{a_{is}},$$


Из правила прямоугольника следует, что когда в разрешающей строке (столбце) есть нулевые элементы, то элементы столбцов (строк), пересекающих эти нулевые элементы, остаются без изменения.

В процессе решения задачи линейного программирования симплекс-методом возможно «заикливание». Поясним его суть. Пусть в процессе решения задачи линейного программирования на некотором шаге симплекс-метода наименьших положительных отношений свободных членов к коэффициентам разрешающего столбца оказалось больше одного, т. е. выбор разрешающего элемента неоднозначен. После этого шага все упомянутые свободные члены, за исключением свободного члена разрешающей строки, обратятся в нуль. Этот случай называют *вырождением*: сливаются две или большее число вершин выпуклого многогранника D , когда ребро (или ребра), соединяющие эти вершины, стягиваются в точку.

В алгоритме симплекс-метода каждый шаг означает переход по ребру от данной вершины многогранника D к соседней (расположенной на том же ребре), а при вырождении — совпадение двух соседних вершин — алгоритм может потерять монотонность, т. е. может случиться, что после указанного шага мы остались в той же вершине, только выраженной с помощью другого набора из n уравнений, относящихся к этой вершине. Если продолжать решение симплекс-методом, то не исключено, что после некоторого числа шагов мы вернемся к уже взятой ранее вершине и процесс начнет повторяться. Произойдет *заикливание*. Если в процессе решения проводилось запоминание уже испытанных ребер,

то для прерывания заикливания достаточно сменить генеральный элемент.

Существуют алгоритмы, где автоматически предусмотрены меры против заикливания — «расклеивание» слипшихся вершин.

2.4. Как спланировать выпуск продукции пошивочному предприятию

Выпуск продукции пошивочного предприятия может быть определен из решения задачи линейного программирования. Решать ее будем симплекс-методом. Формулировку задачи и процедуру применения симплекс-метода рассмотрим на конкретном примере.

Задача. Намечается выпуск двух видов костюмов — мужских и женских. На женский костюм требуется 1 м шерсти, 2 м лавсана и 1 человеко-день трудозатрат; для мужского костюма — 3,5 м шерсти, 0,5 м лавсана и тоже 1 человеко-день трудозатрат. На пошив этих костюмов имеется 350 м шерсти, 240 м лавсана и 150 человеко-дней трудозатрат. По плану костюмов не должно быть менее 110 штук и необходимо обеспечить прибыль не менее 1400 руб. Требуется определить оптимальное число костюмов каждого вида, обеспечивающее максимальную прибыль, если прибыль от реализации женского костюма составляет 10 руб., а от мужского — 20 руб.

Решение. Пусть x_1 — число женских костюмов, а x_2 — мужских. Прибыль от женских костюмов составляет $10 x_1$ руб., а от мужских $20 x_2$ руб., т. е. необходимо максимизировать целевую функцию $f(x) = 10 x_1 + 20 x_2$.

Расход шерсти составляет $x_1 + 3,5 x_2$, лавсана $2 x_1 + 0,5 x_2$, трудовых ресурсов $x_1 + x_2$. Поэтому ограничения задачи имеют вид:

$$x_1 + 3,5x_2 \leq 350 ;$$

$$2x_1 + 0,5x_2 \leq 240 ;$$

$$x_1 + x_2 \leq 150 ;$$

$$x_1 + x_2 \geq 110 ;$$

$$10x_1 + 20x_2 \geq 1400 ;$$

$$x_1 \geq 0 ; x_2 \geq 0 .$$

Первые три неравенства описывают ограничения по ресурсам, четвертое и пятое — соответственно плановое задание по общему числу костюмов и ограничение по прибыли.

Для решения задачи симплекс-методом сведем систему ограничений к равенствам путем введения неотрицательных слабых переменных x_3, x_4, x_5, x_6, x_7 (в первом и втором ограничениях про-

Исходная таблица

Базисные переменные	Столбец свободных членов b_i	$-x_1$	$-x_2$	$-x_3$	$-x_4$	$-x_5$	$-x_6$	$-x_7$	Контроль
x_3	700	2	7	1	0	0	0	0	710
x_4	480	4	1	0	1	0	0	0	486
x_5	150	1	1	0	0	1	0	0	153
x_6	-110	-1	-1	0	0	0	1	0	-111
x_7	-140	-1	-2	0	0	0	0	1	-142
$f(x)$	0	-10	-20	0	0	0	0	0	-30
$\varphi(x)$	-250	-2	-3	0	0	0	1	1	-253

пересчетов сумма всех чисел в строках должна быть равной числу в контрольном столбце.

Проведем максимизацию функции $\varphi(x)$. Среди элементов строки $\varphi(x)$ есть отрицательные. Берем меньший отрицательный коэффициент, равный -3 ; он указывает, что переменную x_2 надо перевести в базисные. Чтобы определить, какую переменную надо из базисных перевести в свободные, рассмотрим положительные отношения b_i к соответствующим элементам столбца x_2 : $700/7 = 100$; $480/1 = 480$; $150/1 = 150$; $-110/(-1) = 110$; $-140/(-2) = 70$. Минимальное значение, равное 70, указывает, что x_7 надо перевести в свободные, а *генеральным (разрешающим)* элементом является (-2) (обведен). Для получения следующей симплекса-таблицы применим метод полного исключения Жордана (табл. 2.2).

Таблица 2.2

Первая итерация

Базис	b_i	$-x_1$	$-x_2$	$-x_3$	$-x_4$	$-x_5$	$-x_6$	$-x_7$	Контроль
x_3	210	$-3/2$	0	1	0	0	0	$7/2$	213
x_4	410	$7/2$	0	0	1	0	0	$1/2$	415
x_5	80	$1/2$	0	0	0	1	0	$1/2$	82
x_6	-40	$-1/2$	0	0	0	0	1	$-1/2$	-40
x_2	70	$1/2$	1	0	0	0	0	$-1/2$	71
$f(x)$	1400	0	0	0	0	0	0	-10	1390
$\varphi(x)$	-40	$-1/2$	0	0	0	0	1	$-1/2$	-40

Разрешающую строку (для x_7) делим на (-2) и заносим в табл. 2.2; столбец x_2 заполняем нулями. Столбцы для x_3, x_4, x_5, x_6 переносим без изменения, так как они пересекают нулевые элементы разрешающей строки. Остальные элементы таблицы пере-

считываем по правилу прямоугольника. Проверяем, совпадает ли сумма чисел в строке с числом в контрольном (последнем) столбце. Если совпадения нет, произошла ошибка в расчете. Вместе с функцией $\varphi(x)$ пересчитывают и целевую функцию $f(x)$.

Продолжаем максимизировать $\varphi(x)$. В базисные переменные можно перевести x_1 и x_7 , так как для них есть отрицательные коэффициенты в строке $\varphi(x)$. Переводим x_1 в базисные переменные. По минимуму положительных отношений b_i к элементам столбца ($-x_1$) выбираем элемент x_6 , который надо перевести в свободные; генеральный элемент — $(-1/2)$, разрешающая строка для x_6 и разрешающий столбец для x_1 . Получим табл. 2.3.

Т а б л и ц а 2.3

Вторая итерация

Базис	b_i	$-x_1$	$-x_2$	$-x_3$	$-x_4$	$-x_5$	$-x_6$	$-x_7$	Контроль
x_3	330	0	0	1	0	0	-3	<u>5</u>	333
x_4	130	0	0	0	1	0	7	-3	135
x_5	40	0	0	0	0	1	1	0	42
x_1	80	1	0	0	0	0	-2	1	80
x_2	30	0	1	0	0	0	1	-1	31
$f(x)$	1400	0	0	0	0	0	0	-10	1390
$\varphi(x)$	0	0	0	0	0	0	0	0	0

Из табл. 2.3 видно, что достигнут максимум фиктивной целевой функции ($\varphi(x)=0$), и все коэффициенты в строке $\varphi(x)$ равны нулю, т. е. получено *опорное* решение $x_6=x_7=0$, $x_1=80$, $x_2=30$, $x_3=330$, $x_4=130$, $x_5=40$. Это решение не является оптимальным, так как в строке $f(x)$ имеется отрицательный коэффициент.

Продолжаем улучшать решение симплекс-методом. Строку $\varphi(x)$ исключаем. Генеральным будет элемент (1, 7), равный 5; x_7 переводим в базис (вместо x_3). Получим табл. 2.4, затем табл. 2.5. В табл. 2.4 отмечен генеральный элемент.

Т а б л и ц а 2.4

Третья итерация

Базис	b_i	$-x_1$	$-x_2$	$-x_3$	$-x_4$	$-x_5$	$-x_6$	$-x_7$	Контроль
x_7	66	0	0	1/5	0	0	-3/5	5/5	333/5
x_4	328	0	0	3/5	1	0	26/5	0	1674/5
x_5	40	0	0	0	0	1	<u>1</u>	0	42
x_1	14	1	0	-1/5	0	0	-7/5	0	67/5
x_2	96	0	1	1/5	0	0	2/5	0	488/5
$f(x)$	2060	0	0	2	0	0	-6	0	2056

Оптимальное решение

Базис	b_i	$-x_1$	$-x_2$	$-x_3$	$-x_4$	$-x_5$	$-x_6$	$-x_7$	Контроль
x_7	90	0	0	1/5	0	3/5	0	1	459/5
x_4	120	0	0	3/5	1	-26/5	0	0	582/5
x_6	40	0	0	0	0	1	1	0	42
x_1	70	1	0	-1/5	0	7/5	0	0	361/5
x_2	80	0	1	1/5	0	-2/5	0	0	404/5
$f(x)$	2300	0	0	2	0	6	0	0	2308

В табл. 2.5 все коэффициенты строки $f(x)$ неотрицательны, значит максимум $f(x)$ достигнут и получено соответствующее ему решение: $x_1=70$, $x_2=80$, $f_{\max}(x)=2300$. Таким образом, максимальная прибыль составит 2300 рублей при производстве 70 женских и 80 мужских костюмов. Слабые переменные оказались равными $x_3=0$, $x_4=120$, $x_5=0$, $x_6=40$, $x_7=90$. Что же они показывают? Значения x_3 , x_4 , x_5 показывают остатки ресурсов: шерсть и трудовые ресурсы израсходованы полностью, лавсана осталось 120 м; x_6 и x_7 показывают, на сколько перевыполнены плановые задания по числу костюмов и по прибыли (имеем в виду, что $x_7=90 \cdot 10=900$ руб.).

Замечание. Обратить внимание на тот факт, что в рассмотренной задаче оптимальное решение мы устанавливали по наличию неотрицательных коэффициентов в строке максимизируемой целевой функции, а в примере § 2.2 — по наличию неотрицательных коэффициентов в минимизируемой целевой функции. Здесь никакого противоречия нет: в нашем примере переменные указаны со знаком минус, поэтому и рассматривались неотрицательные коэффициенты.

Рассмотрим графическое решение этой задачи. Поскольку задача двумерная, то решим ее графически. Система ограничений-неравенств определяет многоугольник допустимых решений (рис. 2.3). Определим полуплоскости, задаваемые неравенствами-ограничениями задачи. Для этого построим прямые, заменив в ограничениях знаки неравенств на знаки равенств. Чтобы выяснить, какую часть плоскости описывает неравенство, подставляем в него пробную точку, например $(0, 0)$, и устанавливаем, удовлетворяет ли она неравенству. Если неравенство удовлетворяется, то искомая полуплоскость включает точку $(0, 0)$. В противном случае берут другую половину плоскости.

Для первого неравенства прямую $l_1 (x_1+3,5x_2=350)$ строим по точкам $x_1=0$, $x_2=350/3,5=100$ и $x_2=0$, $x_1=350$. Пробная точка $(0, 0)$ удовлетворяет неравенству $0 < 350$, т. е. точка $(0, 0)$ входит в искомую полуплоскость (она отмечена стрелочками у прямой l_1). Прямую $l_2 (2x_1+0,5x_2=240)$ строим аналогично: при

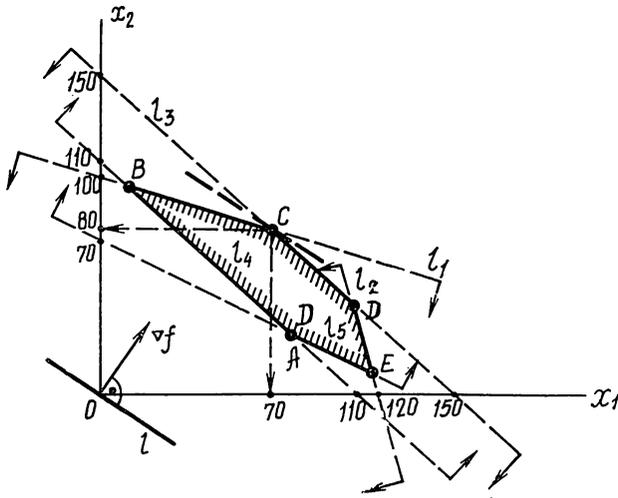


Рис. 2.3. Графическое решение

$x_1=0$, $x_2=240/0,5=480$ и при $x_2=0$ $x_1=240/2=120$. Точка $(0, 0)$ принадлежит искомой полуплоскости. Рассмотрим последнее неравенство $10x_1+20x_2 \geq 1400$, ему соответствует прямая l_5 ($x_1+2x_2=140$): при $x_1=0$, $x_2=140/2=70$ и при $x_2=0$, $x_1=140$. Точка $(0, 0)$ не удовлетворяет неравенству $0 \geq 1400$ (ложно), т. е. надо **взять** полуплоскость, не содержащую точку $(0, 0)$. Пересечение полуплоскостей дает выпуклый многоугольник $ABCDE$. Для нахождения максимума функции $f(x)$ надо построить линию уровня. Пусть $f(x)=0$, тогда уравнение линии уровня l будет $10x_1+20x_2=0$ — прямая, проходящая через начало координат параллельно прямой l_5 : $x_2=-0,5x_1$. Градиент целевой функции $\nabla f = \{10, 20\}$ показывает направление ее возрастания. Прямую l перемещаем параллельно самой себе в направлении ∇f до тех пор, пока она «не выйдет» из области D . Получаем точку C , точку пересечения прямых l_1 и l_3 :

$$\begin{cases} x_1 + 3,5x_2 = 350 ; \\ x_1 + x_2 = 150 \end{cases}$$

Решая полученную систему уравнений, находим оптимальное решение — координаты точки C ($x_1=70$, $x_2=80$) и вычисляем максимальное значение целевой функции

$$f_{\max}(x) = 10 \cdot 70 + 20 \cdot 80 = 2300 \text{ руб.}$$

Замечание. Допустим, что в рассматриваемой задаче требовалось найти минимум целевой функции

$$f(x) = 10x_1 + 20x_2.$$

В этом случае линия уровня «вошла» бы в область по линии l_5 , т. е. все точки отрезка AE являлись бы оптимальным решением (бесчисленное множество решений).

2.5. Двойственность в задачах линейного программирования

В § 1.6 мы убедились, что в ряде случаев проще решить двойственную задачу математического программирования, чем прямую. Рассмотрим теорию двойственности для задач линейного программирования. Для каждой задачи линейного программирования можно построить другую задачу линейного программирования, называемую двойственной. Понятие двойственности дает ощутимые преимущества при построении алгоритмов решения задач линейного программирования.

Запишем обе задачи:

Прямая задача

минимизировать

$$f(x) = \sum_{j=1}^n c_j x_j$$

при условиях

$$\sum_{j=1}^n a_{ij} x_j \geq b_i, \quad i = \overline{1, k};$$

$$\sum_{j=1}^n a_{ij} x_j = b_i, \quad i = k + \overline{1, m};$$

$$x_j \geq 0, \quad j = \overline{1, l};$$

$$x_j \geq 0, \quad j = \overline{l + 1, n};$$

Двойственная задача

максимизировать

$$\Phi(y) = \sum_{i=1}^m b_i y_i$$

при условиях

$$y_i \geq 0, \quad i = \overline{1, k}$$

$$y_i \geq 0, \quad i = k + \overline{1, m};$$

$$\sum_{i=1}^m a_{ij} y_i \leq c_j, \quad j = \overline{1, l};$$

$$\sum_{i=1}^m a_{ij} y_i = c_j, \quad j = \overline{l + 1, n}$$

Симметричность обеих задач очевидна. Неравенству в одной задаче соответствует неотрицательная переменная другой. Равенству одной задачи соответствует свободная переменная другой. Задача, двойственная к двойственной задаче, есть исходная (прямая) задача. Таким образом, любую из этой пары задач можно считать прямой.

Для стандартного и канонического видов задачи линейного программирования двойственные задачи можно записать следующим образом:

Стандартный вид

Прямая задача

минимизировать

$$f(x) = \sum_{j=1}^n c_j x_j$$

Двойственная задача

максимизировать

$$\Phi(y) = \sum_{i=1}^m b_i y_i$$

при условиях

$$\sum_{j=1}^n a_{ij}x_j \geq b_i, \quad i = \overline{1, m};$$

$$x_j \geq 0, \quad j = \overline{1, n};$$

при условиях

$$\sum_{i=1}^m a_{ji}y_i \leq c_j, \quad j = \overline{1, n}.$$

$$y_i \geq 0, \quad i = \overline{1, m}$$

Канонический вид

Прямая задача

минимизировать

$$f(x) = \sum_{j=1}^n c_j x_j$$

при условиях

$$\sum_{j=1}^n a_{ij}x_j = b_i, \quad i = \overline{1, m};$$

$$x_j \geq 0, \quad j = \overline{1, n};$$

Двойственная задача

максимизировать

$$\varphi(y) = \sum_{i=1}^m b_i y_i$$

при условиях

$$\sum_{i=1}^m a_{ji}y_i \leq c_j, \quad j = \overline{1, n};$$

$$y_i \leq 0, \quad i = \overline{1, m}.$$

Из сравнения обеих задач нетрудно видеть, что:

1) матрицу из коэффициентов при переменных в исходной задаче

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

и аналогичную матрицу в двойственной задаче

$$A^T = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix}$$

получают друг из друга простой заменой строк столбцами с сохранением их порядка (такую операцию называют *транспонированием* и обозначают значком «Т»);

2) в исходной задаче имеется n переменных и m ограничений; в двойственной — m переменных и n ограничений;

3) в правых частях систем ограничений каждой из задач стоят коэффициенты целевой функции, взятой из другой задачи;

4) в исходной задаче в систему ограничений входят неравенства типа \geq и требуется *минимизировать* целевую функцию $f(x)$; в двойственной задаче в систему ограничений входят неравенства типа \leq и требуется *максимизировать* целевую функцию $\varphi(y)$.

В теории двойственности доказывают следующую теорему: пусть дана пара двойственных задач линейного программирования

ния (заданных в стандартном виде). Тогда справедливо одно и только одно из следующих утверждений.

1. Обе задачи имеют оптимальные решения и оптимальные значения целевых функций равны, т. е.

$$\min f(x) = \max \varphi(y).$$

2. Одна из задач не имеет ни одного допустимого решения, а другая имеет по крайней мере одно допустимое решение, но не имеет оптимального решения (целевая функция на множестве допустимых решений неограничена).

3. Ни одна пара задач не имеет допустимых решений.

Между решениями пары двойственных задач линейного программирования существуют и другие соотношения, которые устанавливаются теоремами о *дополняющей нежесткости*: для того, чтобы допустимые решения x и y прямой и двойственной задач были оптимальными, необходимо и достаточно, чтобы выполнялись следующие соотношения:

$$\overrightarrow{y}(Ax - b) = 0; \tag{а}$$

$$(c - yA)\overrightarrow{x} = 0. \tag{б}$$

Условие (а) равносильно условиям

$$\text{если } y_i > 0, \text{ то } \sum_{j=1}^n a_{ij}x_j = b_i, \quad i = \overline{1, m};$$

$$\text{если } \sum_{j=1}^n a_{ij}x_j > b_i, \text{ то } y_i = 0.$$

Условие (б) равносильно:

$$\text{если } c_j > \sum_{i=1}^m a_{ij}y_i, \text{ то } x_j = 0;$$

$$\text{если } x_j > 0, \text{ то } c_j = \sum_{i=1}^m a_{ij}y_i.$$

Это условия дополняющей нежесткости в слабой форме. В сильной форме условия дополняющей нежесткости утверждают:

$$\text{если } y_i = 0, \text{ то } \sum_{j=1}^n a_{ij}x_j - b_i > 0,$$

$$\text{если } \sum_{j=1}^n a_{ij}x_j - b_i = 0, \text{ то } y_i > 0.$$

Может случиться, что $y_i = 0$ и $\sum_{j=1}^n a_{ij}x_j = b_i$ одновременно. Но всег-

да существует по крайней мере одна пара оптимальных решений, для которых условия $y_i = 0$ и $\sum_{j=1}^n a_{ij}x_j = b_i$ не могут выполняться одновременно.

Нетрудно проверить, что, если вектор x — решение прямой задачи, а вектор y — решение двойственной задачи, то сумма произведений соответствующих координат векторов x и y равна нулю (скалярное произведение векторов \vec{x} и \vec{y} равно нулю).

Геометрическая интерпретация теории двойственности в задаче линейного программирования. Выберем задачу линейного программирования стандартного вида:

минимизировать

$$f(x) = \sum_{j=1}^n c_j x_j$$

при условиях

$$\sum_{j=1}^n a_{ij}x_j \geq b_i, \quad i = \overline{1, m};$$

$$x_j \geq 0, \quad j = \overline{1, n}.$$

Пусть a_i^* (b_i^*) при $i = \overline{1, m}$ совпадают с a_i (b_i); a_{m+i}^* — единичный орт e_i , а $b_{m+i}^* = 0$, при $i = \overline{1, n}$. В §§ 1.3—1.5 мы видели, что обычное условие наличия безусловного экстремума функции во внутренней точке есть обращение в нуль градиента функции в этой точке. Если при этом должны выполняться некоторые ограничения на переменные в виде равенств, то условием наличия экстремума в допустимой точке будет требование, чтобы в этой точке градиент функции и нормали к поверхностям, соответствующим ограничениям, были направлены «в одну сторону». Более точно градиент функции в этой точке должен быть неотрицательной линейной комбинацией этих нормалей к поверхностям-ограничениям. В задаче линейного программирования каждое неравенство определяет допустимую область — полупространство. Для того, чтобы допустимая точка x была оптимальной, необходимо, чтобы градиент целевой функции в x выражался в виде неотрицательной линейной комбинации направляющих векторов тех и только тех ограничений, которые в точке x обращаются в равенство, т. е. градиент целевой функции (вектор c) есть неотрицательная линейная комбинация нормалей векторов a_i^* для ограничений, обращающихся в равенство:

$$c = \sum_{i=1}^{m+n} y_i a_i^*, \quad y_i \geq 0 \Rightarrow (a_i^*, x) = b_i^*$$

где y_i — соответствующие коэффициенты линейной комбинации.

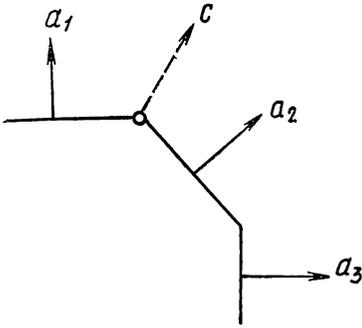


Рис. 2.4. К иллюстрации выполнения условия дополняющей нежесткости

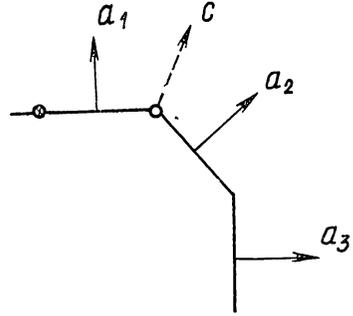


Рис. 2.5. Нарушение сильной формы условия дополняющей нежесткости

Из условий дополняющей нежесткости в слабой форме следовало:

если $y_i > 0$, то $(a_i, x) - b_i = 0$
и если $(a_i, x) - b_i > 0$, то $y_i = 0$.

В сильной форме утверждалось, что,
если $y_i = 0$, то $(a_i, x) - b_i > 0$
и если $(a_i, x) - b_i = 0$, то $y_i > 0$.

На рис. 2.4 изображены три гиперплоскости $(a_i, x) - b_i = 0$ ($i = 1, 2, 3$) и нормали к ним a_1, a_2, a_3 . Если вектор c такой, как показано на рис. 2.4, то он может быть выражен в виде неотрицательной линейной комбинации векторов a_1 и a_2 ; вершина, обозначенная кружком, соответствует оптимальному решению. Здесь выполняются и условия дополняющей нежесткости как в слабой, так и в сильной форме: $y_3 > 0 \Leftrightarrow (a_3, x) - b_3 > 0$; $y_1 > 0$ ($(a_1, x) - b_1 = 0$); $y_2 > 0 \Leftrightarrow (a_2, x) - b_2 = 0$. Если вектор c таков, как показано на рис. 2.5, c — нормаль к одной из гиперплоскостей, $(a_1, x) - b_1 = 0$, то оптимальная вершина в кружке не удовлетворяет сильной форме условия дополняющей нежесткости, поскольку и $y_2 = 0$ и $(a_2, x) - b_2 = 0$. Но точка, помеченная крестом на рис. 2.5 и являющаяся оптимальным решением, удовлетворяет и слабой, и сильной форме дополняющей нежесткости: $y_1 > 0 \Leftrightarrow (a_1, x) - b_1 = 0$, $y_2 = 0 \Leftrightarrow (a_2, x) - b_2 > 0$, $y_3 = 0 \Leftrightarrow (a_3, x) - b_3 > 0$.

Для решения задач линейного программирования разработан так называемый *двойственный симплекс-метод*. Процедуру начинают с двойственно допустимого решения, когда одновременно $b_i \geq 0$ и $c_j \geq 0$, $i = 1, m$, $j = 1, n$, и сохраняют его двойственно допустимым на протяжении всех шагов. Он реализуется посредством таких же таблиц, как и прямой симплекс-метод. Но здесь сначала определяется, какая переменная должна быть выведена из базиса, а затем — какая должна быть введена в базис.

Всегда имеется возможность выбора: решать прямую или двойственную задачу, использовать прямой или двойственный симплекс-метод. Выбирают ту модификацию задачи, которую проще

решать. Например, если исходная задача содержит переменные, на которые не наложено условие неотрицательности, то бывает удобнее решать двойственную задачу. Прежде чем записать двойственную задачу, полезно в исходной освободиться от ограничений в виде равенств, поскольку они будут порождать в двойственной задаче переменные, принадлежащие всей действительной оси. В литературе, кроме того, описаны методы одновременного решения прямой и двойственной задач, например *метод последовательного сокращения невязок* $\left(\sum_{i=1}^n a_{ij}x_j - b_i \right), i = \overline{1, m}$, при фиксированных значениях x_j .

2.6. Как оптимально организовать поставку грузов от поставщиков к потребителям

Симплекс-метод дает возможность решить любую задачу линейного программирования. Однако существует много методов решения задач линейного программирования, которые учитывают конкретные особенности решаемой задачи, а потому более эффективных. Примером одной из таких задач является *транспортная задача*.

Задача. Перевозится однородный груз из трех пунктов A_1, A_2, A_3 к четырем местам назначения B_1, B_2, B_3, B_4 . Из пункта A_1 может быть направлено 50 т, из A_2 — 40 т, из A_3 — 20 т. В пункты назначения должно поступить груза: в B_1 — 30 т, в B_2 — 25 т, в B_3 — 35 т, в B_4 — 20 т. Расстояния c_{ij} от i -го поставщика до j -го потребителя приведены в углах табл. 2.6; $i = \overline{1, 3}, j = \overline{1, 4}$.

Т а б л и ц а 2.6

Исходные данные

	B_1	B_2	B_3	B_4	Запасы
A_1	x_{11}	x_{12}	x_{13}	x_{14}	$a_1=50$
A_2	x_{21}	x_{22}	x_{23}	x_{24}	$a_2=40$
A_3	x_{31}	x_{32}	x_{33}	x_{34}	$a_3=20$
Потребности	$b_1=30$	$b_2=25$	$b_3=35$	$b_4=20$	110

Необходимо составить план перевозки, обеспечивающий наименьший общий пробег транспорта в тонно-километрах при условии, что все запасы должны быть вывезены, а потребитель получит точно необходимое количество груза.

Мы сформулировали сбалансированную транспортную задачу, когда количество груза у поставщиков равно потребности по-

требителей. Несбалансированная транспортная задача сводится к сбалансированной путем введения фиктивного поставщика, если потребности превышают предложения, или фиктивного потребителя в противном случае. Расстояния в фиктивной строке (столбце) указываются равными нулю. Сбалансированная и несбалансированная задачи решаются по одному алгоритму.

Решение. Пусть x_{ij} — количество груза, которое будет доставлено из i -го пункта отправления в j -й пункт назначения; $i = \overline{1,3}$; $j = \overline{1,4}$. Целевая функция задачи — минимизировать общий пробег транспорта в тонно-километрах:

$$f(x) = \sum_{i=1}^3 \sum_{j=1}^4 c_{ij} x_{ij}$$

при условии: весь груз от поставщика должен быть вывезен —

$$\sum_{j=1}^4 x_{ij} = a_i; \quad i = \overline{1,3};$$

каждый потребитель получит необходимое ему количество груза—

$$\sum_{i=1}^3 x_{ij} = b_j, \quad j = \overline{1,4}.$$

Особенностью данной задачи является тот факт, что в матрице ограничений все коэффициенты при неизвестных равны единице. Это облегчает все вычисления по симплекс-методу.

Число переменных в данной задаче равно в общем случае mn , где m — число поставщиков, n — число потребителей. Число уравнений в системе ограничений равно $(m+n)$. Однако нетрудно видеть, что одно из этих уравнений может быть получено из других. Так, например, если определено наличие груза у всех отправителей и потребность всех получателей, кроме одного, то спрос последнего легко установить как разность между общим запасом и общей потребностью остальных получателей, т. е. система ограничений содержит $m+n-1$ независимых уравнений с $m \cdot n$ неизвестными. Число базисных переменных также будет $m+n-1$, остальные переменные свободные.

Алгоритм решения транспортной задачи сопоставим с алгоритмом симплекс-метода.

1. Нахождение допустимого базисного (опорного) решения. В транспортной задаче его находят довольно просто:

а) *метод северо-западного угла.* Удовлетворяем потребность первого потребителя за счет первого поставщика. Если потребности оказались выше возможностей первого поставщика, то подключаем второго поставщика. Если запасы первого поставщика выше потребностей первого потребителя, то остаток запасов первого поставщика передаем второму потребителю и т. д. Мы должны заполнить $m+n-1$ клетку. Может оказаться, что число заполненных клеток меньше $m+n-1$ (случай *вырождения*). Тогда

клетки, недостающие до $m+n-1$, заполняем нулями (эти клетки выбираем произвольно) — это так называемые *условные поставки*. Процесс получения опорного решения нашей задачи методом северо-западного угла представлен в табл. 2.7.

Таблица 2.7

а) первая итерация

	B_1	B_2	B_3	B_4	a_i
A_1	30				50* 20
A_2					40
A_3					20
b_j	30*	25	35	20	110

б) вторая итерация

	B_2	B_3	B_4	a_i
A_1	20			20*
A_2	5			45* 35
A_3				20
b_j	25*	35	20	80

в) третья итерация

	B_3	B_4	a_i
A_2	35		35*
A_3			
b_j	3/5	20	55

г) четвертая итерация

	B_1	B_2	B_3	B_4	a_i
A_1	3 30	2 20	4	1	50
A_2	2	3 5	1 35	5	40
A_3	3	2	4	4 20	20
b_j	30	25	35	20	110

Примечание. Помеченные (*) значения следует читать как зачеркнутые цифры.

За четыре итерации мы заполнили в таблице перевозок пять клеток вместо шести, но удовлетворили условиям-ограничениям. Надо ввести нулевую клетку — условную поставку. Пусть это будет клетка (1,4): $x_{14}=0$. Получили опорное решение: $x_{11}=30$; $x_{12}=20$; $x_{14}=0$; $x_{22}=5$; $x_{23}=35$; $x_{34}=20$. Значение целевой функции при таком решении

$$f(x) = 3 \cdot 30 + 2 \cdot 20 + 1 \cdot 10 + 1 \cdot 35 + 3 \cdot 5 + 4 \cdot 20 = 260 \text{ (тонно-километров);}$$

б) метод учета наименьших расстояний (стоимостей) перевозок. Он аналогичен методу северо-западного угла. Только заполняются в первую очередь те клетки, для которых указанные расстояния (стоимости) наименьшие.

2. Проверка полученного плана перевозок на оптимальность.

С этой целью рассмотрим двойственную задачу к поставленной транспортной задаче. В качестве двойственных переменных введем величины $\alpha_1, \alpha_2, \alpha_3$, соответствующие первым трем ограни-

чениям, и $\beta_1, \beta_2, \beta_3, \beta_4$ — для остальных ограничений. Эти переменные называют *потенциалами*.

Целевая функция двойственной задачи:
максимизировать

$$\varphi(\alpha, \beta) = 50\alpha_1 + 40\alpha_2 + 20\alpha_3 + 30\beta_1 + 25\beta_2 + 35\beta_3 + 20\beta_4$$

при условии (для базисных клеток)

$$\begin{cases} \alpha_1 + \beta_1 \leq 3; \\ \alpha_1 + \beta_2 \leq 2; \\ \alpha_1 + \beta_4 \leq 1; \\ \alpha_2 + \beta_2 \leq 3; \\ \alpha_2 + \beta_3 \leq 1; \\ \alpha_3 + \beta_4 \leq 4. \end{cases} \quad (2.3)$$

Для оптимального плана исходной задачи условия-ограничения двойственной задачи выполнялись бы как равенства, так как согласно условиям дополняющей нежесткости, если в оптимальном плане исходной задачи значение какой-либо переменной строго больше нуля, то соответствующее ограничение двойственной задачи при подстановке в него оптимального плана становится равенством. Получим систему (2.3) из шести уравнений с семью переменными. Поскольку независимых переменных в данной системе ровно $3+4-1=6$, то одна переменная свободная. Пусть это будет α_1 . Положим $\alpha_1=0$, получим $\beta_1=3, \beta_2=2, \beta_4=1, \alpha_2=1, \beta_3=0, \alpha_3=3$. Составим таблицу перевозок для данной итерации (табл. 2.8).

Т а б л и ц а 2.8

Первая итерация

	$\beta_1=3$	$\beta_2=2$	$\beta_3=0$	$\beta_4=1$
$\alpha_1=0$	3 30	2 20	4	1 0
$\alpha_2=1$	2	3 5	1 35	5
$\alpha_3=3$	3	2	4	4 20

Полученное решение $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3, \beta_4$ подставляем в ограничения двойственной задачи, не вошедшие в систему уравнений (2.3), т. е. соответствующие пустым клеткам. Если эти ограничения являются верными неравенствами для найденного решения, то проверяемый допустимый план исходной задачи является оп-

тимальным. В противном случае — не является. Для пустых клеток имеем

$$\left\{ \begin{array}{l} \alpha_1 + \beta_3 \leq 4, \quad 0 \leq 4; \\ \alpha_2 + \beta_1 \leq 2, \quad \underline{4 \leq 2}; \\ \alpha_2 + \beta_4 \leq 5, \quad \underline{2 \leq 5}; \\ \alpha_3 + \beta_1 \leq 3, \quad \underline{6 \leq 3}; \\ \alpha_3 + \beta_2 \leq 2, \quad \underline{5 \leq 2}; \\ \alpha_3 + \beta_4 \leq 4, \quad 3 \leq 4. \end{array} \right. \quad (2.4)$$

Второе, четвертое и пятое неравенства являются неверными, поэтому решение не является оптимальным. Необходимо провести улучшение плана.

3. Составление нового допустимого плана. Наметив свободную переменную, которую надо перевести в базисную, определим базисную переменную, переводимую в свободную. Переменные, соответствующие свободным клеткам (2, 1); (3, 1) и (3, 2), в которых неравенства (2.4) нарушаются, могут быть переведены в базисные. Выбираем клетку (3, 2). В симплекс-методе для выбора генерального элемента требуется рассмотреть положительные отношения в столбце x_{32} . В матрице перевозок положительные коэффициенты в столбце x_{32} равны +1 и отвечают тем базисным клеткам, которые соответствуют отрицательным вершинам некоторой замкнутой ломаной линии, называемой *циклом пересчета* для x_{32} . Следовательно, генеральным элементом является базисная переменная из числа отвечающих отрицательным вершинам цикла пересчета, значение которой минимально.

Цикл пересчета — это замкнутая ломаная линия, начинающаяся в свободной клетке, все остальные вершины которой помещены в базисные клетки и соединены звеньями, лежащими вдоль строк и столбцов матрицы.

В каждой вершине встречаются только два звена, причем одно из них расположено по строке, другое по столбцу. Никакие три вершины, встречающиеся подряд при обходе, не лежат на одной прямой. Если циклом служит самопересекающаяся линия, то точки самопересечения не могут быть ее вершинами. Свободной клетке в цикле присваивают знак «+», другим вершинам — чередующиеся по ходу знаки «—», «+», «—» и т. д. Построим цикл пересчета для свободной клетки (3, 2) (табл. 2.9).

В отрицательных вершинах цикла пересчета стоят два числа: 20, 20; минимальное из них 20. Так как число положительных и отрицательных вершин одинаково, то баланс не нарушится, если в отрицательных вершинах вычесть число a , а в положительных вершинах прибавить это же число a . Вычитая минимальное из чисел, стоящих в отрицательных вершинах, мы получаем новую свободную переменную x_{34} , а базисной станет x_{32} . Прибавим 20 в положительных вершинах цикла, вычтем 20 в отрицательных вершинах и получим следующее допустимое решение (табл. 2.10).

Цикл пересчета для клетки (3,2)

	$\beta_1=3$	$\beta_2=2$	$\beta_3=0$	$\beta_4=1$
$\alpha_1=0$	30	20	—	+
$\alpha_2=1$		5	35	
$\alpha_3=3$			+	—

Т а б л и ц а 2.10

Второе допустимое решение

	$\beta_1=3$	$\beta_2=2$	$\beta_3=0$	$\beta_4=1$
$\alpha_1=0$	3	2	0	1
$\alpha_2=1$	2	3	5	5
$\alpha_3=0$	3	2	20	4

4. Проверка вновь полученного решения на оптимальность. Находим значения потенциалов для данного допустимого решения:

$$\begin{cases} \alpha_1 + \beta_1 = 3; \\ \alpha_1 + \beta_2 = 2; \\ \alpha_1 + \beta_4 = 1; \\ \alpha_2 + \beta_2 = 3; \\ \alpha_2 + \beta_3 = 1; \\ \alpha_3 + \beta_2 = 2. \end{cases}$$

Пусть $\alpha_1=0$, тогда $\beta_1=3$, $\beta_2=2$, $\beta_4=1$, $\alpha_2=1$, $\beta_3=0$, $\alpha_3=0$. Очевидно, что для клетки (2, 1) неравенство $\alpha_2 + \beta_1 \leq 2$ не выполняется: $3 \leq 2$, т. е. данное решение не является оптимальным. Шаги 3 и 4 повторяем до тех пор, пока не достигнем оптимального решения.

Цикл пересчета для клетки (2, 1) показан в табл. 2.10. Третье допустимое решение приведено в табл. 2.11.

Здесь же показаны значения рассчитанных для данной итерации потенциалов. Нетрудно убедиться, что для всех свободных клеток сумма потенциалов меньше c_{ij} , т. е. получено оптимальное решение. Оно означает, что от первого поставщика к первому потребителю надо перевезти 25 т груза, ко второму 5 т, к четвертому 20 т; от второго поставщика надо перевезти к первому потре-

Оптимальное решение

	$\beta_1=3$	$\beta_2=2$	$\beta_3=2$	$\beta_4=1$	a_i
$\alpha_1=0$	3 25	2 5	4	1 20	50
$\alpha_2=-1$	2 5	3	1 35	5	40
$\alpha_3=0$	3	2 20	4	4	20
b_j	30	25	35	25	110

лю 5 т груза, к третьему 35 т; от третьего поставщика надо перевезти 20 т груза только ко второму потребителю. Минимальное значение целевой функции

$$f_{\min}(x) = 3 \cdot 25 + 2 \cdot 5 + 1 \cdot 20 + 2 \cdot 5 + 1 \cdot 35 + 2 \cdot 20 = 190 \text{ (тонно-километров)}.$$

2.7. Задача о перевозках с перегрузкой

Забегав несколько вперед (гл. 3), укажем, что сетевой подход позволяет использовать алгоритм транспортной задачи для решения более сложных задач. 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 грузов из источника в сток через другие источники и стоки, то новая задача может быть сведена к обычной транспортной задаче. Объем вычислений, естественно, возрастает, так как в сеть будут включены дополнительные маршруты, соединяющие каждый источник со всеми другими источниками и каждый сток со всеми другими стоками. Считаем, что транспортные затраты c_{ij} , соответствующие дополнительным маршрутам, известны. Новая задача сводится к модифицированной транспортной задаче, в которой предложения и спрос, соответствующие дополнительным маршрутам, заданы таким образом, что они не влияют на выбор маршрутов, осуществляемый в основном алгоритме. Выполнение последнего требования необходимо, поскольку ограничения на поток по дополнительным маршрутам, задаваемые предложением и спросом, являются фиктивными и вводятся только для вычислительных целей.

Пусть d — минимальная из величин $\sum_i a_i$ и $\sum_j b_j$ — суммарных предложения и спроса, т. е. d — это реальный поток, протекающий 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 принять равными d , т. е. весь поток может протекать через один источник или через один сток. Поскольку $\bar{b}_i = \bar{a}_j = d$, то исходные величины предложения и спроса должны быть увеличены на d . Если же увеличить исходные данные на $\bar{d} < d$, то некоторые планы перевозок, допустимые в исходной задаче, в модифицированной задаче станут недопустимыми. Выбор $\bar{d} > d$ не скажется на решении задачи. Поэтому выбирают d .

Рассмотрим традиционную транспортную задачу (табл. 2.12).

Таблица 2.12

Исходная транспортная задача

	B_1	B_2	B_3	Запасы
A_1	3	4	7	20
A_2	6	3	2	10
Потребности	10	12	8	

Пусть в ней каждый источник и каждый сток может использоваться в качестве промежуточного пункта (узла). Здесь $\sum_{i=1}^2 a_i = \sum_{j=1}^3 b_j = 30$, следовательно, в модифицированной задаче надо брать $d = 30$.

Новая задача о перевозках может быть сведена к транспортной задаче со следующими величинами предложения и спроса для стоков и источников:

- 1) $\bar{a}_i = a_i + 30$ — для исходных источников, $i = \overline{1, 2}$;
- 2) $\bar{b}_i = 30$ — для исходных источников, $i = \overline{1, 2}$;
- 3) $\bar{a}_j = 30$ — для исходных стоков, $j = \overline{1, 3}$;
- 4) $\bar{b}_j = b_j + 30$ — для исходных стоков, $j = \overline{1, 3}$.

Матрица условий модифицированной транспортной задачи для решения задачи перевозок с перегрузкой приведена в табл. 2.13. Транспортные затраты c_{ij} для дополнительных маршрутов (три последних строки и два последних столбца в табл. 2.13) предполагаются известными.

Модифицированная транспортная задача

	Исходные истоки			Исходные источники		Запасы
	B_1	B_2	B_3	\bar{A}_1	\bar{A}_2	
A_1	3	4	7	0	5	$20+30=50$
A_2	6	3	2	3	0	$10+30=40$
\bar{B}_1	0	5	4	2	5	30
\bar{B}_2	9	0	1	3	2	30
\bar{B}_3	2	4	0	2	6	30
Потребности	$10+30=40$	$12+30=42$	$8+30=38$	30	30	

Эту задачу можно решить методом потенциалов. Решив модифицированную транспортную задачу, тем самым решим поставленную задачу о перевозках.

2.8. Целочисленное линейное программирование

Рассмотрим следующую задачу линейного программирования: максимизировать

$$f(x) = a_{00} - a_{01}x_1 - a_{02}x_2 - \dots - a_{0n}x_n$$

при условии

$$x_{n+1} = a_{n+1,0} - a_{n+1,1}x_1 - a_{n+1,2}x_2 - \dots - a_{n+1,n}x_n. \quad (2.5)$$

\vdots

$$x_{n+m} = a_{n+m,0} - a_{n+m,1}x_1 - a_{n+m,2}x_2 - \dots - a_{n+m,n}x_n.$$

$$x_j \geq 0 \quad (j = 1, \dots, n+1, \dots, n+m).$$

Заметим, что x_{n+1}, \dots, x_{n+m} — слабые переменные, а x_1, \dots, x_n — исходные переменные задачи (2.5). Если наряду с ограничениями (2.5) потребовать, чтобы все x_j ($j=1, \dots, m$) были целыми, то задача будет называться *задачей целочисленного программирования*. Существует большое число задач, особенно комбинаторных, которые можно сформулировать как задачи целочисленного программирования.

Ограничения (2.5) определяют выпуклую область $OABCD$ в n -мерном пространстве, как показано на рис. 2.6. Узлы целочисленной решетки на рис. 2.6 изображены точками. Такие точки, расположенные внутри области $OABCD$, являются допустимыми

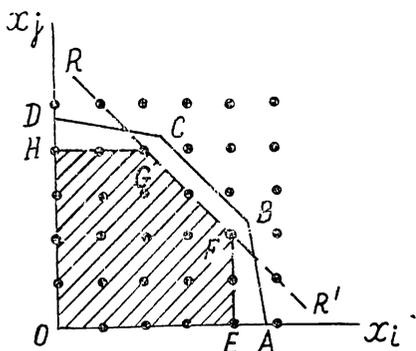


Рис. 2.6. Области допустимых решений для задач линейного и целочисленного линейного программирования

можно рассматривать как область допустимых решений некоторой другой задачи линейного программирования. Действительно, если к задаче линейного программирования, определяющей допустимую область $OABCD$, добавить ограничение типа RR' , как показано на рис. 2.6, то для вновь полученной задачи $OEF GH$ будет областью допустимых решений. Такая область обладает двумя важными свойствами: 1) содержит все допустимые целочисленные точки исходной задачи линейного программирования (поскольку является выпуклой оболочкой этих точек); 2) все крайние точки новой области — целочисленны. Поэтому любое базисное оптимальное решение модифицированной задачи линейного программирования имеет своими компонентами целые числа и является оптимальным решением исходной задачи целочисленного программирования.

Как только будут введены дополнительные ограничения, можно решать модифицированную задачу линейного программирования любым обычным методом, и полученное базисное оптимальное решение автоматически будет целочисленным. Представленный ниже целочисленный алгоритм обладает следующими свойствами: 1) все дополнительные ограничения сохраняют допустимые точки исходной целочисленной задачи; 2) за конечное число шагов создается достаточное число дополнительных ограничений для того, чтобы оптимальное решение модифицированной задачи было целочисленным; 3) дополнительные ограничения (гиперплоскости) проходят по крайней мере через одну целочисленную точку, хотя и не обязательно находящуюся внутри выпуклой оболочки; 4) каждое новое ограничение сокращает область допустимых решений исходной задачи целочисленного программирования. Следует подчеркнуть, что оптимальное решение исходной задачи может быть получено прежде, чем размер допустимой области будет сокращен до размеров выпуклой оболочки. К тому же, поскольку оптимальное целочисленное решение определяется пере-

решениями задачи целочисленного программирования. Оптимальные решения задачи линейного программирования всегда располагаются на границе области решений. В данном случае граничные точки не являются даже допустимыми решениями, поскольку ни одна из них не целочисленна. Предположим, что область допустимых решений сужена до выпуклой оболочки допустимых целых точек внутри допустимой области. На рис. 2.6 эта выпуклая оболочка показана затененной областью $OEF GH$, которую

сечением n гиперплоскостей, таких гиперплоскостей существует не более, чем это необходимо; некоторые из них могут соответствовать ограничениям исходной задачи.

Задачу целочисленного программирования также можно записать в виде табл. 2.14.

Т а б л и ц а 2.14

	1	$-x_1$. . .	$-x_n$
$f(x)$	a_{00}	a_{01}		a_{0n}
x_1	0	-1		
.	.			
.	.			
x_n	0			-1
x_{n+1}	$a_{n+1, 0}$	$a_{n+1, 1}$. . .	$a_{n+1, n}$
x_{n+m}	$a_{n+m, 0}$	$a_{n+m, 1}$.	$a_{n+m, n}$

Обычно в ограничения задачи (2.5) включают тривиальные соотношения $x_j = -(-x_j)$ ($j = \overline{1, n}$). Причины представления переменных в виде $(-x_1), (-x_2), \dots, (-x_n)$ — чисто исторические, но это стало практикой в целочисленном программировании. Будем использовать a_j ($j = \overline{0, n}$), для обозначения j -го столбца текущей таблицы и a_{ij} ($i = \overline{0, n+m}; j = \overline{0, n}$) — для обозначения элемента i -й строки и j -го столбца таблицы. Предполагается, что все a_{ij} в исходной таблице целые. Следовательно, все слабые переменные x_{n+1}, \dots, x_{n+m} должны быть также неотрицательными целыми числами.

Вначале задачу целочисленного программирования рассматривают как задачу линейного программирования и решают ее с помощью прямого или двойственного симплекс-метода. В конце работы алгоритма $a_{i0} \geq 0$ ($i = \overline{1, n+m}$) и $a_{0j} \geq 0$ ($j = \overline{1, n}$). Если $a_{i0} \geq 0$ и целые для всех i , то получено оптимальное решение целочисленной задачи. В этом случае решение получается сразу, без использования ограничений целочисленности. Если $a_{i0} \geq 0$, но не все целые, к ограничениям (2.5) добавляют еще одно. Новое ограничение записывают внизу таблицы так, чтобы задача перестала быть прямо допустимой, т. е. $a_{i0} < 0$ для $i = n+m+1$. Затем используют двойственный симплекс-метод с целью сделать все $a_{i0} \geq 0$. Если a_{i0} получаются нецелыми, в таблицу добавляют новые ограничения до тех пор, пока все a_{i0} ($i = \overline{1, n+m}$) не станут целыми и неотрицательными.

Если после введения дополнительного ограничения текущая таблица перестает быть прямо допустимой, то текущее решение, представляющее собой вершину многогранника решений, не удовлетворяет этому дополнительному ограничению. Другими словами, дополнительное ограничение отсекает часть пространства решений. Если дополнительные ограничения не отсекают ни одной целочисленной точки пространства решений исходной задачи, то, вполне вероятно, после введения достаточного числа дополнительных ограничений вершины суженного множества решений будут целочисленными. Тогда, используя симплекс-метод, можно найти оптимальное целочисленное решение. Трудность состоит в систематическом получении дополнительных ограничений и доказательстве конечности алгоритма.

Каждый раз после проведения итерации симплекс-метода происходит изменение множества небазисных переменных. Изменяется и таблица. Будем использовать t для обозначения t -таблицы. Изложим сам алгоритм.

Шаг 1. Решить задачу целочисленного программирования как задачу линейного программирования с помощью прямого или двойственного симплекс-метода. Если получено оптимальное решение задачи линейного программирования, то $a_{i0} \geq 0$ ($i = 1, m+n$) и $a_{0j} \geq 0$ ($j = \overline{1, n}$).

Шаг 2. Если a_{i0} — все целые, то задача решена и решение получено без использования дополнительных ограничений. В противном случае пусть a_{i0}^t — первая нецелочисленная компонента в a_{i0}^t . Тогда i -я строка называется *производящей*. Записать внизу таблицы уравнение

$$s = -f_{i0}^t - \sum_j f_{ij}^t (-x_j^t), \quad (2.6)$$

где $f_{i0} = a_{i0} - [a_{i0}]$, $f_{i0} \geq 0$, $f_{ij} = a_{ij} - [a_{ij}]$;
 $[\cdot]$ — ближайшее к числу \cdot целое.

Переменную s называют *слабой переменной Гомори*, а уравнение (2.6) — *отсечением Гомори*. Прodelать шаг (двойственного) симплекс-метода, используя в качестве ведущей строки отсечение Гомори (2.6). При этом таблица останется двойственно допустимой. Повторять до тех пор, пока все a_{i0} ($i = 1, n+m$) не станут целыми неотрицательными. Если a_{i0} на некотором шаге остается отрицательным, следующий шаг (двойственного) симплекс-метода производится без введения отсечения Гомори. (Если a_{00} становится отрицательным, нулевую строку не выбирают в качестве производящей. Если a_{00} становится нецелым, следует выбрать нулевую строку в качестве производящей).

В приведенном ниже числовом примере все дополнительные ограничения сохраняются на протяжении вычислений. Это сделано для того, чтобы показать, что эти дополнительные ограничения представляют собой неравенства. Причем, если эти неравенст-

ва выразить через исходные небазисные переменные, они будут иметь целые коэффициенты.

Если сохранять все строки, соответствующие слабым переменным Гомори, то эти слабые переменные могут стать базисными. Если слабая переменная Гомори вошла в базис с неотрицательным значением, то соответствующая строка представляет собой неравенство, справедливое при текущем решении, и эта строка может быть вычеркнута. Если слабая переменная Гомори становится базисной с отрицательным значением, соответствующую строку следует использовать в качестве ведущей. Если сохранять все строки, соответствующие всем отсечениям Гомори, то, вообще говоря, потребуется меньшее число дополнительных ограничений, однако увеличение таблицы предпочтительнее, чем введение лишних дополнительных ограничений.

Приведем пример, иллюстрирующий алгоритм.

Пример. Рассмотрим задачу целочисленного программирования:

максимизировать

$$f(x) = 4x_1 + 5x_2 + x_3$$

при условиях

$$3x_1 + 2x_2 \leq 10 ;$$

$$x_1 + 4x_2 \leq 11 ;$$

$$3x_1 + 3x_2 \leq 13 .$$

$$x_1, x_2, x_3 \geq 0 \quad (\text{целые}).$$

Вводя слабые переменные x_4, x_5, x_6 , получаем:

	1	$-x_1$	$-x_2$	$-x_3$
$f(x)$	0	-4	-5	-1
x_1	0	-1	0	0
x_2	0	0	-1	0
x_3	0	0	0	-1
x_4	10	3	2	0
x_5	11	1	4	0
x_6	13	3	3	1

Решаем задачу линейного программирования (ведущий элемент отмечен):

	1	$-x_1$	$-x_2$	$-x_3$
$f(x)$	55/4	-11/4	5/4	-1
x_1	0	-1	0	0
x_2	11/4	1/4	1/4	0
x_3	0	0	0	-1
x_4	18/4	<u>10/4</u>	-2/4	0
x_5	0	0	-1	0
x_6	19/4	9/4	-3/4	1

	1	$-x_4$	$-x_5$	$-x_3$
$f(x)$	187/10	11/10	-7/10	-1
x_1	18/10	4/10	-2/10	0
x_2	23/10	-1/10	3/10	0
x_3	0	0	0	-1
x_4	0	-1	0	0
x_5	0	0	-1	0
x_6	7/10	-9/10	-3/10	<u>1</u>

Ведущий столбец



	1	$-x_4$	$-x_5$	$-x_6$	
$f(x)$	194/10	2/10	4/10	1	Производящая строка
x_1	18/10	4/10	-2/10	0	
x_2	23/10	-1/10	3/10	0	
x_3	7/10	-9/10	-3/10	1	
x_4	0	-1	0	0	
x_5	0	0	-1	0	
x_6	0	0	0	-1	
s_1	-7/10	-1/10	<u>-7/10</u>	0	

Получено оптимальное решение задачи линейного программирования: $f(x) = 194/10$, $x_1 = 18/10$, $x_2 = 23/10$, $x_3 = 7/10$. Оно не целочисленное. Приступаем к шагу 2, дописываем в последней таблице уравнение отсечения и назначаем производящую строку и ведущий столбец:

	1	$-x_4$	$-s_1$	$-x_6$
$f(x)$	19	1,7	4/7	1
x_1	2	3/7	-2/7	0
x_2	2	-1/7	3/7	0
x_3	1	-6/7	-3/7	1
x_4	0	-1	0	0
x_5	1	1/7	-10/7	0
x_6	0	0	0	-1
s_1	0	0	-1	0

Откуда находим оптимальное целочисленное решение

$$f_{\max}(x) = 19, x_1 = 2, x_2 = 2, x_3 = 1.$$

Выразив x_4, x_5 и x_6 через исходные небазисные переменные x_1, x_2 и x_3 , получим неравенство $s_1 \geq 0$ с целыми коэффициентами:

$$-\frac{7}{10} + \frac{1}{10}(10 - 3x_1 - 2x_2) + \frac{7}{10}(11 - x_1 - 4x_2) \geq 0,$$

или $x_1 + 3x_2 \leq 8$. Чтобы получить матрицу, полностью целочисленную, просто продолжим введение отсечений:

Ведущий столбец

	1	$-x_4$	$-s_1$	$-x_6$
$f(x)$	19	1/7	4/7	1
x_1	2	3/7	-2/7	0
x_2	2	-1/7	3/7	0
x_3	1	-6/7	-3/7	1
x_4	0	-1	0	0
x_5	1	1/7	-10/7	0
x_6	0	0	0	-1
s_1	0	0	-1	0
s_2	0	-1/7	-4/7	0

Производящая строка

	1	$-s_2$	$-s_1$	$-x_6$
$f(x)$	19	1	0	1
x_1	2	3	-2	0
x_2	2	-1	1	0
x_3	1	-6	3	1
x_4	0	-7	4	0
x_5	1	1	-2	0
x_6	0	0	0	-1
s_1	0	0	-1	0
s_2	0	-1	0	0

2.9. Постановка задачи об оптимальном раскрое материалов (о минимизации отходов)

Пусть некоторый полуфабрикат (например, листы фанеры) поступил на предприятие в виде m различных партий, содержащих соответственно a_1, \dots, a_m единиц полуфабриката одинакового для каждой партии размера. Из поступивших полуфабрикатов требуется изготовить возможно большее число комплектов деталей, в каждый из которых входит k_1 деталей первого вида, k_2 деталей второго вида и т. д., k_l деталей l -го вида. Пусть каждую единицу полуфабриката можно раскроить на детали n различными способами, причем при раскрое единицы i -й партии j -м способом получается a_{ijs} деталей s -го вида.

Обозначим через x_{ij} число единиц из i -й партии полуфабрикатов, которые намечено раскроить j -м способом, так что из i -й партии при j -м способе раскроя будет получено $a_{ijs} x_{ij}$ деталей s -го вида. Всего же по плану из всей i -й партии деталей s -го вида будет получено $\sum_{j=1}^n a_{ijs} x_{ij}$, а из всех m партий их

будет получено $\sum_{i=1}^m \sum_{j=1}^n a_{ijs} x_{ij}$. Так как в каждый комплект гото-

вой продукции должно входить k_s -деталей s -го вида, то $\sum_{i=1}^m \sum_{j=1}^n a_{ijs} x_{ij}$ деталей позволит их использовать для составления

$$\frac{\sum_{i=1}^m \sum_{j=1}^n a_{ijs} x_{ij}}{k_s}$$

комплектов, и, таким образом, число полных комплектов, которое можно будет выпустить по данному плану $\|x_{ij}\|$, равно наименьшему из частных

$$\frac{\sum_{i=1}^m \sum_{j=1}^n a_{ij1} x_{ij}}{k_1}, \dots, \frac{\sum_{i=1}^m \sum_{j=1}^n a_{ijl} x_{ij}}{k_l}.$$

Введением дополнительной переменной ξ сведем рассматриваемую задачу к задаче максимизации $f(x) = \xi$ при ограничениях

$$\frac{\sum_{i=1}^m \sum_{j=1}^n a_{ijs} x_{ij}}{k_s} \geq \xi \quad (s = \overline{1, l}),$$

$$x_{i1} + \dots + x_{in} = a_i \quad (i = \overline{1, m}),$$

$\xi > 0$, $x_{ij} \geq 0$ ($i = \overline{1, m}$; $j = \overline{1, n}$) — целые числа.

Не всегда эту задачу решают как целочисленную. Если значения переменных в полученном оптимальном решении задачи линейного программирования достаточно велики, то приближенное целочисленное решение получают из него путем округления до ближайших допустимых целых чисел.

2.10. Задача о наилучшем использовании посевной площади

Пусть под посев n культур отведено m земельных массивов площадью соответственно в a_1, \dots, a_m гектаров, причем пусть средняя урожайность j -й культуры на i -м массиве составляет a_{ij} центнеров с гектара, а выручка за один центнер j -й культуры составляет p_j рублей.

Определить, какую площадь на каждом массиве следует отвести под каждую из культур, чтобы получить максимальную выручку, если по плану должно быть собрано не менее b_j центнеров j -й культуры ($j = \overline{1, n}$).

Обозначим через x_{ij} площадь, которую предполагается отвести под j -ю культуру на i -м массиве, так что

$$x_{i1} + \dots + x_{in} = a_i \quad (i = \overline{1, m}).$$

Ожидаемый средний урожай j -й культуры со всех массивов

$$a_{1j} x_{1j} + \dots + a_{mj} x_{mj}.$$

По плану он должен быть не менее b_j центнеров:

$$a_{1j}x_{1j} + \dots + a_{mj}x_{mj} \geq b_j \quad (j = \overline{1, n}),$$

Ожидаемая выручка за урожай j -й культуры

$$p_j(a_{1j}x_{1j} + \dots + a_{mj}x_{mj}),$$

а за урожай всех культур

$$p_1(a_{11}x_{11} + \dots + a_{m1}x_{m1}) + \dots + p_n(a_{1n}x_{1n} + \dots + a_{mn}x_{mn}).$$

Таким образом, задача заключается в максимизации

$$f(x) = p_1(a_{11}x_{11} + \dots + a_{m1}x_{m1}) + \dots + p_n(a_{1n}x_{1n} + \dots + a_{mn}x_{mn})$$

от $m \times n$ переменных $x_{11}, \dots, x_{ij}, \dots, x_{mn}$ при выполнении следующих ограничений:

$$\begin{aligned} x_{11} + \dots + x_{1n} &= a_1, \\ &\dots \dots \dots \\ x_{m1} + \dots + x_{mn} &= a_m, \\ a_{11}x_{11} + \dots + a_{m1}x_{m1} &\geq b_1, \\ &\dots \dots \dots \\ a_{1n}x_{1n} + \dots + a_{mn}x_{mn} &\geq b_n, \\ x_{ij} &\geq 0 \quad (i = \overline{1, m}; j = \overline{1, n}). \end{aligned}$$

Как обычно, задачу решают симплекс-методом.

Иногда задача об оптимальном распределении посевной площади выглядит так: имеющуюся посевную площадь распределить под посев n культур так, чтобы обеспечить максимальный урожай при соблюдении определенного соотношения $k_1 : k_2 : \dots : k_n$, в котором должны производиться эти культуры.

Введением дополнительной переменной ξ , такой, что

$$\frac{a_{1j}x_{1j} + \dots + a_{mj}x_{mj}}{k_j} \geq \xi \quad (j = \overline{1, n}),$$

рассматриваемая задача сводится к задаче линейного программирования — максимизировать

$$f(x) = \xi$$

при ограничениях

$$\frac{a_{1j}x_{1j} + \dots + a_{mj}x_{mj}}{k_j} \geq \xi \quad (j = \overline{1, n}).$$

$$x_{i1} + \dots + x_{in} = a_i \quad (i = \overline{1, m}),$$

$$\xi, x_{ij} \geq 0 \quad (i = \overline{1, m}; j = \overline{1, n}).$$

2.11. Задача о закреплении самолетов за воздушными линиями

Эта задача возникает при выборе оптимального варианта плана закрепления самолетов за данными воздушными линиями, обеспечивающего необходимые объемы перевозок при минималь-

ных суммарных эксплуатационных расходах. Она формулируется так.

Пусть имеется n различных типов самолетов, которые нужно распределить между m авиалиниями. Пусть месячный объем перевозок самолетом i -го типа на j -й авиалинии равен a_{ij} единицам, а связанные с этим месячные эксплуатационные расходы составляют b_{ij} рублей. Определить число x_{ij} самолетов i -го типа, которое следует закрепить за j -й авиалинией для обеспечения перевозки по этой линии a_j единиц ($i = \overline{1, n}$; $j = \overline{1, m}$) при минимальных суммарных эксплуатационных расходах, если известно, что имеется N_i самолетов i -го типа ($i = \overline{1, n}$).

Так как объем перевозок по j -й авиалинии

$$a_{1j}x_{1j} + \dots + a_{nj}x_{nj} \quad (j = \overline{1, m}),$$

а суммарные расходы составляют при этом

$$\sum_{j=1}^m \sum_{i=1}^n b_{ij}x_{ij},$$

то задача состоит в минимизации

$$f(x) = \sum_{j=1}^m \sum_{i=1}^n b_{ij}x_{ij}$$

при ограничениях

$$a_{1j}x_{1j} + \dots + a_{nj}x_{nj} \geq a_j \quad (j = \overline{1, m}),$$

$$x_{i1} + \dots + x_{im} = N_i \quad (i = \overline{1, n}),$$

$$x_{ij} \geq 0 \quad (i = \overline{1, n}; j = \overline{1, m}).$$

и решается симплекс-методом с учетом целочисленности x_{ij} .

Пример. Пусть три типа самолетов следует распределить между четырьмя авиалиниями. В приводимой ниже таблице задано число самолетов каждого типа, месячный объем перевозок каждым самолетом на каждой авиалинии и соответствующие эксплуатационные расходы.

Тип самолета	Число самолетов	Месячный объем перевозок одним самолетом по авиалиниям				Эксплуатационные расходы на один самолет по авиалиниям			
		I	II	III	IV	I	II	III	IV
1	50	15	10	20	50	15	20	25	40
2	20	30	25	10	17	70	28	15	45
3	30	25	50	30	45	40	70	40	65

Надо распределить самолеты по авиалиниям так, чтобы при минимальных суммарных эксплуатационных расходах перевезти по каждой из четырех авиалиний соответственно не менее 300, 200, 1000 и 500 единиц груза.

Обозначим через x_{ij} число самолетов i -го типа ($i = \overline{1,3}$), которое планируется закрепить за j -й ($j = \overline{1,4}$) авиалинией. Тогда задача сводится к минимизации

$$f(x) = 15x_{11} + 20x_{12} + 25x_{13} + 40x_{14} + 70x_{21} + 28x_{22} + \\ + 15x_{23} + 45x_{24} + 40x_{31} + 70x_{32} + 40x_{33} + 65x_{34}$$

при ограничениях

$$\begin{aligned} 15x_{11} + 30x_{21} + 25x_{31} &\geq 300, \\ 10x_{12} + 25x_{22} + 50x_{32} &\geq 200, \\ 20x_{13} + 10x_{23} + 30x_{33} &\geq 1000, \\ 50x_{14} + 17x_{24} + 45x_{34} &\geq 500, \\ x_{11} + x_{12} + x_{13} + x_{14} &= 50, \\ x_{21} + x_{22} + x_{23} + x_{24} &= 20, \\ x_{31} + x_{32} + x_{33} + x_{34} &= 30, \\ x_{ij} &\geq 0 \quad (i = \overline{1,3}; j = \overline{1,4}). \end{aligned}$$

Перепишем ограничения в виде

$$\begin{aligned} y_1 &= 15x_{11} + 30x_{21} + 25x_{31} - 300 \geq 0 \\ y_2 &= 10x_{12} + 25x_{22} + 50x_{32} - 200 \geq 0, \\ y_3 &= 20x_{13} + 10x_{23} + 30x_{33} - 1000 \geq 0, \\ y_4 &= 50x_{14} + 17x_{24} + 45x_{34} - 500 \geq 0, \\ 0 &= -x_{11} - x_{12} - x_{13} - x_{14} + 50, \\ 0 &= -x_{21} - x_{22} - x_{23} - x_{24} + 20, \\ 0 &= -x_{31} - x_{32} - x_{33} - x_{34} + 30, \\ x_{ij} &\geq 0 \quad (i = \overline{1,3}; j = \overline{1,4}) \end{aligned}$$

Составив таблицу

	$-x_{11}$	$-x_{12}$	$-x_{13}$	$-x_{14}$	$-x_{21}$	$-x_{22}$	$-x_{23}$	$-x_{24}$	$-x_{31}$	$-x_{32}$	$-x_{33}$	$-x_{34}$	1
y_1	-15	0	0	0	-30	0	0	0	-25	0	0	0	-300
y_2	0	-10	0	0	0	-25	0	0	0	-50	0	0	-200
y_3	0	0	-20	0	0	0	-10	0	0	0	-30	0	-1000
y_4	0	0	0	-50	0	0	0	-17	0	0	0	0	-45
0	1	1	1	1	0	0	0	0	0	0	0	0	50
0	0	0	0	0	1	1	1	1	0	0	0	0	20
0	0	0	0	0	0	0	0	0	1	1	1	1	30
$f(x)$	-15	-20	-25	-40	-70	-28	-15	-45	-40	-70	-40	-65	0

и освободившись от 0-уравнений, т. е. переведя последние три переменные в свободные и вычеркнув столбцы, соответствующие этим свободным переменным, получим преобразованную таблицу, в которой еще есть отрицательные свободные члены.

	$-x_{12}$	$-x_{13}$	$-x_{14}$	$-x_{21}$	$-x_{22}$	$-x_{24}$	$-x_{31}$	$-x_{32}$	$-x_{34}$	1
y_1	15	15	15	-30	0	0	-25	0	0	450
y_2	$\overline{-10}$	0	0	0	-25	0	0	-50	0	-200
y_3	0	-20	0	10	10	10	30	30	-30	100
y_4	0	0	-50	0	0	-17	0	0	45	-500
x_{11}	1	1	1	0	0	0	0	0	0	50
x_{23}	0	0	0	1	1	1	0	0	0	20
x_{33}	0	0	0	0	0	0	1	1	1	30
$f(x)$	-5	-10	-25	-55	-13	-30	0	-30	-25	2250

Переходим к отысканию опорного решения. Сделав два шага модифицированных жордановых исключений, приходим к таблице

	$-y_2$	$-x_{13}$	$-y_4$	$-x_{21}$	$-x_{22}$	$-x_{24}$	$-x_{31}$	x_{32}	$-x_{34}$	1
y_1										0
x_{12}										20
y_3										100
x_{14}										10
x_{11}										20
x_{23}										20
x_{33}										30
$f(x)$	$-\frac{1}{2}$	-10	$-\frac{1}{2}$	-55	$-\frac{1}{2}$	$-\frac{43}{2}$	0	-5	$-\frac{5}{2}$	2600

откуда видно, что решение

$$x_{13}=x_{21}+x_{22}=x_{24}=x_{31}=x_{32}=x_{34}=0;$$

$$x_{11}=20, x_{12}=20, x_{14}=10, x_{23}=20, x_{33}=30,$$

для которого

$$f_{\min}(x) = 2600,$$

является не только опорным, но и оптимальным (все коэффициенты $f(x)$ -строки неположительны!).

2.12. Задача о назначениях (проблема выбора)

Будем рассматривать задачу о распределении n механизмов (работников) на n работ таким образом, чтобы каждый механизм (работник) выполнял только одну работу и чтобы при заданной производительности каждого механизма на каждой из работ суммарный эффект был максимальным.

Обозначим через c_{ij} ($i, j=1, n$) производительность i -го механизма на j -й работе. Тогда данная задача, известная под названием задачи о назначениях, будет заключаться в таком выборе n элементов из матрицы $\|c_{ij}\|$ по одному из каждой строки и каждого столбца, чтобы их сумма $c_{1j_1} + c_{2j_2} + \dots + c_{nj_n}$ ($j_k \neq j_l$) при $k \neq l$ была максимальной.

Обозначив через x_{ij} переменную, равную единице (если j -й механизм назначен на j -ю работу) и равную нулю (если он на эту работу не назначен), мы сведем задачу к следующей задаче линейного программирования.

Среди неотрицательных целочисленных решений системы $2n$ уравнений

$$x_{i1} + x_{i2} + \dots + x_{in} = 1 \quad (i = \overline{1, n}),$$

$$x_{1j} + x_{2j} + \dots + x_{nj} = 1 \quad (j = \overline{1, n}),$$

означающих, что каждый механизм выполняет одну работу и что каждая работа обеспечивается одним механизмом, найти то, которое максимизирует

$$f(x) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij},$$

означающее суммарную производительность всех механизмов. Эту задачу решают с помощью сетевых методов, менее громоздких, чем симплекс-метод (см. гл. 3).

Пример. Имеются три механизма A_1, A_2, A_3 , каждый из которых может быть использован на каждом из трех видов работ B_1, B_2, B_3 с производительностью (в условных единицах), заданной в виде таблицы

	B_1	B_2	B_3
A_1	1	2	3
A_2	2	4	1
A_3	3	1	5

Требуется так распределить механизм по одному на каждую из работ, чтобы суммарная производительность всех механизмов была максимальной.

Обозначим, как рекомендовалось выше, через x_{ij} переменную, равную единице, если механизм A_i назначен на работу B_j , и равную нулю, если механизм A_i не назначен на работу B_j . Тогда суммарная производительность механизмов

$$f(x) = x_{11} + 2x_{12} + 3x_{13} + 2x_{21} + 4x_{22} + x_{23} + 3x_{31} + x_{32} + 5x_{33},$$

ограничения

$$x_{11} + x_{12} + x_{13} = 1,$$

$$x_{21} + x_{22} + x_{23} = 1,$$

$$x_{31} + x_{32} + x_{33} = 1,$$

$$x_{11} + x_{21} + x_{31} = 1,$$

$$x_{12} + x_{22} + x_{32} = 1,$$

$$x_{13} + x_{23} + x_{33} = 1.$$

Надо максимизировать $f(x)$ при этих ограничениях.

Перепишав ограничения задачи в виде 0-уравнений, составим таблицу

	$-x_{11}$	$-x_{12}$	$-x_{13}$	$-x_{21}$	$-x_{22}$	$-x_{23}$	$-x_{31}$	$-x_{32}$	$-x_{33}$	1
0	$\boxed{1}$	1	1	0	0	0	0	0	0	1
0	0	0	0	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1	1	1	1
0	1	0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	0	1
0	0	0	1	0	0	1	0	0	1	1
$f(x)$	-1	-2	-3	-2	-4	-1	-3	-1	-5	0

После избавления от нулевых строк и вычеркивания столбцов, расположенных под перенесенными наверх нулями, получим таблицу

	$-x_{22}$	$-x_{23}$	$-x_{32}$	$-x_{33}$	1
x_{11}	$\boxed{-1}$	-1	-1	-1	-1
x_{21}	1	1	0	0	1
x_{31}	0	0	1	1	1
x_{12}	1	0	1	0	1
x_{13}	0	1	0	1	1
$f(x)$	1	3	3	0	9

Среди свободных членов таблицы есть отрицательный, так что опорное решение еще не получено.

Сделав шаг метода исключения Жордана (разрешающий элемент -1 отмечен), получим таблицу

	$-x_{11}$	$-x_{23}$	$-x_{32}$	$-x_{33}$	1
x_{22}	-1	1	1	1	1
x_{21}	$\boxed{1}$	0	-1	-1	0
x_{31}	0	0	1	1	1
x_{12}	1	-1	0	-1	0
x_{13}	0	1	0	1	1
$f(x)$	-1	4	4	1	10

В ней все свободные члены неотрицательны, так что можно перейти к отысканию оптимального решения. Выбрав разрешающий элемент, как указано в таблице, и сделав один шаг модифицированного жорданова исключения, получим окончательно

	$-x_{21}$	$-x_{23}$	$-x_{32}$	$-x_{33}$	1
x_{22}					1
x_{11}					0
x_{31}					1
x_{12}					0
x_{13}					1
$f(x)$	1	4	3	0	10

Из последней таблицы следует, что максимальная суммарная производительность всех механизмов равна 10 (условным единицам) и достигается при

$$x_{11}=0, x_{12}=0, x_{13}=1, x_{21}=0, x_{22}=1, \\ x_{23}=0, x_{31}=1, x_{32}=0, x_{33}=0,$$

т. е. при следующих назначениях:

$$\begin{array}{lcl} \text{механизма } A_1 & \text{на работу } B_3, \\ \text{» } A_2 & \text{» } B_2 \\ \text{» } A_3 & \text{» } B_1. \end{array}$$

Так как в $f(x)$ -строке последней таблицы есть нулевой элемент, то полученное оптимальное решение задачи не единственное. Сделав шаг модифицированного жорданова исключения с разрешающим элементом из столбца, содержащего нуль в $f(x)$ -строке, можно найти другое оптимальное решение этой же задачи.

2.13. Задача об оптимальном распределении самолетов между войсками и учебными полигонами

Рассмотрим задачу о наилучшем распределении выпускаемых промышленностью самолетов между войсками и учебными полигонами, на которых обучаются экипажи, для обеспечения лучшей боеготовности противовоздушной обороны.

Пусть некоторая воинская часть, участвующая в боевых вылетах и подготавливающая, кроме того, на своем учебном полигоне экипажи (летчиков), снабжается на протяжении n месяцев самолетами по a_j единиц ($j=\overline{1, n}$) в начале каждого месяца. Из них часть x_j , для которой имеются в наличии подготовленные экипажи, направляется сразу в войска, а оставшаяся часть $a_j - x_j$ — на полигоны для обучения новых экипажей. Боеготовность части определяется числом самолетов с подготовленными экипажами и временем их пребывания в войсках, так называемым числом активных самолето-месяцев.

Возникает задача такого распределения получаемых ежемесячно самолетов между войсками и учебными полигонами, при котором будет достигнута максимальная боеготовность части.

Пусть до начала рассматриваемой программы на полигоне имеется z_0 самолетов. Число самолетов z_j , используемых на полигоне для обучения в j -месяце, будет состоять из z_{j-1} самолетов, имевшихся в предыдущем месяце, плюс $a_j - x_j$ самолетов, полученных в начале j -го месяца:

$$z_j = z_{j-1} + (a_j - x_j) \quad (j = \overline{1, n}).$$

Пусть к началу программы имеется y_1 обученных экипажей, но не обеспеченных самолетами, и в течение месяца на одном самолете обучается k экипажей. Число экипажей y_j , обученных к началу j -го месяца, будет состоять из числа y_{j-1} обученных эки-

пажей, имевшихся к началу предыдущего месяца, и числа kz_{j-1} экипажей, обученных за этот $(j-1)$ месяц:

$$y_j = y_{j-1} + kz_{j-1} \quad (j = \overline{2, n}).$$

Общее число самолетов, направленных в войска к началу j -го месяца, не должно превышать обученных к этому моменту экипажей, так что

$$x_1 + \dots + x_j \leq y_j \quad (j = \overline{1, n}).$$

К концу периода, на который рассчитана программа, самолеты, поступившие в войска в первый месяц, обеспечат nx_1 активных самолето-месяцев. Самолеты, поступившие во второй месяц, обеспечат $(n-1)x_2$ активных самолето-месяцев и т. д. Самолеты, поступившие в течение всех n месяцев, обеспечат боеготовность в количестве

$$nx_1 + (n-1)x_2 + \dots + [n-(j-1)]x_j + \dots + 2x_{n-1} + x_n$$

активных самолето-месяцев.

Задача заключается в максимизации

$$f(x) = nx_1 + (n-1)x_2 + \dots + 2x_{n-1} + x_n$$

при ограничениях

$$x_j + z_j - z_{j-1} - a_j = 0 \quad (j = \overline{1, n});$$

$$y_j - y_{j-1} - kz_{j-1} = 0 \quad (j = \overline{2, n});$$

$$y_1 - x_1 - x_2 - \dots - x_j \geq 0 \quad (j = \overline{1, n});$$

$$x_j \geq 0, \quad y_j \geq 0, \quad z_j \geq 0 \quad (j = \overline{1, n}).$$

Замечание. К подобной математической модели сводится задача о распределении поступающего в ограниченном количестве сырья (например, металла), часть которого идет на изготовление некоторого изделия, а часть для выпуска оборудования, используемого для производства этого изделия, если конечной целью является максимальный выпуск рассматриваемого изделия.

2.14. Задача о рациональном соотношении между различными типами броневых снарядов

Пусть известно, что противник располагает m видами танковой брони, но неизвестно, в каком соотношении он ее использует для производства своих танков. Пусть, кроме того, имеется n видов броневых снарядов и известен закон поражения каждого вида брони каждым из видов снарядов, т. е. известна вероятность p_{ij} поражения танка с i -й броней снарядом j -го вида ($i = \overline{1, m}$; $j = \overline{1, n}$).

Требуется определить, в каком соотношении надо брать разные виды снарядов, чтобы, смешав их и использовав случайным образом, обеспечить максимальное значение математического

2.15. Задачи о покрытии множества

Одним из вариантов задачи размещения производства является задача о покрытии множества, т. е. задача определения мест размещения и их числа. Например, задача размещения складов, при котором расстояние от склада до каждого потребителя не превышает 100 км, может быть сформулирована как задача о покрытии множества. Другим примером задачи о покрытии множества является задача определения числа и размещения на территории города студенческих общежитий, при котором каждый студент тратит на дорогу до учебного заведения не более одного часа, или задача размещения пожарных команд, при котором расстояние до любой точки города покрывается за 5 мин.

Задача о покрытии множества может быть сформулирована следующим образом:
найти

$$\min f(x) = \sum_{j=1}^n c_j x_j$$

при ограничении

$$\sum_{j=1}^n a_{ij} x_j \geq 1, \quad i = \overline{1, m}, \quad (2.7)$$

$$x_j = (0; 1), \quad j = \overline{1, n}.$$

Величины a_{ij} называются *коэффициентами покрытия* и принимают значения, равные единице, если потребитель находится в пределах j -й области (т. е. покрывается j -й областью), в противном случае a_{ij} равны нулю. Аналогично x_j принимает значение, равное единице, если в j -й области расположен некоторый объект, и равное нулю в противном случае. Ограничения в задаче требуют, чтобы каждый из m потребителей был «покрыт» по крайней мере одним из n объектов. Цель в этом случае состоит в том, чтобы «покрыть» потребителей с минимальными затратами, причем c_j — стоимость помещения объекта в j -ю область.

Поясним смысл термина «покрывать». Если имеется ряд жилых строений и решается задача размещения пожарных команд, то i -е жилое строение считается покрытым, если пожарная команда находится в пределах пяти минут езды от этого строения; аналогично, если существует i -й потребитель и речь идет о размещении заводов, один из которых должен удовлетворять потребности этого потребителя, то последний считается «покрытым» при условии, что завод расположен, например, в местах 1, 2 или 3. Таким образом, $a_{11} = a_{12} = a_{13} = 1$ и все остальные a_{ij} равны нулю для $j \neq \overline{1, 3}$.

Поскольку (2.7) является и задачей целочисленного линейного программирования, то любой приемлемый для ее решения метод может быть использован для решения задачи (2.7). Однако

из-за особой структуры задачи (2.7) специально для ее решения был разработан ряд методов: *неявного перебора, секущей плоскости, отсечения, эвристические*.

Обычно задача о покрытии множества при решении проблемы размещения состоит в определении минимального числа объектов, необходимых для удовлетворения (покрытия) потребностей некоторого множества потребителей. В подобной ситуации (2.7) сводится к так называемой задаче о *полном покрытии*, которая получается из задачи (2.7), если положить все c_{ij} равными единице. Ряд задач размещения экстренных служб может быть сформулирован как задачи о полном покрытии.

Помимо задачи о полном покрытии возможна постановка задачи о *частичном покрытии*. Если задача о полном покрытии состоит в определении мест расположения объектов и их минимального числа, при котором удовлетворяются все потребители, то задача о частичном покрытии связана с определением размещения заданного числа объектов, при котором удовлетворяется максимальное число потребителей. Практически не всегда можно обеспечить такое число объектов, которое полностью удовлетворяло бы всех потребителей. Обычно число имеющихся в распоряжении объектов достаточно только для частичного «покрытия» множества потребностей. В таких ситуациях целесообразно задачу размещения свести к задаче о частичном покрытии.

Математическая задача о частичном покрытии может быть сформулирована следующим образом:

найти

$$\max \tilde{f}(x) = \sum_{i=1}^m \max a_{ij} x_j, \quad 1 \leq j \leq n,$$

при ограничениях

$$\sum_{j=1}^n x_j \leq K, \quad x_j = (0; 1), \quad j = \overline{1, n}, \quad (2.8)$$

где K — максимальное число объектов, подлежащих размещению, величины a_{ij} и x_j те же, что и в задаче (2.7).

Из вида функционала $\max a_{ij} x_j$, используемого в целевой функции в задаче (2.8), следует, что если некоторое место расположения потребителя покрывается более чем одним размещаемым объектом, то при вычислении $\tilde{f}(x)$ учитывается только максимальная величина a_{ij} . Ограничение в задаче (2.8) показывает, что в лучшем случае имеется K объектов для размещения. Заметим, если $\tilde{f}(x)$ равно m — числу потребителей, то это значит, что K достаточно велико, чтобы полностью удовлетворить всех потребителей. Таким образом, задача о полном покрытии может быть сведена к задаче о частичном покрытии (2.8) для различных значений K , и решение задачи (2.8) при наименьшем значении K , для которо-

го $f(x) = m$, будет оптимальным решением задачи о полном покрытии.

Однако обычно задачу о полном покрытии не решают таким образом, так как существуют более эффективные методы ее решения.

Точное решение задачи (2.8) может быть получено с помощью методов динамического программирования, метода ветвей и границ и двойственных методов. Однако они не приемлемы с вычислительной точки зрения при $n \geq 20$ и $K \geq 10$, и поэтому для решения задачи (2.8) был разработан ряд эвристических методов.

Глава 3. СЕТЕВЫЕ (ПОТОКОВЫЕ) ЗАДАЧИ

3.1. Основные определения и приложения потоковых моделей

Многие задачи линейного программирования могут быть сформулированы как *сетевые*. Примером таких задач является транспортная задача (рис. 1.2). Хотя можно указать и такие сетевые задачи, которые нельзя сформулировать в виде задачи линейного программирования. Например, таковой является *задача коммивояжера*.

Вследствие специальной структуры сетевых задач для них получено большое число эффективных алгоритмов и изящных теорем, обеспечивающих решение широкого круга практических задач. В большинстве сетевых задач оптимальные решения целочисленны, чего нельзя сказать о решении общей задачи линейного программирования.

Основные определения в сетевых (потоковых) задачах. Введем основные определения и понятия. *Сеть* состоит из множества *узлов* (называемых также *вершинами*, или *точками соединения*) и множества *дуг* (называемых также *звеньями*, или *ребрами*), которые связывают эти *узлы*. Если дуга имеет определенную ориентацию (направление), то ее называют *ориентированной*, или *направленной*.

Сеть называют *связной*, если при любом разбиении множества узлов сети на подмножества X и \bar{X} найдется дуга A_{ij} или дуга A_{ji} , когда i -й узел N_i принадлежит подмножеству X , $N_i \in X$, а j -й узел N_j принадлежит подмножеству \bar{X} , $N_j \in \bar{X}$.

Мы будем рассматривать только *связные сети* и будем считать, что между любыми двумя узлами N_i и N_j имеется не больше одной ориентированной дуги A_{ij} и одной ориентированной дуги A_{ji} , либо имеется только одна неориентированная дуга A_{ij} . *Одну неориентированную дугу можно заменить двумя ориентированными*. Существование *петель* (дуг, ведущих из некоторого узла в тот же узел) исключается.

Последовательность узлов и дуг сети $N_1, A_{12}, N_2, A_{23}, \dots, N_{k-1}, A_{k-1, k}, N_k$ называют *цепью* (ориентированной цепью), ведущей

из узла N_1 в узел N_k . Если $N_1 = N_k$, то такую последовательность называют *ориентированным циклом*. Цепь называют простой, если она не содержит циклов.

Путем будем называть последовательность $N_1, A_{12}, N_2, \dots, N_{k-1}, A_{k-1, k}, N_k$, где N_1, N_2, \dots, N_k — узлы сети и либо $A_{i, i+1}$, либо $A_{i+1, i}$ ($i = \overline{1, k-1}$) — дуга сети. Путь отличается от цепи тем, что при движении по пути от N_1 до N_k можно пройти дугу сети и в направлении, противоположном ее ориентации. Для неориентированных сетей понятия цепи и пути совпадают. В частности, *циклами* являются *контурь*.

Контуром называют конечную цепь, начальный и конечный узлы которой совпадают. Очевидно, что циклы являются *замкнутыми путями*, а контурь *замкнутыми цепями*. *Вырожденный* цикл называют *петлей*. Петля образуется одним узлом и одной дугой и поэтому является как контуром, так и циклом.

В *связной* сети для любых двух различных узлов существует по крайней мере один соединяющий их путь или цепь. Частным случаем связных ориентированных и неориентированных сетей являются *деревья*. Если множество всех узлов и дуг задать в виде графа $G = (N, A)$, где N — множество узлов, A — множество дуг, то дерево определяют как связное подмножество (*подграф*) G множества (графа) G , не содержащее циклов, т. е. для любых двух узлов дерева существует единственный путь, соединяющий их. В сети, содержащей n узлов, подграф из k узлов ($k \leq n$) 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 связующим деревом (остовом) называют дерево, содержащее все узлы сети. Если сеть содержит n узлов, дерево с n узлами и с $n-1$ дугами является *остовом*. *Кратчайшим (максимальным) остовом графа (сети)* называют дерево с минимальным (максимальным) весом среди всех связующих деревьев этого графа. *Вес дерева* определяют как сумму весов (длин) его дуг. *Весом (длиной) дуги* называют число, соответствующее некоторой характеристике дуги (расстояние, стоимость и т. п.). Каждой дуге A_{ij} ставят в соответствие положительное число b_{ij} , называемое *пропускной способностью дуги (ребра)*.

В сети выделяют два специальных узла: один из них называют *источником* N_s , а другой — *стоком* N_t .

Сеть можно рассматривать как водопроводную систему, в которой трубы соответствуют дугам, источник воды — источнику N_s , сток воды — стоку N_t , а соединения между трубами — остальным узлам сети. В качестве пропускной способности выступает поперечное сечение трубы.

Полезные аналоги могут быть составлены и с электрическими цепями.

Потоком из источника N_s в сток N_t в сети называют множество неотрицательных чисел x_{ij} , поставленных в соответствие некоторой дуге сети, если эти числа удовлетворяют следующим линейным ограничениям:

$$\sum_i x_{ij} - \sum_k x_{jk} = \begin{cases} -v, & \text{если } j = s; \\ 0, & \text{если } j \neq s, j \neq t; \\ v, & \text{если } j = t \end{cases} \quad (3.1)$$

$$v \geq 0; 0 \leq x_{ij} \leq b_{ij} \text{ для всех } i, j. \quad (3.2)$$

Здесь первая сумма берется по дугам, ведущим в узел N_j , а вторая сумма — по дугам, ведущим из узла N_j . Неотрицательное число v называют *величиной потока*. Число x_{ij} называют *поток* по дуге A_{ij} , или *дуговым потоком*. Ограничения (3.1) выражают тот факт, что в каждый узел (кроме источника и стока) приходит столько потока, сколько из него уходит (условие сохранения потока). Ограничение (3.2) означает, что поток x_{ij} по дуге ограничен пропускной способностью дуги b_{ij} .

Очевидно, что задача нахождения величины максимального потока в любой сети является задачей линейного программирования: максимизировать $v = \sum_j x_{sj}$ при ограничениях (3.1)—(3.2). Но в силу специфики задачи сетевые методы решения здесь оказываются более эффективными, чем симплекс-метод для общей задачи линейного программирования.

Если сеть является цепью $N_1, A_{12}, N_2, \dots, N_k$ с источником N_1 и стоком N_k , максимальная величина потока, который может быть пропущен через сеть, ограничивается минимальной из пропускных способностей дуг этой сети. Дуга с минимальной пропускной способностью является «узким местом» в сети. В произвольной сети «узкое место» определяют *разрезом*. Пусть X — некоторое подмножество узлов сети, \bar{X} — дополнение подмножества X (объединение X и \bar{X} определяет множество узлов сети). *Разрезом* (X, \bar{X}) называют множество всех дуг A_{ij} , для которых $N_i \in X; N_j \in \bar{X}$. Таким образом, разрез представляет собой множество дуг, удаление которых из сети превращает сеть в несвязанную. Разрез (X, \bar{X}) называют *разделяющим узлы* N_s и N_t (или *отделяющим узел* N_s от N_t), если $N_s \in X; N_t \in \bar{X}$. Пропускной способностью $c(X, \bar{X})$, или *величиной разреза*, называют сумму $\sum_{ij} b_{ij}$, которую берут по всем ориентированным дугам, соединяющим $N_i \in X$ и $N_j \in \bar{X}$. При определении разреза учитывают все дуги между подмножеством X и подмножеством \bar{X} , а при определении пропускной способности разреза — только пропускные способности дуг из X в \bar{X} , ориентированные дуги из X в \bar{X} не учитывают. Поэтому в общем случае $c(X, \bar{X}) \neq c(\bar{X}, X)$.

Ясно, что из-за ограничений (3.1)—(3.2) максимальный поток меньше или равен пропускной способности любого разреза, разделяющего N_s и N_t :

$$v = \sum_{N_i \in \alpha, N_j \in \bar{\alpha}} x_{ij} - \sum_{N_i \in \alpha, N_j \in \bar{\alpha}} x_{ij}.$$

В любой сети величина максимального потока из источника N_s в сток N_t всегда равна минимальной пропускной способности всех разрезов, разделяющих N_s и N_t . Разрез, разделяющий N_s и N_t и обладающий минимальной пропускной способностью, называют *минимальным разрезом*. Эти утверждения объединены в теореме о максимальном потоке и минимальном разрезе.

Обозначим через F_{st} множество неотрицательных чисел x_{ij} , удовлетворяющих ограничениям (3.1)—(3.2). Будем называть путь из N_s в N_t увеличивающим поток F_{st} , если $x_{ij} < b_{ij}$ на всех прямых дугах и $x_{ij} > 0$ на всех обратных дугах этого пути. Поток F_{st} максимален тогда и только тогда, когда не существует пути, увеличивающего поток F_{st} .

Величина максимального потока в любой сети принимает, безусловно, единственное значение. Но может существовать несколько различных максимальных потоков F_{st} , имеющих одинаковые значения. Может существовать и несколько минимальных разрезов в сети.

Итак, всякая задача о потоке в сети может быть сформулирована как задача линейного программирования, но общая задача линейного программирования не всегда имеет целочисленное решение. Здесь нас будет интересовать подкласс тех задач линейного программирования, которые обладают целочисленным оптимальным решением. Задача линейного программирования с ограничениями

$$\sum_{j=1}^n a_{ij} x_j \leq b_i; \quad i = \overline{1, m}; \quad x_j \geq 0; \quad j = \overline{1, n},$$

всегда имеет целочисленное оптимальное решение при любом целочисленном векторе ограничений $\mathbf{b} = \{b_1, b_2, \dots, b_m\}^T$, если матрица \mathbf{A} (прямоугольная таблица чисел)

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

является *абсолютно унимодулярной*.

Матрицу \mathbf{A} называют *абсолютно унимодулярной*, если все ее миноры равны либо 0, либо ± 1 . (Минором k -го порядка $k \leq m$, $k \leq n$, для матрицы \mathbf{A} является определитель k -го порядка, построенный из элементов, стоящих на пересечении произвольных k столбцов и k строк матрицы.)

Целочисленность оптимального решения означает, что выпуклый многогранник, определяемый ограничениями

$$\sum_{j=1}^n a_{ij}x_j \leq b_i, i = \overline{1, m}; x_j \geq 0, j = \overline{1, n},$$

имеет целочисленные крайние точки при любом целочисленном векторе \mathbf{b} , если матрица \mathbf{A} абсолютно унимодулярна. Оказывается, что условие абсолютной унимодулярности матрицы \mathbf{A} является не только необходимым, но и достаточным для целочисленного оптимального решения.

Проверить унимодулярность матрицы \mathbf{A} , вычислив всевозможные миноры, достаточно сложно. Существуют достаточные, но не необходимые условия абсолютной унимодулярности матрицы \mathbf{A} , которые проверить гораздо легче. Матрица \mathbf{A} абсолютно унимодулярна, если:

- 1) каждый ее элемент равен 0, +1, -1;
- 2) каждый ее столбец содержит не более двух ненулевых элементов;
- 3) строки матрицы \mathbf{A} можно разбить на два непересекающихся множества R_1 и R_2 таким образом, что
 - а) если столбец из \mathbf{A} содержит два ненулевых элемента одного знака, то один из них входит в R_1 , другой — в R_2 ;
 - б) если столбец из \mathbf{A} содержит два ненулевых элемента с противоположными знаками, то оба они входят либо в R_1 , либо в R_2 .

Приложение потоковых моделей. В приложениях часто встречаются задачи, которые могут быть сформулированы в виде задач о *кратчайшей цепи*, о *потоке минимальной стоимости*, о *максимальном потоке* и т. п. Например, задача о *кратчайшей цепи* заключается в следующем. Заданы множества дуг и узлов. 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 из s в t . Примером подобной задачи может быть задача о замене устаревшего оборудования на новое. Здесь минимизируются общие затраты на закупку и обслуживание оборудования, если 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. Обычно задают и величину планируемого периода.

Значительное место в приложениях занимают сетевые задачи, связанные с планированием и составлением расписания выполнения работ по осуществлению больших проектов, по проведению научных исследований и опытных, конструкторских разработок, а также при составлении расписаний движения транспорта, календарного планирования, распределения ресурсов и т. д.

В более общей интерпретации дугу сети представляют как звено в механизме, предназначенном для транспортировки потока, который может протекать по дуге в единицу времени, что определяет пропускную способность дуги.

Многие обобщения задачи о максимальном потоке по существу сводят к поиску максимального потока в сети, к легко осуществимому поиску некоторых цепей сети. Примером одной из таких задач является *задача о потоке в сети с несколькими источниками и стоками*, когда заданы мощности источников и возможности (спрос) стоков. В данной задаче множество всех узлов разбивается на источники S , промежуточные узлы R и стоки T . Каждому узлу $N_i \in S$ ставится в соответствие неотрицательное число a_i (*возможности*), а каждому узлу $N_j \in T$ — неотрицательное число b_j (*спрос*).

Возникает система ограничений

$$\sum_j x_{ij} - \sum_k x_{ki} = a_i, \quad N_i \in S;$$

$$\sum_j x_{ij} - \sum_k x_{ki} = 0, \quad N_i \in R;$$

$$\sum_j x_{ij} - \sum_k x_{ki} = b_j, \quad N_j \in T;$$

$$0 \leq x_{ij} \leq b_{ij}.$$

Если потоку разрешается течь из любого источника в любой сток, то эта задача легко сводится к задаче с одним источником и одним стоком путем добавления одного дополнительного источника и одного дополнительного стока (рис. 3.1). Помимо этого добавляют новые ориентированные дуги, ведущие из дополнительного источника во все $N_i \in S$ и имеющие пропускные способности a_i , а также ориентированные дуги с пропускными способностями b_i , ведущие из каждого $N_j \in T$ в дополнительный сток. Теперь задача об удовлетворении требуемого спроса заданным предложениям (возможностям) сводится к нахождению максимального потока в расширенной сети.

Если в сети с несколькими источниками и стоками поток должен идти из определенных источников в заданные стоки, возникает так называемая задача о *многопродуктовых потоках в сети*.

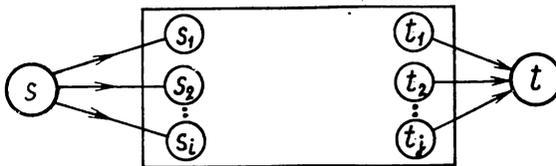


Рис. 3.1. Сведение сетевой задачи к задаче с одним источником и одним стоком

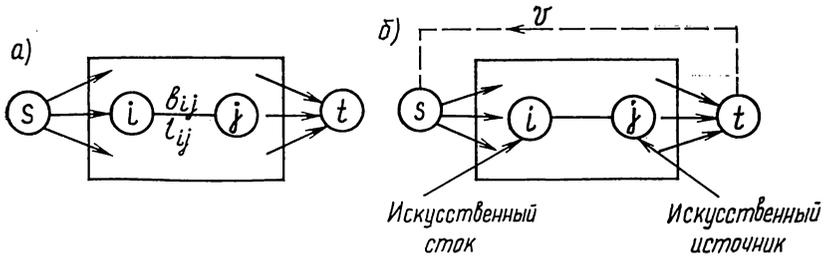


Рис. 3.2. Исходная (а) и расширенная (б) сети

Получим еще одно обобщение задачи о потоке, если для каждой дуги введем ограничение на пропускную способность и сверху, и снизу, т. е.

$$0 \leq l_{ij} \leq x_{ij} \leq b_{ij}.$$

Требуется определить, существует ли поток из источника N_s в сток N_t , удовлетворяющий на дугах ограничениям сверху и снизу.

Пусть в сети имеется только одна ориентированная дуга A_{ij} с ограниченным снизу дуговым потоком l_{ij} . Расширим сеть, добавив два новых узла — искусственный источник N_j с предложением l_{ij} и искусственный сток с таким же спросом l_{ij} . Пропускную способность дуги A_{ij} при этом изменим: если она была равна b_{ij} , то в новой сети она станет равной $b_{ij} - l_{ij}$. Добавим, кроме того, ориентированную дугу из N_t в N_s с бесконечной пропускной способностью. Будем искать в расширенной сети максимальный поток из источника N_j в сток N_i . Если величина этого потока в расширенной сети больше или равна l_{ij} , а поток по дуге A_{ts} равен x_{ts} , то в исходной сети существует такой поток из N_s в N_t величины $v = x_{ts}$, что $l_{ij} \leq x_{ij}$. Исходная и расширенная сети приведены на рис. 3.2 а, б. Если в сети имеется несколько дуг, обладающих нижними границами для дуговых потоков, то следует ввести несколько искусственных источников и стоков, а затем задачу с несколькими источниками и стоками свести к задаче с одним источником и одним стоком введением дополнительного источника и дополнительного стока, как это было сделано в предыдущей задаче.

3.2. Задача о покупке автомобиля

Одним из примеров задачи о *наискратчайшей сети*, или о *сети наименьшей стоимости*, является задача об оптимизации расходов на приобретение и эксплуатацию автомобиля.

Со временем эксплуатационные расходы на содержание автомобиля заметно возрастают, да и сам автомобиль устаревает морально и технически. Возникает вопрос, когда же следует заменить автомобиль и что принять за критерий, определяющий его замену? Выберем в качестве критерия общие затраты на покупку

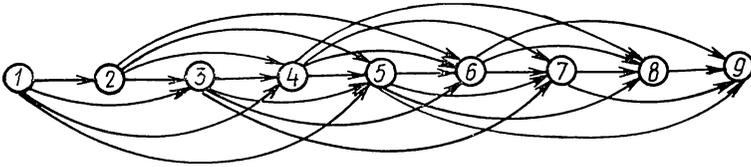


Рис. 3.3. Сеть для задачи о покупке автомобиля

и содержание автомобиля за некоторый период времени T . Для определенности будем считать: $T=8$ лет; в начальный момент автомобиля нет; решение о покупке автомобиля может приниматься в начале каждого года исходя из затрат на его приобретение, эксплуатационных расходов за период, в течение которого автомобиль будет использоваться, и ликвидационной стоимости автомобиля в момент его замены на новый. Предположим, что замена автомобиля должна совершаться по крайней мере каждые 4 года. На рис. 3.3 изображена сеть для рассматриваемой задачи.

Началу каждого года соответствует узел. Если автомобиль куплен в начале i -го года, а заменен в начале j -го года, то такому варианту соответствует дуга (i, j) . Общие затраты на дуге

(i, j) $c_{ij} = P_i + \sum_{k=1}^{j-1} m_k - s_j$, 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. Значения c_{ij} для нашего случая приведены в табл. 3.1.

Таблица 3.1

Общие затраты по дугам

Шаг	2	3	4	5	6	7	8	9
1	1450	2750	3850	4750				
2		1700	3200	4450	5450			
3			1850	3425	4750	5800		
4				1950	3650	5050	6150	
5					2075	3875	5350	6500
6						2200	4100	5650
7							2325	4325
8								2450

Вследствие инфляции и модернизации автомобилей возрастает стоимость автомобиля и увеличиваются расходы на его содержание. Оптимальному решению данной задачи соответствует кратчайшая цепь из узла $s=1$ в узел $t=9$. Поэтому цепь можно рассматривать как цепь, минимизирующую стоимость единицы потока из узла 1 в узел 9 при условии, что стоимость единицы по-

тока по дуге (i, j) равна c_{ij} . Для решения этой задачи воспользуемся алгоритмом Дейкстры (Дикстры).

В алгоритме Дейкстры минимизируется либо стоимость, либо время прохождения единицы потока по данной цепи. Каждой дуге ориентированной сети ставится в соответствие обобщенная стоимость дуги c_{ij} . Фиктивным («бесплатным») дугам приписывается стоимость $c_{ij}=0$, а каждой паре узлов (i, j) , для которых не существует дуги, соединяющей их, — стоимость $c_{ij}=\infty$. Математическая постановка этой задачи следующая:

минимизировать

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

при условии

$$\sum_j f_{sj} - \sum_i f_{is} = 1;$$

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

$$\sum_i f_{ti} - \sum_j f_{jt} = -1; \quad f_{ij} \geq 0; \quad c_{ij} \geq 0.$$

Получили задачу линейного программирования с неизвестными f_{ij} . Согласно первому равенству единица потока вытекает из источника s , а согласно третьему — единица потока втекает в сток t . Второе равенство гарантирует сохранение потока при протекании по сети (естественно, не рассматриваются источник s и сток t). В качестве кратчайшей цепи может быть взята последовательность смежных дуг (i, j) , для которых $f_{ij}=1$.

В алгоритме Дейкстры узлам приписывают либо временные, либо *постоянные пометки*. Первоначально каждому узлу, исключая источник, приписывают пометку, соответствующую длине кратчайшей дуги, ведущей из источника в данный узел. Источнику приписывают постоянную пометку, значение которой равно нулю. Каждому узлу, в который нельзя попасть непосредственно из источника, приписывают временную пометку ∞ , а остальным узлам — временные пометки c_{sj} , $j \neq s$. Если определено, что узел принадлежит кратчайшей цепи, его пометка становится постоянной. Алгоритм Дейкстры основан на следующем простом факте: если известна кратчайшая цепь из узла s (источника) 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 увеличивается на единицу; при $j=t$ алгоритм завершает свою работу. Итерационная процедура алгоритма состоит в следующем. Для заданного узла j обозначим «длину» кратчайшей цепи c_{sj} из источника s в узел j . Если эта «длина» не может быть «улучшена», то

соответствующее значение называют постоянной пометкой. В противном случае «длину» называют временной пометкой. Сначала постоянную пометку присваивают 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. Для получения этой постоянной пометки необходимо следующее.

Шаг 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. Если величина старой временной пометки меньше второй из сравниваемых величин, то пометка остается прежней.

Шаг 2. Среди временных пометок выбрать ту, значение которой минимально и объявить ее постоянной пометкой. Если при этом постоянную пометку приписывают узлу t (стоку), то алгоритм завершает работу, в противном случае перейти на шаг 1.

Для нашего примера:

Шаг 0. Припишем источнику (узел 1) постоянную пометку 0, а узлам $j=2-9$ — временные пометки $c_{sj} = \infty$.

Шаг 1. Для источника остается постоянная пометка 0, для узла 2 временная пометка $c_{12} = 1450$, для узла 3 $c_{13} = 2750$, для узла 4 $c_{14} = 3850$, для узла 5 $c_{15} = 4750$. Для остальных узлов временные пометки равны ∞ . Последним из постоянно помеченных узлов является узел 1 (единственный, иначе надо было взять узел с минимальной пометкой).

Шаг 2. С узлом 1 непосредственно связаны узлы 2—5. Им надо приписать новые временные пометки, равные $\delta_2 = 0 + c_{12} = 1450$, $\delta_3 = 0 + c_{13} = 2750$, $\delta_4 = 0 + c_{14} = 3850$, $\delta_5 = 0 + c_{15} = 4750$. Поскольку δ_2 минимальна из всех, то узлу 2 приписывают постоянную пометку $\delta_2 = 1450$.

Шаг 3. Рассматриваем движение из узла 2; оставляем постоянные пометки 0 для узла 1 и 1450 для узла 2. Узлам 3—6, непосредственно связанным с узлом 2, приписываем временные пометки соответственно $\delta_3 = c_{23} = 2750$, $\delta_4 = c_{24} = 3850$, $\delta_5 = c_{25} = 4750$, $\delta_6 = c_{26} = 6900$, минимальные из возможных временных пометок при движении из узла 2 в узлы $j=3-6$. Например, $\delta_3 = \delta_2 + c_{23} = 1450 + 1700 = 3150 > 2750$; $\delta_4 = 1450 + 3200 = 4650 > 3850$, ... Выбираем $\delta_3 = 2750$, $\delta_4 = 3850$, ... Остальным узлам приписываем временные пометки, равные ∞ .

Шаг 4. Поскольку минимальное значение временных пометок δ_3 , δ_4 , δ_5 , δ_6 равно 2750, узлу 3 приписывают постоянную пометку $\delta_3 = 2750$. Дальнейшие шаги повторяем аналогично шагам 3 и 4. Результаты вычислений сведем в табл. 3.2.

Обратим внимание на выбор временной пометки δ_6 на шаге 9. Последняя постоянная пометка присвоена узлу 5 $\delta_5 = 4750$, тогда

Результаты вычислений по алгоритму Дейкстры

Шаг	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]	[6900]	8550	10 000	11 250
12	[0]	[1450]	[2750]	[3850]	[4750]	[6825]	[8550]	10 000	11 250
13	[0]	[1450]	[2750]	[3950]	[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
16	[0]	[1450]	[2750]	[3850]	[4750]	[6825]	[8550]	[10 000]	[11 250]

для узла 6 временная пометка будет $\delta_6 = \delta_5 + c_{56} = 4750 + 2075 = 6825 < 6900$ (временная пометка δ_6 на шаге 8). Поэтому на шаге 9 берут минимальное значение $\delta_6 = 6825$.

Из решения задачи следует, что минимальные общие затраты на покупку и содержание автомобиля составляют за рассмотренный период 11 250 руб. Кратчайшая цепь состоит из дуг, для каждой из которых разность между значениями постоянных пометок ее концевых узлов равна длине этой дуги: $[\delta_j] = [\delta_i] + c_{ij}$. Последнее соотношение можно использовать рекурсивно, двигаясь от стока t к источнику s . Определив узел, непосредственно предшествующий t в кратчайшей цепи, будем повторять данную процедуру до тех пор, пока не достигнем узла s . В нашем случае первое значение c_{ij} , совпадающее с разностью между значениями постоянных пометок, равно 6500, т. е. узел 5 непосредственно предшествует стоку t (узлу 9) в кратчайшей цепи. Следующим узлом кратчайшей цепи (и исходным) будет узел 1. Отсюда для минимизации общих затрат автомобиль следует заменять в начале первого, пятого и девятого годов.

3.3. Задача о многополюсной кратчайшей цепи

Рассмотрим задачу нахождения кратчайших цепей между всеми парами узлов сети. Кратчайшей цепью между двумя произвольными узлами является цепь, стоимость единицы потока по которой минимальна. Поскольку направление потока в неориентированных дугах нельзя определить заранее, то каждую такую дугу следует заменить двумя ориентированными дугами с противо-

положными направлениями и длинами (стоимостями), 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 решения данной задачи разработан Флойдом. Пусть $N = \{1, 2, \dots, n\}$ — множество узлов, а c_{ij} — количественный параметр (длина, стоимость) дуги (i, j) , направленной от узла i к узлу j . Обозначим через d_{ik}^* длину кратчайшей цепи из узла i в узел k .

Алгоритм Флойда. Он работает следующим образом. Первоначально за длину d_{ik}^* кратчайшей цепи между двумя произвольными узлами i и k (между которыми могут быть и промежуточные узлы) принимают длину дуги (i, k) , соединяющей эти узлы. Затем последовательно проверяют 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}^* присваивают новое значение; если $d_{ik}^* > d_{ij} + d_{jk}$, то значение d_{ik}^* заменяют значением $(d_{ij} + d_{jk})$. Такую процедуру повторяют для всевозможных пар узлов, пока не будут получены все значения d_{ik}^* .

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 . Алгоритм Флойда позволяет решать задачу о многополюсной кратчайшей цепи (пути) для сети из n узлов за n итераций. Обозначим символом d_{ik}^j оценку длины кратчайшей цепи из узла i в узел k , полученную на j -й итерации, и рассмотрим следующую задачу.

Задача. Необходимо соединить восемь объектов многополюсной цепью кратчайшей длины, причем один из объектов (№ 8) может быть только направляющим информацию, а остальные могут и направлять, и получать информацию без каких-либо ограничений (рис. 3.4). На рис. 3.4 каждый объект представлен узлом, а каждая линия — дугой. Ориентированные дуги соответствуют распределительным звеньям, которые могут быть использованы для передачи информации только в указанном направлении. Числа, приписанные дугам, соответствуют расстоянию между объек-

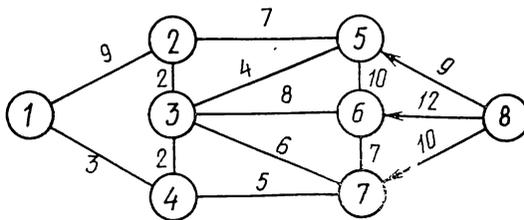


Рис. 3.4. Пример задачи о многополюсной кратчайшей сети

тами. Требуется найти для каждого объекта кратчайшие пути, связывающие его с другими объектами.

Решение. Воспользуемся алгоритмом Флойда. Поскольку $n=8$, то число итераций в алгоритме будет 8. На каждой итерации строят матрицу длин кратчайших путей, которые содержат текущие оценки длин кратчайших цепей $\mathbf{D}^j = \|d_{ik}^j\|$, где $\mathbf{D}^0 = \|c_{ik}\|$, и матрицу маршрутов \mathbf{R}^j , служащую для нахождения промежуточных узлов (если таковые имеются) кратчайших цепей. На j -й итерации $\mathbf{R}^j = \|r_{ik}^j\|$, где r_{ik}^j — первый промежуточный узел кратчайшей цепи из i в k , выбираемый среди множества $\{1, 2, \dots, j\}$, $i \neq j \neq k$; $\mathbf{R}^j = \|r_{ik}^0\|$, где $r_{ik}^0 = k$. Узел r_{ik}^j может быть получен из следующего соотношения:

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

Строим матрицу длин кратчайших цепей \mathbf{D}^0 и матрицу маршрутов \mathbf{R}^0 , отсутствие связи помечаем знаком ∞ ; нулем обозначаем связи внутри одного узла. Для \mathbf{D}^0 получаем

$$\begin{bmatrix} 0 & 9 & \infty & 3 & \infty & \infty & \infty & \infty \\ 9 & 0 & 2 & \infty & 7 & \infty & \infty & \infty \\ \infty & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & \infty & 2 & 0 & \infty & \infty & 5 & \infty \\ \infty & 7 & 4 & \infty & 0 & 10 & \infty & \infty \\ \infty & \infty & 8 & \infty & 10 & 0 & 7 & \infty \\ \infty & \infty & 6 & 5 & \infty & 7 & 0 & \infty \\ \infty & \infty & \infty & \infty & 9 & 12 & 10 & 0 \end{bmatrix}$$

для \mathbf{R}^0

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

Итерация 1. Выбираем базовый узел $j=1$. В матрице \mathbf{D}^0 вычеркиваем j -ю (базовую) строку и j -й (базовый) столбец. Чтобы определить, приведет ли использование узла 1 к более коротким цепям, необходимо исследовать элементы матрицы \mathbf{D}^0 с помощью *трехместной операции*: $d_{ik}^1 = \min[d_{ik}^{j-1}; d_{ij}^{j-1} + d_{jk}^{j-1}]$, $i \neq j \neq k$. Если $d_{ik}^1 = \infty$, т. е. j -й элемент базового столбца равен ∞ , то $d_{ik}^1 = d_{ik}^{j-1}$. Если $d_{jk}^{j-1} = \infty$, т. е. j -й элемент базовой строки равен ∞ , то $d_{ik}^1 = d_{ik}^{j-1}$. 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. Столбцы 3, 5, 7 и 8 содержат элементы, равные ∞ и принадлежащие базовой строке, а строки 3, 5—8 содержат элементы, равные ∞ и принадлежащие базовому столбцу, т. е. исследовать надо элементы

(2,2); (2,4); (4,2); (4,4). Поскольку диагональные элементы можно не рассматривать, необходимо исследовать лишь оценки d_{24}^0 и d_{42}^0 :

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

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

Оценки d_{24}^1 и d_{42}^1 лучше оценок d_{24}^0 и d_{42}^0 и должны быть внесены в матрицу \mathbf{D}^1 , а в матрице маршрутов надо положить $r_{24}^1 = j$, $r_{24}^1 = 1$ и $r_{42}^1 = 1$. Остальные элементы матриц \mathbf{D}^0 и \mathbf{R}^0 остаются без изменений. Получим матрицы

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

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

Итерация 2. Определим узел 2 как базовый, т. е. проверим, приведет ли его использование к более коротким цепям, и выделяем в матрице \mathbf{D}^1 вторую строку и второй столбец. Здесь столбцы 6, 7, 8 содержат элементы, равные ∞ , принадлежащие базовой строке, а строки 6, 7, 8 — элементы, равные ∞ , принадлежащие базовому столбцу; столбцы и строки 6, 7, 8 не рассматриваем. Исключаем и диагональные элементы. Остается исследовать лишь элементы (1,3), (1,4), (1,5), (3,1), (3,4), (3,5), (4,1), (4,3), (4,5), (5,1), (5,3), (5,4). Нетрудно проверить, что здесь улучшены могут быть только оценки $d_{13}^1, d_{15}^1, d_{31}^1, d_{45}^1, d_{51}^1, d_{54}^1$, равные ∞ :

$$d_{13}^2 = \min [d_{13}^1; d_{12}^1 + d_{23}^1] = \min [\infty; 9 + 2] = 11;$$

$$d_{31}^2 = \min [d_{31}^1; d_{32}^1 + d_{21}^1] = \min [\infty; 2 + 9] = 11;$$

$$d_{15}^2 = \min [d_{15}^1; d_{12}^1 + d_{25}^1] = \min [\infty; 9 + 7] = 16;$$

$$d_{51}^2 = \min [d_{51}^1; d_{52}^1 + d_{21}^1] = \min [\infty; 7 + 9] = 16;$$

$$d_{45}^2 = \min [d_{45}^1; d_{42}^1 + d_{25}^1] = \min [\infty; 12 + 7] = 19;$$

$$d_{54}^2 = \min [d_{54}^1; d_{52}^1 + d_{24}^1] = \min [\infty; 7 + 12] = 19.$$

Таким образом, $r_{13}^2 = r_{15}^2 = r_{31}^2 = r_{45}^2 = r_{51}^2 = r_{54}^2 = 2$; остальные элементы r_{ir}^2 остаются без изменения. Новые матрицы имеют вид

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

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

Итерация 3. Определяем, приведет ли использование узла 3 к более коротким цепям. Берем узел 3 в качестве базового и выделяем 3-ю строку и 3-й столбец в матрице D^2 . Исключаем диагональные элементы, элементы 8-го столбца и 8-й строки, исследуем оставшиеся элементы матрицы D^2 , получим новые матрицы D^3 и R^3 . Аналогично получаем матрицы D^j и R^j ; $j=4, 8$. Матрицы D^j , R^j , $j=5, 8$, остаются без изменений. Следовательно, оптимальное решение соответствует матрицам D^5 и R^5 , определяющим и оптимальное расстояние для передачи между объектами и последовательность передачи информации.

$$D^5 = \begin{bmatrix} 0 & 7 & 5 & 3 & 9 & 13 & 8 & \infty \\ 7 & 0 & 2 & 4 & 6 & 10 & 8 & \infty \\ 5 & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & 4 & 2 & 0 & 6 & 10 & 5 & \infty \\ 9 & 6 & 4 & 6 & 0 & 10 & 10 & \infty \\ 13 & 10 & 8 & 10 & 10 & 0 & 7 & \infty \\ 8 & 8 & 6 & 5 & 10 & 7 & 0 & \infty \\ 18 & 5 & 13 & 15 & 9 & 12 & 10 & 0 \end{bmatrix}$$

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

Например, определим кратчайшую цепь из узла 1 в узел 5. По таблице D^5 определяем, что длина этой цепи $d_{15}^5 = 9$. Чтобы найти соответствующую последовательность узлов, обратимся к матрице R^5 . Значение $r_{15}^5 = 4$, т. е. узел 4 является первым промежуточным узлом в кратчайшей цепи из узла 1 в узел 5. Теперь надо определить, какой узел следует за узлом 4 в кратчайшей цепи из узла 4 в узел 5. Рассмотрим r_{45}^5 . Данное значение равно 3. Значит, за узлом 4 следует узел 3. Аналогично, за узлом 3 следует узел 5, так как $r_{35}^5 = 5$. Следовательно, кратчайшая цепь из узла 1 в узел 5 определяется последовательностью узлов 1, 4, 3, 5.

3.4. Анализ сложности алгоритмов поиска кратчайших путей

В алгоритмах Дейкстры (§ 3.2) и Флойда (§ 3.3) выполняются только две элементарные операции — сложение и сравнение.

Рассмотрим сеть, содержащую n узлов. В наихудшем случае в алгоритме Дейкстры конечный узел сети будет n -м по счету узлом, которому приписывают постоянную пометку. Пусть в некоторый момент работы алгоритма m узлам приписаны постоянные пометки, а $n-m$ узлам — временные. Для определения $(m+1)$ -го узла, которому должна быть приписана постоянная пометка, необходимо вычислить $n-m$ временных пометок, выполнив при этом каждый раз операции сложения и 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 $(n-m-1)$ сравнений. 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,

$$\sum_{m=1}^n 3(n-m) = 3 \sum_{m=1}^{n-1} (n-m) = 3[(n-1) + (n-2) + \dots + 1] = 3n(n-2)/2.$$

В алгоритме Флойда на каждой итерации суммарное число элементов, значение которых должно быть оценено с помощью трехместной операции, можно вычислить, учтя, что:

- 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 них не надо получать 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;
- 6) максимальное число операций, выполняемых на одной итерации алгоритма, приблизительно равно $2n(n-3)$;
- 7) поскольку число итераций равно числу узлов n , общее число элементарных операций в алгоритме Флойда в наихудшем случае равно $2n^2(n-3)$.

3.5. Задача о назначениях

В практических приложениях достаточно часто необходимо: так распределить рабочих по рабочим местам, чтобы время изготовления изделия было минимальным; так разместить датчики по объектам, чтобы информация о работе объектов была максимальной; так распределить экипажи самолетов по рейсам, чтобы время простоя техники было минимальным, и т. д. Особенность этой задачи заключается в том, что каждый ресурс (рабочий, датчик, экипаж) используется ровно один раз и каждому объекту будет приписан ровно один ресурс.

Решение задачи может быть записано в виде двумерного массива $\mathbf{X} = [x_{ij}]$, $i=1, m, j=1, m$, где m — число объектов или ресурсов. Искомая переменная

$$x_{ij} = \begin{cases} 1, & \text{если } i\text{-й ресурс назначается на } j\text{-й объект,} \\ 0 & \text{в противном случае.} \end{cases}$$

Критерий эффективности задачи включает в себя элементы матрицы стоимостей c_{ij} — затраты, связанные с назначением i -го ресурса на j -й объект. Для любого недопустимого назначения соответствующую ему стоимость полагают равной достаточно большому числу M . Допустимое решение задачи называют назначением. Для заданного значения m ресурсов и объектов существует $m!$ допустимых решений. Допустимое решение строят путем выбора ровно одного элемента в каждой строке матрицы $\mathbf{X} = \|x_{ij}\|$ и ровно одного элемента в каждом столбце этой матрицы.

В задаче о назначениях требуется минимизировать общую стоимость назначений, т. е. необходимо минимизировать

$$f(x) = \sum_{i=1}^m \sum_{j=1}^m c_{ij} x_{ij}$$

при ограничениях:

- а) каждый ресурс используется ровно один раз —

$$\sum_{j=1}^m x_{ij} = 1 \text{ для всех } i;$$

- б) каждому объекту будет приписан ровно один ресурс —

$$\sum_{i=1}^m x_{ij} = 1 \text{ для всех } j.$$

Очевидно, что задача о назначениях является частным случаем транспортной задачи при единичных значениях параметров a_i и b_i . Поэтому для решения задачи о назначениях можно воспользоваться любым алгоритмом линейного программирования или методом потенциалов для решения транспортной задачи. Но для подобных задач разработан так называемый *венгерский алгоритм*, который наиболее эффективно использует специфику задачи о назначениях.

Задача. Разместить 4 датчика у 4 объектов таким образом, чтобы стоимость такого размещения была минимальна. Матрица стоимости назначений имеет вид

$$C = \|c_{ij}\| = \begin{pmatrix} 2 & 10 & 9 & 7 \\ 15 & 4 & 14 & 8 \\ 13 & 14 & 16 & 11 \\ 4 & 15 & 13 & 19 \end{pmatrix}$$

Решение. В процессе решения задачи о назначениях используют тот факт, что если к каждому элементу i -й строки добавлять действительное число γ_i , а к каждому элементу j -го столбца — действительное число δ_j , то минимизация целевой функции $\sum_i \sum_j c_{ij}x_{ij}$ эквивалентна минимизации функции $\sum_i \sum_j d_{ij}x_{ij}$, где $d_{ij} = c_{ij} + \gamma_i + \delta_j$.

Шаг 1. Редукция строк и столбцов. Данный шаг предназначен для получения в матрице стоимости назначений возможно большего числа нулевых элементов. Для этого в каждой строке из всех элементов вычитают минимальный:

$$\|c_{ij}\| = \begin{pmatrix} 0 & 8 & 7 & 5 \\ 11 & 0 & 10 & 4 \\ 2 & 3 & 5 & 0 \\ 0 & 11 & 9 & 15 \end{pmatrix}.$$

Затем в каждом столбце вычитают минимальные элементы соответствующего столбца — 0, 0, 5, 0 соответственно:

$$\|c_{ij}\| = \begin{pmatrix} \boxed{0} & 8 & 2 & 5 \\ 11 & \boxed{0} & 5 & 4 \\ 2 & 3 & 0 & 0 \\ \boxed{0} & 11 & 4 & 15 \end{pmatrix}.$$

Шаг 2. Определение назначений. Если в полученной матрице стоимостей можно выбрать по *одному* нулевому элементу так, что соответствующее этим элементам решение будет допустимым, то данное назначение оптимально.

Рассмотрим сначала строки матрицы стоимости назначений. Строки 1, 2 и 4 содержат по одному нулю. Взяв эти строки в порядке возрастания их номеров, произведем вначале назначение, соответствующее элементу (1,1), и вычеркнем нулевой элемент

(4, 1), так как в столбце 1 может быть только один нулевой элемент. Затем произведем назначение, соответствующее элементу (2, 2). Элемент (4, 1) был уже вычеркнут.

Рассмотрим столбцы матрицы стоимости назначений с учетом вычеркнутых нулевых элементов по строкам. Столбцы 3 и 4 содержат по одному нулевому элементу. Проведем третье назначение, соответствующее элементу (3, 3). В столбце 4 назначение невозможно, так как нулевой элемент стоит в строке 3, а назначение в ней выполнено. После выполнения второго шага матрица стоимостей приняла следующий вид:

$$\|c_{ij}\| = \begin{pmatrix} \boxed{0} & 8 & 2 & 5 \\ 11 & \boxed{0} & 5 & 4 \\ 2 & 3 & \boxed{0} & 0 \\ 0 & 11 & 4 & 15 \end{pmatrix}.$$

Поскольку полного назначения нулевой стоимости не может быть получено, необходимо провести дальнейшую модификацию редуцированной матрицы стоимости.

Шаг 3. Модификация редуцированной матрицы. Получим новые нулевые элементы. Определим для редуцированной матрицы стоимостей минимальное множество строк и столбцов, содержащих нулевые элементы, и найдем минимальный элемент вне данного множества:

а. Число нулей в строках 1—4 матрицы стоимости соответственно равно 1, 1, 2 и 1, в столбцах оно равно 2, 1, 1 и 1.

б. Максимальное число нулей (по два) содержат строка 3 и столбец 1. Вычеркиваем строку 3.

в. Число невычеркнутых нулей равно 1, 1, 1 соответственно в строках 1, 2 и 4. Для столбцов число невычеркнутых нулей равно 2, 1, 0 и 0. Выбираем столбец 1 и его вычеркиваем. Остался только один невычеркнутый нуль — элемент (2, 2), поэтому можно вычеркнуть либо строку 2, либо столбец 2. Вычеркиваем строку 2 и получаем следующую матрицу:

$$\begin{pmatrix} | & 8 & 2 & 5 \\ \hline | & & & \\ \hline | & 11 & 4 & 5 \end{pmatrix}.$$

Если число линий, необходимое для того, чтобы вычеркнуть нулевые элементы, равно числу строк или столбцов матрицы стоимости, то существует назначение нулевой стоимости.

г. Минимальное значение невычеркнутого множества равно 2. Вычитая его из всех оставшихся элементов

$$\begin{pmatrix} | & 6 & 0 & 3 \\ \hline | & & & \\ \hline | & 9 & 2 & 5 \end{pmatrix}$$

и складывая его со всеми элементами, расположенными на пересечении двух штриховых линий (столбца 1 со строками 2 и 3), получаем новую редуцированную матрицу стоимостей:

$$\|c_{ij}\| = \begin{pmatrix} 0 & 6 & 0 & 3 \\ 13 & 0 & 5 & 4 \\ 4 & 3 & 0 & 0 \\ 0 & 9 & 2 & 13 \end{pmatrix}$$

Остальные элементы предыдущей матрицы стоимости остаются без изменений.

Объяснение последней операции следующее. Если значение этого минимального элемента вычесть из всех остальных элементов матрицы, но на месте нулей будут стоять отрицательные величины и, по крайней мере, один элемент, не принадлежащий выделенному (вычеркнутому) множеству строк и столбцов, станет равным нулю. Однако теперь назначение нулевой стоимости может не быть оптимальным, поскольку матрица содержит отрицательные элементы. Чтобы матрица не содержала отрицательных элементов, прибавим абсолютное значение наименьшего отрицательного элемента ко всем элементам выделенных строк и столбцов. Отметим, что к элементам, расположенным на пересечении выделенных строк и столбцов, данная величина прибавляется дважды. Кроме того, все отрицательные элементы преобразуют в нулевые или положительные.

Новая редуцированная матрица стоимостей содержит еще один нуль — элемент (1, 3). Возвращаемся к шагу 2. Шаги 2 и 3 выполняют до тех пор, пока не будет получено оптимальное решение.

Шаг 4. Проведение назначений. Проведем назначение, соответствующее элементу 1, 3; нуль (1, 1) зачеркиваем. Затем проведем назначения по элементам (2, 2), (4, 1), (3, 4). Получили оптимальное решение, поскольку проведено полное назначение:

$$C^* = \|c_{ij}\| = \begin{pmatrix} 0 & 6 & 0 & 3 \\ 13 & 0 & 5 & 4 \\ 4 & 3 & 0 & 0 \\ 0 & 9 & 2 & 13 \end{pmatrix}; \quad X = \|x_{ij}\| = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Минимальное значение целевой функции (c_{ij} берем из исходной матрицы стоимости назначений):

$$f(x) = 9 + 4 + 11 + 4 = 28.$$

Замечание. Если в задаче о назначениях следует обратить в максимум критерий эффективности, то все элементы матрицы стоимостей следует умножить на -1 (тем самым переходим к минимизации исходной целевой функции) и сложить их с достаточно большим числом M так, чтобы вновь получившаяся матрица стоимости не содержала отрицательных элементов (M до-

статочно выбрать равным по модулю наименьшему отрицательному элементу в матрице стоимости). Теперь задачу можно решать как задачу минимизации.

3.6. Задача размещения производства

Пусть намечается выпуск m видов продукции, которые могли бы производиться на n предприятиях, $n > m$. Следует из n предприятий выбрать такие m , каждое из которых будет производить один вид продукции. Издержки производства и сбыта единицы продукции, плановый объем годового производства продукции и плановая стоимость единицы продукции каждого вида известны.

Эту задачу путем введения фиктивных $(n - m)$ видов продукции с нулевыми стоимостями сводим к задаче о назначениях.

Издержки производства и сбыта заданы в табл. 3.3 и 3.4.

Таблица 3.3

Издержки производства на единицу продукции (руб.)

Вид продукции	Предприятие				
	1	2	3	4	5
1	20	23	38	15	35
2	8	29	6	35	35
3	5	8	3	4	7

Таблица 3.4

Издержки сбыта на единицу продукции (руб.)

Вид продукции	Предприятие				
	1	2	3	4	5
1	20	50	20	10	13
2	7	90	8	35	60
3	5	5	4	15	6

Плановый объем производства соответственно 35 000 ед.; 16 000 ед.; 54 000 ед.; плановая стоимость — 55, 50, 30 руб.

Решение. Вычислим прибыль на единицу продукции (разность цены и суммарных издержек на производство и сбыт) (табл. 3.5).

Таблица 3.5

Прибыль на единицу продукции

Вид продукции	Предприятие				
	1	2	3	4	5
1	15	-18	-3	30	7
2	35	-69	36	-20	-45
3	20	17	23	11	17

Умножив прибыль, приходящуюся на единицу продукции, на годовой объем сбыта, получим общую годовую прибыль, соответствующую каждой паре: вид продукции — предприятие (табл. 3.6).

Таблица 3.6

Прибыль предприятий

Вид продукции	Предприятие				
	1	2	3	4	5
1	525	—630	—105	1050	245
2	5600	—11 040	5760	—3200	—7200
3	1080	918	1242	594	918

Чтобы свести исходную задачу к задаче о назначениях, введем два вида фиктивной продукции (4 и 5), которой соответствует нулевая прибыль. Поскольку это задача максимизации прибыли, то умножим все элементы матрицы на -1 . Наименьшее отрицательное число -5760 — элемент (2, 3). Затем ко всем элементам матрицы (кроме нулевых) прибавим число 5760. В результате получим матрицу стоимости назначений (табл. 3.7). Далее редуцируем матрицу. Модифицируем редуцированную матрицу, вычеркивая в ней третий и четвертый столбцы, четвертую и пятую строки. Получим оптимальное решение (табл. 3.8).

Таблица 3.7

Матрица стоимости

Вид продукции	Предприятие				
	1	2	3	4	5
1	5235	6390	5865	4710	5515
2	160	16 800	0	8960	12 960
3	4680	4842	4518	5166	4842
4	0	0	0	0	0
5	0	0	0	0	0

Таблица 3.8

Оптимальное решение

Вид продукции	Предприятие				
	1	2	3	4	5
1	365	1520	1155	[0]	645
2	[0]	16 640	0	8960	12 800
3	2	164	[0]	648	164
4	0	[0]	160	160	0
5	0	0	160	160	[0]

Отсюда производство первого вида продукции назначается 4-му предприятию, второго вида — 1-му, третьего вида — 3-му. Суммарная годовая прибыль, соответствующая данному решению, равна 7 892 000 руб.

Модифицированный алгоритм решения задачи о назначениях может быть использован и при решении задачи коммивояжера (см. § 3.9).

3.7. Задача о максимальном потоке

Пусть задана ориентированная сеть с одним источником s и одним стоком t и пусть дуги (i, j) имеют ограниченную пропускную способность. Задача о максимальном потоке заключается в поиске таких потоков по дугам, что результирующий поток, протекающий из источника s в сток t , является максимальным. Задача о максимальном потоке может быть сформулирована как задача линейного программирования и поэтому может быть решена симплекс-методом.

Здесь для ее решения будет использован более эффективный метод расстановки пометок, разработанный Фордом и Фалкерсоном. Алгоритм начинает работу с некоторого допустимого решения, узлы рассматривают как промежуточные пункты передачи потока, а дуги — как распределительные каналы. Для описания алгоритма вводят два понятия — *пометки и увеличивающие (аугментальные) пути потока*. Пометку узла используют 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 посылают из узла i в узел j и это вызывает увеличение потока по данной дуге, то будем говорить, что узел j *помечен* из узла i символом $(+q_j)$; в данном случае узлу j приписывают пометку $[+q_j; i]$. Если посылка q_j единиц потока вызывает уменьшение потока по дуге, то будем говорить, что узел j *помечен* из узла i символом $(-q_j)$. В таком случае узлу j приписывают пометку $[-q_j; i]$.

Текущий поток из узла i в узел j увеличивается, когда q_j единиц дополнительного потока послано в узел j по ориентированной дуге (i, j) в направлении, совпадающем с ее ориентацией. В таком случае дугу (i, j) называют *прямой*. Если q_i единиц дополнительного потока послано в узел j по ориентированной дуге (j, i) , т. е. в направлении, противоположном ее ориентации, то текущий поток из j в i уменьшается, а дугу (j, i) называют *обратной*. Узел j может быть помечен из узла i только после того, как узлу i приписана пометка. Если узел j помечен 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*. Эту величину обычно называют *остаточной пропускной способностью*. Если некоторому узлу приписана пометка и при этом использует-

ся прямая ветвь, то она может иметь только *остаточную пропускную способность*.

Увеличивающий (аугментальный) путь потока из s в t определяют как связную 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 используют 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.

Как конкретно можно увеличить поток на q_j единиц? Предположим, что дуге (i, j) уже приписан поток $f_{ij} \geq 0$, $f_{ij} \leq u_{ij}$, где u_{ij} — пропускная способность дуги (i, j) . Величина q_j не может превосходить остаточной пропускной способности $u_{ij} - f_{ij}$. Но 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 добавлено в узел i , т. е. самое большое q_i . Следовательно, поток по прямой дуге (i, j) можно увеличить на величину $q_j = \min[q_i, u_{ij} - f_{ij}]$. Аналогично помечают узел j , если дуга (j, i) является *обратной*. Уменьшение потока по дуге (j, i) возможно только в том случае, когда $f_{ji} > 0$. Поток может быть уменьшен самое большее на число единиц потока, которые можно взять из узла i , т. е. на величину q_i . Следовательно, поток по обратной дуге (j, i) может быть уменьшен на величину $q_j = \min[q_i; f_{ji}]$.

В начале работы алгоритма источнику приписывают пометку $[\infty; -]$, указывающую 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 (за исключением источника) в начальный момент не помечены. Пытаясь достичь стока, мы проходим, выбирая дуги с наибольшими пропускными способностями, по прямым и обратным дугам и последовательно помечаем принадлежащие им узлы. Возможны два случая.

1. Стоку t приписывается метка $[+q_t; k]$, k — номер узла, связанного со стоком. В этом случае аугментальный путь потока найден и поток по каждой дуге этого пути может быть увеличен или уменьшен на величину q_t . После изменения дуговых потоков текущие пометки стирают и всю описанную процедуру выполняют заново.

2. Сток t не может быть намечен. Это означает, что аугментальный путь потока не может быть найден. Следовательно, построенные дуговые потоки образуют оптимальное решение (максимальный поток).

Задача. Найти максимальное количество информации, которое может быть передано из источника s в сток t по сети, представленной на рис. 3.5. Пропускные способности дуг u_{ij} отмечены на рис. 3.5.

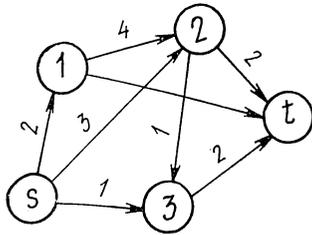


Рис. 3.5. Исходные данные для задачи о максимальном потоке (о максимальном количестве информации)

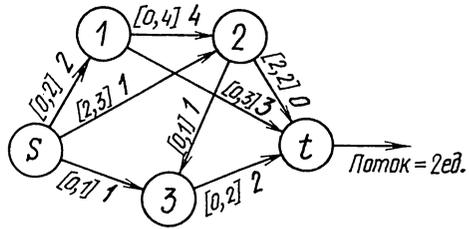


Рис. 3.6. Первая итерация

Решение.

Итерация 1. Каждой дуге (i, j) приписывают пометку $[f_{ij}; u_{ij}]$, где f_{ij} — текущий дуговой поток, u_{ij} — пропускная способность дуги. Приписываем узлу s пометку $[\infty; -]$. Из источника s максимальный поток идет в узел 2; узлу 2 приписываем пометку $[+3; s]$, так как $q_2 = \min [q_s; u_{s2}] = \min [\infty; 3] = +3$. Из узла 2 максимальная пропускная способность принадлежит дуге $(2, t)$, ведущей в сток t . Приписываем стоку пометку $[+2, 2]$, так как $q_t = \min [q_2; u_{2t}] = \min [3, 2] = 2$. Изменение дуговых потоков возможно на 2 единицы $f_{s2} = 2$; $f_{2t} = 2$. Дуги получают следующие пометки (рис. 3.6).

Итерация 2. Теперь просматриваем остаточные пропускные способности $u_{ij} - f_{ij}$ (выделены жирным шрифтом). Источнику s приписываем пометку $[\infty; -]$. Из источника s предпочтительнее двигаться в узел 1; узлу 1 приписываем пометку $[+2; s]$. Из узла 1 идем в узел 2; узлу 2 приписываем пометку $[+2; 1]$, так как $q_2 = \min [q_1; u_{12} + f_{12}] = 2$. Из узла 2 идем в узел 3. Для узла 3 $q_3 = \min [q_2; u_{23} - f_{23}] = \min [2; 1] = 1$, узлу 3 приписываем пометку $[+1; 2]$. Из узла 3 идем в сток. Стоку приписываем пометку $[+1; 3]$. Минимальное изменение дуговых потоков равно 1, т. е. изменение дуговых потоков $f_{31} = 1$, $f_{12} = 1$, $f_{23} = 1$, $f_{3t} = 1$. Максимальный поток равен $2 + 1 = 3$ ед. Дуги получают следующие пометки (рис. 3.7).

Итерация 3. Приписываем узлу s пометку $[\infty; -]$. Из узла s движемся в узел 1: $q_1 = \min [q_s; u_{s1} - f_{s1}] = \min [\infty; 1] = +1$. Узлу 1 приписываем пометку $[+1; s]$. Из узла 1 идем в сток t : $q_t = \min [q_1; u_{1t} - f_{1t}] = \min [1, 3] = 1$. Стоку t приписываем пометку $[+1; 1]$. Изменение дуговых потоков для f_{s1} равно 1; для f_{1t} равно 1, изменятся только значения потоков на дугах $(s, 1)$ и $(1, t)$. Полные потоки станут равными: $f_{s1} = 2$, $f_{1t} = 1$. Дуги получают следующие пометки (рис. 3.8). Максимальный поток станет равным $3 + 1 = 4$ ед.

Итерация 4. Приписываем узлу s пометку $[\infty; -]$. Из узла s направляемся в узел 2: $q_2 = \min [q_s; u_{s2} - f_{s2}] = \min [\infty; 1] = +1$; узлу 2 приписываем пометку $[+1; s]$. Из узла 2 мы можем по-

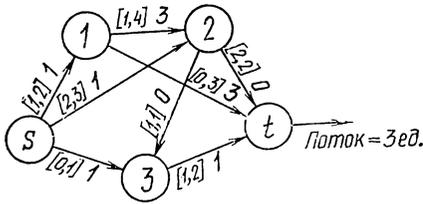


Рис. 3.7. Вторая итерация

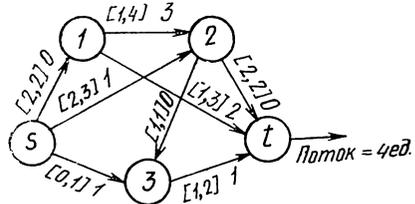


Рис. 3.8. Третья итерация

пасть в сток t , только пройдя по обратной дуге в узел 1, а затем в сток t . Определим пометку узла 1 в данном случае: $q_1 = \min [q_2; f_{12}] = \min [+1; 4] = +1$, т. е. узлу 1 приписываем пометку $[-1; 2]$. Для узла t : $q_t = \min q_i$; $u_{1t} - f_{1t} = \min [+1; 1] = +1$; узлу t приписываем пометку $[+1; 1]$. Изменение дуговых потоков для f_{s2} равно 1, для f_{12} равно 1, для f_{1t} равно 1. Потоки по этим дугам будут соответственно $f_{s2} = 2$, $f_{12} = 0$, $f_{1t} = 2$; максимальный поток из s в t равен $4 + 1 = 5$ ед. Дуги получают следующие пометки [изменяют пометки только на дугах $(s, 2)$, $(1, 2)$ и $(1, t)$] (рис. 3.9).

Итерация 5. Приписываем узлу s пометку $[\infty, -]$. Из узла мы можем попасть только в узел 3. Для узла 3: $q_3 = \min [q_s; u_{s3} - f_{s3}] = \min [\infty; +1] = +1$; узел 3 получает пометку $[+1; s]$. Из узла 3 попадаем в сток t : $q_t = \min [q^3; u_{3t} - f_{3t}] = \min [1; 1] = +1$. Сток t получает пометку $[+1, 3]$. Изменение дуговых потоков для f_{s3} и f_{3t} равно 1; соответственно потоки $f_{s3} = 1$; $f_{3t} = 2$ (На второй итерации $f_{3t} = 1$). Максимальный поток равен $5 + 1 = 6$ ед. Дуги получают следующие пометки (рис. 3.10).

Итерация 6. Приписать узлу s пометку $[\infty, -]$. Аугментальный поток не может быть найден, т. е. построенные дуговые потоки образуют оптимальное решение. Максимальный поток (максимальное количество информации) из источника s в сток t равен 6 ед.: 2 единицы потока идут из источника s через узел 1 в сток t ; 1 единица потока идет из источника s через узел 3 в сток t ; 3 единицы потока идут из источника s в узел 2, затем 2 единицы потока поступают из узла 2 в сток t , а 1 единица потока поступает в сток t через узел 3. Через дугу $(1, 2)$ информация не идет.

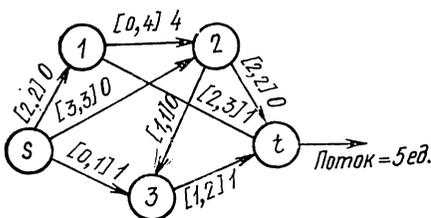


Рис. 3.9. Четвертая итерация

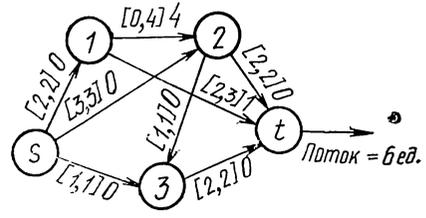


Рис. 3.10. Оптимальное решение

3.8. Задача о многополюсном максимальном потоке

Мы рассмотрели задачу о максимальном потоке из единственного узла s в единственный узел t . В целом ряде технических и экономических приложений возникают задачи, в которых надо определить максимальный поток между любыми выбранными парами узлов в сети. Примерами таких систем являются: сети автомобильных дорог, где автострады изображаются дугами, пропускные способности которых соответствуют максимально допустимой интенсивности движения; телефонная или информационная сеть, где линии представляются дугами, а их пропускные способности соответствуют максимальному числу вызовов (или объему информации), которые могут обслуживаться в каждый момент времени; электрические и электроэнергетические системы, где линии электропередачи представлены дугами, а пропускные способности соответствуют максимальному объему электроэнергии, который может передаваться по линиям в единицу времени и т. п.

Задачи о многополюсных максимальных потоках бывают двух видов: анализа и синтеза сети.

Задача анализа сети. Задана сеть с ограниченными пропускными способностями дуг. Следует определить, каковы величины максимальных потоков, которые можно пропустить между всеми парами узлов в заданной сети.

Задача синтеза сети. Требуется построить сеть, в которой величины максимальных потоков f_{ij} между всеми парами узлов удовлетворяют заданным ограничениям снизу и в которой общая пропускная способность всех дуг максимальна.

Задачу о многополюсном максимальном потоке можно решить, решив рассмотренную задачу о максимальном потоке между единственным узлом и единственным стоком. 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. Рассмотрим здесь алгоритм Гомори—Ху, согласно которому максимальный поток в задаче о многополюсном максимальном потоке определяют только $(n-1)$ раз. Основная идея алгоритма состоит в итеративном построении *максимального острова дерева*, ветви которого соответствуют разрезам, а параметры ветвей — величинам разрезов. Пусть $G=(N, A)$ — неориентированная сеть, где N — множество узлов, A — множество дуг, и пусть c_{ij} — пропускная способность дуги (i, j) из множества A , причем $c_{ij}=c_{ji}$. Максимальный поток между узлами i и j равен v_{ij} ; $(X, \bar{X})_{ij}$ — минимальный разрез, отделяющий i от j ($i \in X$; $j \in \bar{X}$; $c(X, \bar{X})_{ij}$ — пропускная способность минимального разреза, отделяющего i от j . Согласно теореме о максимальном потоке и минимальном разрезе $v_{ij}=c(X, \bar{X})_{ij}$. Если некоторый узел $k \in \bar{X}$, то $v_{ik} \leq c(X, \bar{X})_{ij}$, а если $k \in X$ то $v_{kj} \leq c(X, \bar{X})_{ij}$. Следовательно,

$v_{ij} \geq v_{ik}$ и $v_{ij} \geq v_{kj}$. Поэтому $v_{ij} \geq \min[v_{ik}; v_{kj}]$. Если рассмотреть в общем случае связанное множество узлов $\{i; p; k; q; j\}$, то

$$v_{ij} \geq \min[v_{ip}, v_{pk}, v_{kq}, v_{qj}].$$

Но с другой стороны для максимального остовного дерева

$$v_{ij} \leq \min[v_{ip}, v_{pk}, v_{kq}, v_{qj}],$$

где (i, j) — произвольная дуга, не принадлежащая данному дереву. Если это не так, то вместо любой дуги пути из i в j можно взять дугу (i, j) , в результате чего будет построено дерево с *большим весом*. Отсюда для любой дуги, не принадлежащей дереву,

$$v_{ij} = \min[v_{ip}, v_{pk}, v_{kq}, v_{qj}].$$

Максимальное остовное дерево, удовлетворяющее последнему равенству, называют *деревом разрезов* потому, что каждая его ветвь соответствует разрезу, а вес ветви равен пропускной способности разреза. Если требуется определить величину максимального потока между двумя противоположными узлами, надо в дереве найти путь, соединяющий эти два узла, и выбрать в этом пути дугу с минимальным весом. Ее вес равен величине максимального потока между рассматриваемыми узлами.

Если некоторый узел s рассматривать как источник, а другой узел t как сток, то максимальный поток между ними $v_{st} = c(\bar{X}, \bar{X})_{st}$. Если затем в качестве источника и стока выбирают другую пару узлов (i, j) , удовлетворяющих условию, что оба они принадлежат \bar{X} (или \bar{X}), то множество узлов \bar{X} [или \bar{X} , если (i, j) принадлежат \bar{X}] может быть объединено в один *конденсированный узел*. При этом величина максимального потока из i в j будет одной и той же для исходной и *конденсированной сетей*, и в алгоритме Гомори — Ху при определении величины v_{ij} используют решение задачи о максимальном потоке, найденное на предыдущем шаге.

Пусть \bar{N}_{ij} — множество узлов, образуемое в результате конденсации всех узлов, лежащих по ту сторону разреза, где не содержатся узлы i и j ; \bar{A}_{ij} — множество дуг, соединяющих узлы из множества \bar{N}_{ij} . Если известны пропускные способности дуг, принадлежащих \bar{A}_{ij} , то для нахождения величины максимального потока между узлами i, j можно воспользоваться процедурой расстановки *пометок*.

В свою очередь, пропускные способности дуг из \bar{A}_{ij} определяют следующим образом. Пусть j_1, j_2, \dots, j_r — узлы из множества \bar{X} , непосредственно связанные с узлом $i \in \bar{X}$. При конденсации множества \bar{X} дуги $(i, j_1), (i, j_2), \dots, (i, j_r)$ заменяют одной дугой соединяющей узел i и конденсированный узел \bar{X} . Пропускная способность такой дуги

$$c_{i\bar{X}} = c_{ij_1} + c_{ij_2} + \dots + c_{ij_r} = \sum_{m=1}^r c_{ij_m}.$$

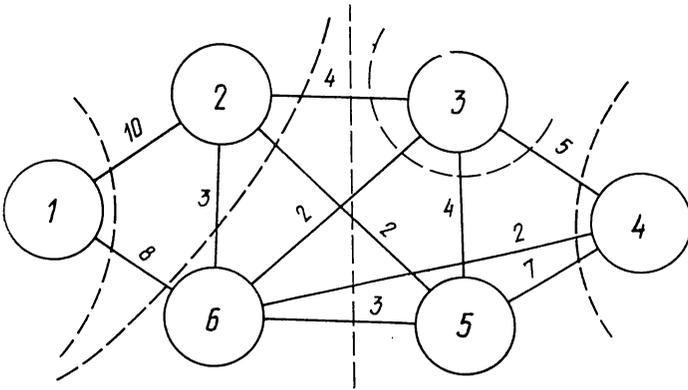


Рис. 3.11. Исходные данные для задачи о многополюсном максимальном потоке информации

Для определения величины v_{ij} вновь надо найти минимальный разрез, отделяющий i от j — $(X, \bar{X})_{ij}$ с минимальной пропускной способностью. Теперь можно выбрать другую пару узлов, принадлежащих либо X , либо \bar{X} , и построить конденсированную сеть. В результате выполнения процедуры расстановки пометок можно будет определить другой разрез и построить новую конденсированную сеть и т. д.

Задача. В сети (рис. 3.11) для каждой пары узлов определить величину максимального потока информации между ними.

Решение. Для простоты сначала найдем максимальные потоки между узлами, N_1, N_3, N_4 и N_5 .

Итерация 1. Выберем два произвольных узла, скажем N_1 и N_3 , и найдем максимальный поток между ними (путем процедуры расстановки пометок). Получим минимальный разрез $(N_1, N_2, N_6 | N_3, N_5)$ величины $4+2+2+2+3=13$ и начало построения дерева разрезов (рис. 3.12).

Итерация 2. Найдем максимальный поток между N_3 и N_1 (рис. 3.13). Здесь узлы 1, 2 и 6 объединены в конденсированный узел. Узел 3

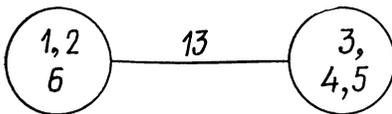


Рис. 3.12. Начало построения дерева разрезов

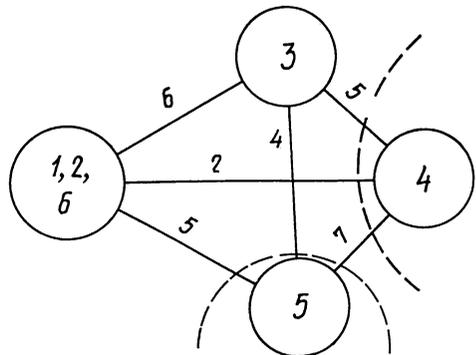


Рис. 3.13. К определению максимального потока между N_3 и N_1

соединен с узлами 2 и 6, т. е. пропускная способность дуги от конденсированного узла к узлу 3 будет равна $4+2=6$. Узел 5 соединен с узлами 2 и 6, т. е. пропускная способность дуги, соединяющей конденсированный узел с узлом 5, равна $2+3=5$. Величина максимального потока равна $5+2+7=14$, минимальный разрез $(N_1, N_2, N_6, N_3, N_5 | N_4)$. По минимальному разрезу определим, что узлы 3, 5 и (1, 2, 6) лежат по одну сторону в минимальном разрезе, разделяющем N_3 и N_4 . Получим рис. 3.14.

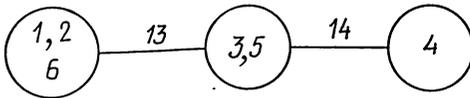


Рис. 3.14. Построение дерева разрез

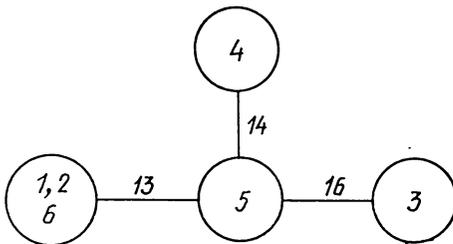


Рис. 3.15. Символическое изображение сети с помощью дерева разрез

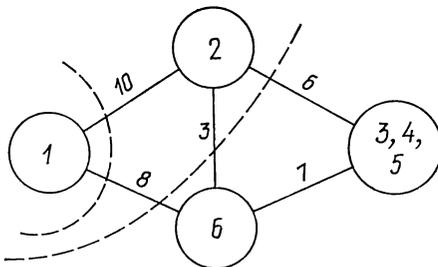


Рис. 3.16. К нахождению максимального потока между узлами N_1 и N_6

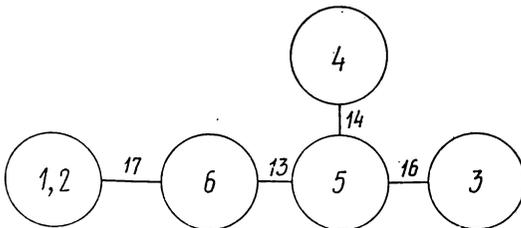


Рис. 3.17. К построению дерева разрез

Итерация 3. Найдем максимальный поток в той же сети (рис. 3.13) между N_3 и N_5 . Получим минимальный разрез $(N_1, N_2, N_6, N_5, N_4 | N_3)$ величиной $5+4+7=16$ и символическое изображение сети (рис. 3.15). Теперь каждая интересующая нас вершина в полученном дереве (рис. 3.15) содержит только по одному полюсу и максимальные потоки будут $f_{13}=f_{14}=f_{15}=13$ (они равны минимальной пропускной способности дуги на данном пути), $f_{34}=f_{45}=14$; $f_{35}=16$. Эти потоки равны соответствующим максимальным потокам в исходной сети (рис. 3.15).

Если необходимо найти величины максимальных потоков между всеми парами узлов, то следует процесс продолжить.

Итерация 4. Найдем максимальный поток между узлами N_1 и N_6 в сети, изображенной на рис. 3.16. Минимальный разрез $(N_1, N_2 | N_6, N_3, N_4, N_5)$ величины $6+3+8=17$ изображен на рис. 3.17.

Итерация 5. Найдем максимальный поток меж-

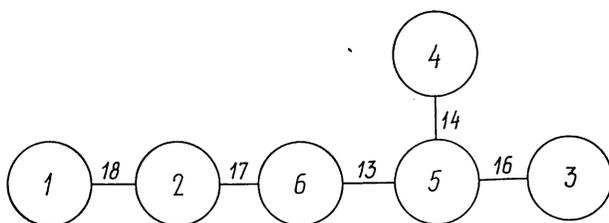


Рис. 3.18. Дерево разрезов

ду узлами N_1 и N_2 . Минимальный разрез $(N_1 | N_2, N_3, N_4, N_5, N_6)$ величины $10+8=18$ приведен на рис. 3.18. С помощью дерева, представленного на рис. 3.18, можно определить величины максимальных потоков между всеми парами узлов (табл. 3.9).

Таблица 3.9

Величины максимальных потоков между узлами сети

	1	2	3	4	5	6
1	∞	18	13	13	13	17
2	18	∞	13	13	13	17
3	13	13	∞	14	16	13
4	13	13	14	∞	14	13
5	13	13	16	14	∞	13
6	17	17	13	13	13	∞

Если в двух сетях равны величины максимальных потоков между парами узлов, принадлежащих некоторому заданному множеству, то эти сети называют *потоко-эквивалентными по отношению к заданному множеству узлов*. Заметим, что существует много деревьев, которые потоко-эквивалентны некоторой заданной сети. Потоко-эквивалентное дерево (рис. 3.18) обладает следующей особенностью: каждая ветвь этого дерева соответствует некоторому минимальному разрезу в исходной сети. Поэтому его называют *деревом разрезов*. На дереве разрезов для сети, содержащей n узлов, показано $(n-1)$ минимальных разрезов исходной сети, не пересекающихся друг с другом (см. рис. 3.11).

3.9. Задача коммивояжера

В гл. 1 мы упоминали задачу о Кенигсбергских мостах, в которой требовалось обойти все мосты, не пересекая ни один из них дважды, т. е. пройти все мосты по кратчайшему пути.

Можно привести и другую словесную постановку этой задачи в более общем виде. Некто решил, что настало время вырваться из дома и посмотреть на жизнь в других странах. Он выбрал семь мест, которые намеревался обязательно посетить, получил инфор-

мацию о стоимости проезда самолетом в каждый из выбранных городов и стоимость проезда из одного города в другой. Учел все льготы, предоставляемые авиакомпаниями, и составил матрицу стоимостей (табл. 3.10) проезда в выбранные города и обратно. Зная матрицу стоимостей, путешественнику надо так составить маршрут путешествия, чтобы затраты на путешествие были минимальными и чтобы выполнялось естественное требование: каждый пункт посещается только один раз. Матрица стоимостей из-за льгот авиакомпаний оказалась несимметричной. При симметричной матрице стоимостей процедура решения задачи не изменяется.

Таблица 3.10

Матрица стоимостей (условные единицы)

№	Пункты	Исходный пункт	Токио	Гонконг	Лондон	Сидней	Рим
		1	2	3	4	5	6
1	Исходный пункт	—	270	430	160	300	260
2	Токио	70	—	160	10	300	300
3	Гонконг	200	130	—	350	50	0
4	Лондон	210	160	250	—	180	180
5	Сидней	120	460	270	480	—	50
6	Рим	230	50	50	90	50	—

Примечание. В процессе проведения расчетов все стоимости будут уменьшены в 10 раз.

Подобные задачи в математическом программировании носят название «задача коммивояжера». Ее формулируют следующим образом. Коммивояжер должен выехать из исходного пункта и побывать в каждом из остальных $(n-1)$ пунктов ровно 1 раз и вернуться в исходный пункт. Задача заключается в определении последовательности объезда пунктов, при которой коммивояжеру требуется минимизировать некоторый критерий эффективности: стоимость проезда, время в пути, суммарное расстояние и т. д. Здесь требуется выбрать один или несколько оптимальных маршрутов из $(n-1)!$ возможных. Если некоторые города для коммивояжера недоступны, то минимальное значение целевой функции должно быть бесконечно большим.

Рассмотрим решение задачи коммивояжера *методом ветвей и границ*. Вначале определяют некоторое допустимое решение (*допустимый маршрут*). После чего множество всех оставшихся маршрутов разбивают на все более мелкие подмножества и при каждом разбиении вычисляют *нижнюю границу* целевой функции текущего наилучшего маршрута. С помощью найденных границ проводят дальнейшее разбиение подмножеств допустимых маршрутов и в конечном итоге определяют оптимальный маршрут. Это разбиение подмножеств маршрутов можно рассматривать как узлы дерева.

Поэтому данный метод называют *методом поиска по дереву решений*, или методом ветвей и границ.

Матрица стоимостей содержит неотрицательные элементы c_{ij} . Маршрут T можно представить как множество упорядоченных пар пунктов $T = \{(i_1, i_2), (i_2, i_3), \dots, (i_{n-1}, i_n), (i_n, i_1)\}$. Каждый допустимый маршрут представляет собой цикл, проходя по которому коммивояжер посещает каждый пункт ровно один раз и возвращается в исходную точку. Каждая упорядоченная пара (i, j) является *дугой*, или *звеном маршрута*. Стоимость маршрута T равна сумме соответствующих элементов матрицы стоимостей, но только тех, что лежат на маршруте T :

$$z(T) = \sum_{i, j \in T} c_{ij}.$$

Величина $z(T)$ определена для любого допустимого маршрута и не может быть меньше длины оптимального маршрута, т. е. текущее значение $z(T)$ является *верхней границей* $z_b(T)$ стоимости оптимального маршрута T .

Для вычисления нижних границ стоимости маршрута используют понятие *редукции строк и столбцов* матрицы стоимости. Процедуру вычитания из каждого элемента строки наименьшего элемента этой же строки и из каждого элемента столбца наименьшего элемента этого же столбца называют соответственно *редукцией строк* и *редукцией столбцов*. Матрицу с неотрицательными элементами, в каждой строке и в каждом столбце которой содержится по крайней мере один нулевой элемент, называют *редуцированной*. Она может быть получена в результате последовательной редукции ее строк и столбцов. Если вычесть из каждого элемента некоторой строки матрицы стоимости постоянную величину c , то стоимость любого маршрута, определяемая новой матрицей, меньше стоимости того же маршрута, определяемого старой матрицей, на величину c , так как для любого допустимого маршрута каждая строка и каждый столбец матрицы стоимости содержит по одному элементу, соответствующему этому маршруту. При редукции относительные стоимости всех маршрутов останутся неизменными. Следовательно, останутся неизменными и все оптимальные маршруты.

Если $z(T)$ — стоимость маршрута T , определяемая матрицей стоимости до выполнения редукции, $z_1(T)$ — стоимость того же маршрута, определяемая редуцированной матрицей, H — сумма всех констант, используемых при вычислении редуцированной матрицы, то $z(T) = z_1(T) + H$. Поскольку редуцированная матрица содержит только неотрицательные элементы, то H является *нижней границей* стоимости маршрута T для нередуцированной матрицы стоимости.

В алгоритме метода ветвей и границ диагональные элементы исходной матрицы стоимости полагают равными ∞ , т. е. $c_{ii} = \infty$.

1. Выберем произвольный допустимый маршрут, например состоящий из звеньев $(1, 4)$ $(4, 5)$; $(5, 3)$; $(3, 6)$; $(6, 2)$; $(2, 1)$. Стоимость данного маршрута $z_b(T) = 16 + 18 + 27 + 0 + 5 + 7 = 73$,

т. е. для оптимального маршрута стоимость поездки коммивояжера не может превосходить значения $z_b(T)$.

2. Выполним редукцию строк, а затем столбцов матрицы стоимости; для этого в каждой определим минимальный элемент и найденное значение c_i вычтем из элементов соответствующей строки. Получим табл. 3.11.

Таблица 3.11

Редукция строк

Узлы	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

3. Затем в полученной табл. 3.11 проведем редукцию столбцов. Из табл. 3.11 видно, что проводить следует только редукцию первого столбца, так как остальные столбцы содержат нулевые элементы. Редукция столбцов показана в табл. 3.12.

Таблица 3.12

Редукция столбцов

Узлы	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	$H = 48$

Строка Q_j содержит вычитаемые константы для каждого столбца при редукции столбцов. Значение нижней границы для всех маршрутов в рассматриваемой задаче равно H — сумме всех вычитае-

мых констант $H = \sum_{i=1}^6 c_i + \sum_{j=1}^6 Q_j = 48$.

4. Теперь следует выбрать оптимальный маршрут. Если бы в каждой строке и в каждом столбце было ровно по одному нулевому элементу, то эти элементы и образовали бы оптимальный маршрут и оптимальная стоимость проезда равнялась бы H .

Однако нулевые элементы не единственны в строках и столбцах. Вместо того чтобы одновременно определять все звенья оптималь-

ного маршрута с помощью текущей матрицы стоимости, воспользуемся алгоритмом, на каждом шаге которого по матрице стоимости строится одно звено оптимального маршрута. Естественно вначале выбрать звено нулевой длины, а затем последовательно добавлять звенья нулевой или минимальной длины. Если выбрать звено (i, j) , то решение не должно содержать других звеньев, соответствующих элементам i -й строки и j -го столбца; если звено (i, j) можно исключить из окончательного решения, то его можно не рассматривать при выполнении последующих операций. Следовательно, для каждого звена достаточно рассмотреть следующие два случая: в первом случае звено включают в текущее и все последующие решения до определения оптимального решения; во втором — звено исключают из дальнейшего рассмотрения. В нашем примере мы уже получили начальный узел дерева ветвления, соответствующий множеству всех маршрутов с нижней границей стоимости всех маршрутов, равной $H=48$, и верхней, равной $z_b(T)=73$.

5. Следующим шагом процедуры является выбор звена, на котором будет базироваться ветвление. Так как в каждой строке и в каждом столбце не единственный элемент имеет $c_{ij}=0$, то надо рассмотреть маршруты, не содержащие звено (i, j) . Пункт должен быть связан с некоторым другим пунктом и поэтому каждый маршрут, не содержащий узел (i, j) , должен содержать звено A , у которого стоимость не меньше минимального элемента i -й строки, не считая c_{ij} . Стоимость звена A обозначим A_i . Аналогично, чтобы в пункт j можно было бы попасть из некоторого другого города, то маршрут, не содержащий узел (i, j) , содержит звено B , у которого стоимость не меньше минимального элемента j -го столбца, не считая c_{ij} . Обозначим стоимость проезда по звену B через B_j , а сумму величин A_i и B_j через Φ_{ij} . Величину Φ_{ij} называют *вторичным штрафом* и она равна минимальному штрафу, которому мы подвергаемся, если не включаем звено (i, j) в оптимальный маршрут. Если штраф за неиспользование звена вычислить для всех звеньев, у которых $c_{ij}=0$, то можно сравнить соответствующие значения Φ_{ij} и включить в текущий маршрут звено (i, j) , за неиспользование которого мы заплатили бы максимальный штраф, т. е. включая звено (i, j) , мы получаем выигрыш в стоимости, равный максимальному значению Φ_{ij} . 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 значение новой нижней границы положить равным сумме значений текущей нижней границы и максимального штрафа за неиспользование звена (i, j) . Для определения максимального значения Φ_{ij} будем исследовать все элементы $c_{ij}=0$; при $c_{ij}\neq 0$ величина $\Phi_{ij}=0$. Данное утверждение справедливо в силу того, что если положить $c_{ij}=\infty$, а затем провести редукцию i -й строки и j -го столбца, то сумма вычитаемых констант будет равна Φ_{ij} . Для нашего случая значения A_i и B_j приведены в табл. 3.13, а значения Φ_{ij} (вторичный штраф) для

Таблица 3.13

Значения A_i и B_j

	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	$H = 48$	
B_j	1	0	9	0	2	0		

узлов, соответствующих $c_{ij} = 0$, — в табл. 3.14. Максимальное значение $\Phi_{ij} = 10$ соответствует звену (1, 4). Следовательно, в качестве базового звена ветвления выбираем звено (1, 4).

Таблица 3.14

Значения Φ_{ij}

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

6. Нижняя граница для маршрутов, не включающих звено (1, 4), равна $H + \Phi_{14} = 48 + 10 = 58$. Чтобы определить новую нижнюю границу для маршрутов, включающих звено (1, 4), необходимо преобразовать матрицу стоимости. Если мы включили в маршрут некоторое звено (k, l), то в дальнейшем мы не рассматриваем k -ю строку и l -й столбец. Кроме того, звено (k, l) является тогда звеном некоторого ориентированного цикла, и звено (k, l) не может принадлежать этому же маршруту. Последнее условие можно выполнить, положив $c_{lk} = \infty$.

Таблица 3.15

Преобразованная матрица стоимости

	1	2	3	5	6
2	1	∞	15	29	29
3	15	13	∞	5	0
4	∞	0	9	2	2
5	2	41	22	∞	0
6	13	0	0	0	∞

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	$H_1 = 1$	
B_j	2	0	9	2	0		

Из рассмотрения следует исключить и так называемые *запрещенные звенья* — звенья, с помощью которых в дальнейшем могут быть образованы циклы, включающие в себя неполное множество пунктов (могут быть образованы подмаршруты). Элементы матрицы стоимости, соответствующие этим звеньям, берут равными ∞ . Преобразованная матрица стоимости имеет вид (табл. 3.15). Запрещенных звеньев в данном случае не существует. Вторая матрица решений после редукции строк и столбцов, а также с указанием значений c_i , A_i и b_j для 2-й матрицы приведена в табл. 3.16. Нижняя граница для маршрута, включающего звено (1, 4), может быть вычислена как сумма всех новых вычитаемых констант H_1 и старой нижней границы, т. е. нижняя граница равна $48 + 1 = 49$.

Дерево решений для рассмотренных двух этапов имеет вид рис. 3.19. На рис. 3.19 показаны маршруты, включающие звено (1, 4) и не включающие это звено — (1, 4). В узлах дерева указаны нижние границы для каждого варианта маршрута.

7. С вычисления второй матрицы решений начинается вторая итерация решения. После второй редукции значения $\Phi_{21} = 16$; $\Phi_{36} = 5$; $\Phi_{42} = 2$; $\Phi_{56} = 2$; $\Phi_{62} = 0$; $\Phi_{63} = 9$ и $\Phi_{65} = 2$. Максимальным среди них является значение $\Phi_{21} = 16$, т. е. выбирают звено (2, 1). Новая нижняя граница для маршрута, не включающего звено (2, 1), равна $49 + 16 = 65$.

Чтобы определить множество маршрутов, содержащих звено (2, 1), вычеркнем во второй матрице стоимости вторую строку и первый столбец. Стоимость звена (1, 2) равна теперь ∞ , но в

Рис. 3.19. Дерево решений для первых двух этапов



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	$H_1 = 2$	
B_j	13	7	0	0		

третью матрицу решений этот элемент не входит. Но звено (4,2) теперь является запрещенным, поскольку оно могло бы образовать подмаршрут. Поэтому полагаем $C_{42} = \infty$. В табл. 3.17 приведена третья матрица решений после выполнения редукции.

Новая нижняя граница для маршрутов, содержащих звено (2,1), равна $49 + 2 = 51$. Ветвление на следующей итерации будет осуществляться из узла (2,1). Полный маршрут приведен на рис. 3.20.

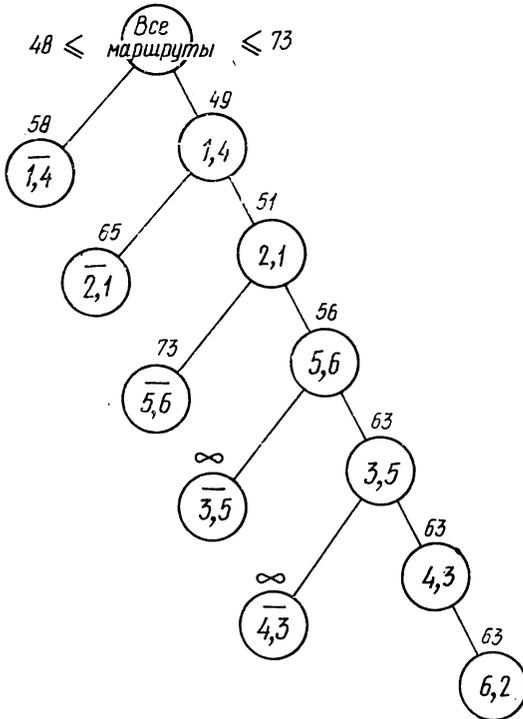


Рис. 3.20. Промежуточное решение

Этот маршрут — промежуточное решение.

8. Построенный полный маршрут будет оптимальным, если его длина не превосходит длины любого маршрута, соответствующего другим звеньям дерева. Длина построенного полного маршрута равна 63, а нижняя граница для узла (1,4), равная 58, меньше 63. Необходимо исследовать и подмножество маршрутов, которые не содержат звено (1,4) — исходная матрица стоимости (табл. 3.10). Но для того чтобы исключить все маршруты, содержащие звено (1,4), значение элемента c_{14} матрицы стоимости примем равным ∞ . Получим табл. 3.18. Процедуру анализа предыдущих промежуточных точек ветвле-

Матрица стоимости возврата

	1	2	3	4	5	6
1	∞	27	43	∞	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	∞

ния, которые могли бы определить более короткий маршрут, называют *возвратом*. Поэтому матрицу стоимости называют в данном случае *матрицей стоимости возврата*.

9. С новой матрицей, матрицей стоимости возврата, выполняют описанные процедуры ветвления и построения границ. Полученное при этом дерево изображено на рис. 3.21. Нахождение верхних границ не обязательно, но иногда позволяет сократить проводимые вычисления. Из рис. 3.21 следует, что нижняя граница даже неполного маршрута, не содержащего звено (1,4), превышает 63. Таким образом, маршрут, содержащий звено (1,4), является оптимальным. Оптимальный маршрут состоит из следующих звеньев или пар пунктов: (6,2); (4,3); (3,5); (5,6); (2,1); (1,4). Он является ориентированным циклом, стоимость проезда по которому равна 63, т. е. 630 условных денежных единиц.

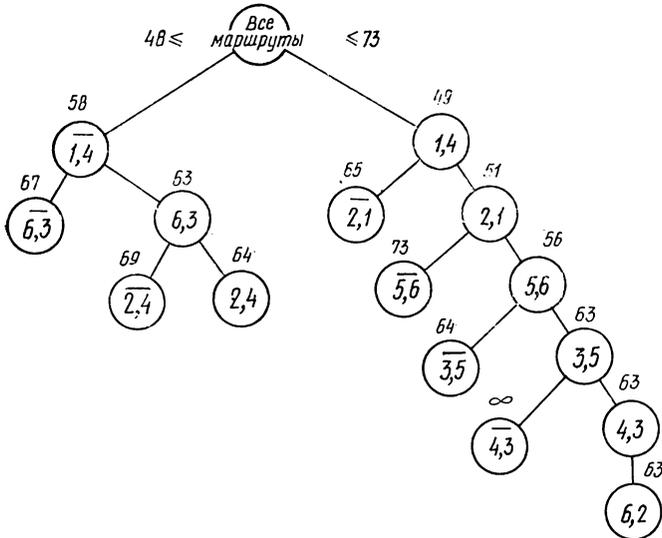


Рис. 3.21. Оптимальное решение

Задача коммивояжера не может быть непосредственно сформулирована и решена как задача линейного программирования. Основная особенность задачи коммивояжера заключается в том, что в ней требуется существование ориентированного цикла, в который ровно один раз входят все узлы сети.

3.10. Задача о многополюсной цепи с максимальной пропускной способностью

С задачей о многополюсном максимальном потоке тесно связана задача о *многополюсной цепи с максимальной пропускной способностью*. Алгоритм, описанный в § 3.8, 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 рассматривают лишь те пути или цепи, с помощью которых можно увеличить поток из одной точки в другую. Рассмотрим простую сеть, изображенную на рис. 3.22. (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: $X_{AB}=20$, $X_{AC}=20$, $X_{CB}=5$, $X_{BD}=25$, $X_{CD}=15$. Узлы A и D соединены тремя цепями, как показано ниже.

Рассмотрим 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 определяют последовательностью узлов $A \rightarrow B \rightarrow D$. 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 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 ($i, j = \overline{1, n}$).

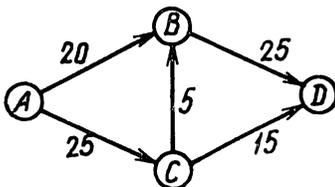


Рис. 3.22. Сеть для задачи о цепи с максимальной пропускной способностью

Шаг 2.

а. Для каждого $j = \overline{1, n}$ 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 $i, k \neq j; j = \overline{1, n}$.

Еще одна матрица, называемая матрицей маршрутов, необходима для определения внутренних узлов каждой цепи. Матрица маршрутов также имеет размеры $n \times n$, где 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:

$$r_{ik} = \begin{cases} j, & \text{если } d_{ik} < \min[d_{ij}, d_{jk}], \\ \text{остается неизменным,} & \text{если } d_{ik} \geq \min[d_{ij}, d_{jk}]. \end{cases}$$

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.

Задача. Производятся перевозки автотранспортом крупногабаритного оборудования. Требуется перевозить несколько видов оборудования с очень большим вертикальным габаритным размером. Это оборудование можно перевозить только по семи утвержденным маршрутам. Задача заключается в выборе маршрутов с максимально допустимыми подмостовыми зазорами.

Каждый элемент матрицы, приведенной в табл. 3.19, равен высоте проезда под мостом. Необходимо получить информацию о максимально допустимых вертикальных габаритных размерах грузов, которые можно транспортировать из каждого пункта погрузки в каждый пункт выгрузки.

Таблица 3.19
Транспортировка крупногабаритного груза

		В пункт						
		1	2	3	4	5	6	7
Из пункта	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

Решение. Для каждой пары узлов определяют 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	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	11	0	$-\infty$	12	2	$-\infty$	$-\infty$
3	30	$-\infty$	0	19	$-\infty$	4	$-\infty$
4	$-\infty$	12	19	0	11	9	$-\infty$
5	$-\infty$	2	0	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	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	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	11	0	11	12	2	$-\infty$	$-\infty$
3	30	11	0	19	$-\infty$	4	$-\infty$
4	$-\infty$	12	19	0	11	9	$-\infty$
5	$-\infty$	2	$-\infty$	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	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

Приведем сразу результаты последней итерации 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 габаритный размер перевозимого груза можно определить непосредственно из последней матрицы пропускных способностей. Оптимальный маршрут также может быть построен с помощью последней матрицы маршрутов. Например, максимальную про-

пусковую способность цепи из пункта 1 в пункт 4 задают значением элемента $d_{14}^* = 19$. Соответствующий маршрут строят 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.



Отметим, что дуги, соединяющей узлы 1 и 4, не существует.

Глава 4. ОСНОВЫ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

4.1. Условия применимости динамического программирования

Как было показано в гл. 1, непосредственное применение только необходимых признаков в задачах математического программирования для нахождения оптимального решения, как правило, не дает нужных результатов. Во-первых, система уравнений, вытекающая из необходимых признаков, оказывается разрешимой только в простейших случаях. Чаще бывает легче непосредственно «нащупать» максимум целевой функции, чем решить такую систему уравнений. Во-вторых, указанный способ вовсе не гарантирует нахождение решения во всех случаях. Даже если составленная система уравнений может быть решена, отыскание абсолютного экстремума целевой функции требует системы проверок, тем более сложной, чем больше аргументов у функции.

Наконец в ряде практических случаев целевую функцию вообще нельзя дифференцировать, например когда элементы $x = \{x_1, x_2, \dots, x_m\}$ представляют собой не непрерывно изменяющиеся, а дискретные величины.

Все эти обстоятельства приводят к тому, что применение классических методов математического анализа или вариационного исчисления в большинстве задач планирования оказывается неэффективным: при этом первоначально поставленная задача отыскания экстремума приводит к таким вторичным задачам, которые оказываются не проще исходной, а зачастую сложнее.

В гл. 1 было также показано, что все задачи математического оптимального программирования в зависимости от вида целевой функции и ограничений могут быть разбиты на классы, характеризуемые своими методами решения. К примеру, линейное программирование изучает класс задач, в которых и целевые функции, и ограничения линейны.

Вместе с тем решение многих таких задач может быть упрощено, если развернуть процесс планирования поэтапно, т. е. использовать *метод динамического программирования*. Идея метода состоит в том, что отыскание точек экстремума целевой функции мно-

гих переменных заменяется многократным отысканием точек экстремума функции одного или небольшого числа переменных.

Итак, динамическое программирование есть поэтапное планирование многошагового процесса, при котором на каждом этапе оптимизируется только один шаг. Причем управление на каждом шаге должно выбираться с учетом всех его последствий в будущем. Динамическое программирование — это планирование с учетом перспектив, с учетом интересов всей задачи.

В задачах, решаемых методом динамического программирования, значение целевой функции (оптимизируемого критерия), достигнутое за весь процесс, получают простым суммированием частных значений $f_i(x)$ того же критерия, достигнутых на отдельных шагах, т. е.

$$f(x) = \sum_{i=1}^m f_i(x).$$

Если критерий $f(x)$ обладает этим свойством, то он называется *аддитивным*.

Во многих практических задачах критерий $f(x)$ — аддитивен; если в первоначальной постановке задачи критерий не аддитивен, то постановку задачи надо видоизменить так, чтобы он стал аддитивным. К примеру если рассматривают критерий $f(x)$, представленный в виде произведения выигрышей, достигаемых на отдельных этапах: $f(x) = f_1(x) \cdot f_2(x) \cdot \dots \cdot f_m(x)$ (такой критерий называют *мультиплексным*), то можно преобразовать его к аддитивному, прологарифмировав,

$$\lg f(x) = \sum_{i=1}^m \lg f_i(x).$$

Обозначим $V = \lg f(x)$, $V_i = \lg f_i(x)$. Получим новый критерий $V = \sum_{i=1}^m V_i$, обладающий свойством аддитивности и обращающийся в максимум (минимум) одновременно с $f(x)$.

Рассмотрим общую схему решения задач, имеющих аддитивный критерий.

Пусть процесс управления состоит из m шагов. На каждом i -м шаге управление x_i переводит систему из состояния S_{i-1} , достигнутого в результате $(i-1)$ -го шага, в новое состояние S_i , которое зависит от состояния S_{i-1} и выбранного управления x_i .

$$S_i = S_i(S_{i-1}, x_i).$$

Здесь существенно, чтобы новое состояние S_i зависело только от состояния S_{i-1} и управления x_i и не зависело от того, каким образом система пришла в состояние S_{i-1} . В крайнем случае это достигается увеличением числа состояний системы. В понятие «состояние системы» вводят те параметры, от которых зависит будущий результат.

В теории динамического программирования, когда рассматривают стратегии, зависящие только от текущего состояния, оптимальную стратегию характеризуют очень просто.

Принцип оптимальности. Оптимальная стратегия обладает тем свойством, что, каковы бы ни были первоначальные состояние и решение, последующее решение должно определять оптимальную стратегию относительно состояния, полученного в результате начального решения.

Рассмотрим задачу о максимизации функции дохода $f(x)$ на m -шаговом процессе.

Под влиянием управлений x_1, x_2, \dots, x_m система переходит из начального состояния S_0 в конечное $S_{\text{кон}}$. За m шагов получают выигрыш

$$f(x) = \sum_{i=1}^m f_i(S_{i-1}, x_i),$$

где $f_i(S_{i-1}, x_i)$ — выигрыш на i -м шаге. Принцип оптимальности позволяет заключить, что при любом начальном управлении x_1

$$\begin{aligned} f(x) &= f_1(S_0, x_1) + [f_2(S_1, x_2) + \dots + f_m(S_{m-1}, x_m)] = \\ &= f_1(S_0, x_1) + f_{m-1}[S_1(S_0, x_1)]. \end{aligned}$$

Поскольку данное соотношение справедливо для всех начальных решений x_1 , то, чтобы найти максимальный доход, надо найти максимум по x_1 значения $f(x)$:

$$f_m(S_0) = \max_{x_1} f(x) = \max_{x_1} [f_1(S_0, x_1) + f_{m-1}[S_1(S_0, x_1)]], \quad m \geq 1,$$

где $f_1(S_0) = \max_{x_1} f_1(S_0, x_1)$. Получили основное функциональное уравнение динамического программирования. Его же можно получить другим путем:

$$\max_{x_1, \dots, x_m} f(x) = \max_{x_1} [\max_{x_2, \dots, x_m} f(x)].$$

Следовательно,

$$\begin{aligned} f_m(S_0) &= \max_{x_1, \dots, x_m} [f_1(S_0, x_1) + f_2(S_1, x_2) + \dots + f_m(S_{m-1}, x_m)] = \\ &= \max_{x_1} \max_{x_2, \dots, x_m} [f_1(S_0, x_1) + f_2(S_1, x_2) + \dots + f_m(S_{m-1}, x_m)] = \\ &= \max_{x_1} [f_1(S_0, x_1) + \max_{x_2, \dots, x_m} [f_2(S_1, x_2) + \dots + f_m(S_{m-1}, x_m)]] = \\ &= \max_{x_1} [f_1(S_0, x_1) + f_{m-1}(S_1(S_0, x_1))]. \end{aligned}$$

Согласно этому выражению алгоритм получения решения в динамическом программировании — это последовательность функций дохода $\{f_n(S_{n-1})\}$ или последовательность стратегий $\{x_n(S_{n-1})\}$. Эти последовательности определяют одна другую. Причем, имеется только одна последовательность оптимальных функций дохода,

хотя может быть много последовательностей оптимальных стратегий, которые приведут к тому же максимальному доходу.

В динамическом программировании, планируя многоэтапную процедуру, управление на каждом шаге выбирают с учетом будущего. И только на одном шаге — последнем — нет такой необходимости и этот последний шаг можно спланировать так, чтобы он принес наибольшую выгоду.

Спланировав оптимальным образом последний шаг, к нему присоединяют предпоследний и находят согласно основному функциональному уравнению наибольшую выгоду на этих двух шагах и т. д. Поэтому в динамическом программировании процесс разворачивают от конца к началу. А как спланировать последний шаг, если мы не знаем, чем кончился предпоследний? Для этого делают разные предположения о том, чем кончится предпоследний шаг и для каждого допущения выбирают управление на последнем шаге и запоминают его до конца решения задачи. Этот процесс повторяют на каждом шаге. Такое оптимальное управление, выбранное при известном условии, чем кончится предыдущий шаг, называют *условным*.

Рассмотрим на примерах схему этой процедуры.

4.2. Задача об оптимальной загрузке транспортного средства неделимыми предметами

В § 1.1 мы сформулировали задачу о бабе, отправляющейся на рынок и решающей, сколько ей надо взять *живых* гусей, кур и уток, чтобы получить наибольшую выручку. Мы знаем и то, что это задача целочисленного линейного программирования. Теперь рассмотрим, как подобные задачи решаются с помощью динамического программирования.

Задача. Пусть в самолет требуется погрузить 4 вида предметов, чтобы эффект от этих предметов (например, стоимость) был максимальным. Грузоподъемность самолета равна ω ; P_1, P_2, P_3, P_4 — вес соответствующей единицы различных предметов; а v_1, v_2, v_3, v_4 — соответствующая эффективность единицы каждого предмета; x_1, x_2, x_3, x_4 — число предметов различных видов, взятых на борт самолета.

Решение. Целевая функция задачи —

$$\sum_{i=1}^4 x_i v_i \equiv x_1 v_1 + x_2 v_2 + x_3 v_3 + x_4 v_4 \rightarrow \max .$$

Ограничение

$$\sum_{i=1}^4 x_i P_i \equiv x_1 P_1 + x_2 P_2 + x_3 P_3 + x_4 P_4 \leq \omega .$$

Эту задачу называют в литературе задачей о «рюкзаке» вследствие следующей гипотетической ситуации. Турист, собираясь в поход, должен выбрать, какие предметы ему взять с собой. Каждый

предмет имеет свой вес и свою ценность. Требуется, чтобы общий вес рюкзака с выбранными предметами не превышал некоторого заданного веса, а их общая ценность была максимальной.

Для численного решения примем, что $w=83$, $P_1=24$, $P_2=22$, $P_3=16$, $P_4=10$, $v_1=96$, $v_2=85$, $v_3=50$, $v_4=20$ некоторых единиц.

Очевидно, что это задача целочисленного программирования.

Применим для ее решения многошаговую процедуру принятия решения (динамическое программирование). Согласно общей схеме решения задач динамического программирования будем рассматривать два этапа. На первом этапе найдем возможные оптимальные варианты при последовательной оптимизации по одной переменной, а на втором этапе из этих «заготовок» выберем оптимальное решение нашей задачи.

Первый этап будет содержать 4 шага. На первом шаге находят возможные оптимальные варианты загрузки самолета только предметами первого вида. Необходимо найти и запомнить значения x_1 и соответствующую им максимальную стоимость груза $f_1(w)$ при различных возможных значениях w . В этом случае максимальная эффективность груза определится как

$$f_1(w) = \max_{x_1} [x_1 v_1]$$

при условии $x_1 P_1 \leq w$, $x_1 = 0, 1, 2, \dots$. Так как $x_1 \leq w / P_1$, то для нахождения максимума $f_1(x)$ надо x_1 взять возможно большим, т. е.

$x_1 = \left[\frac{w}{P_1} \right]$ — наибольшее целое число, не превосходящее $\frac{w}{P_1}$, и та-

ким образом $f_1(w) = \left[\frac{w}{P_1} \right] x_1$. Через $[z]$ обозначают наибольшее целое число, не превосходящее z .

Зададимся значениями w с некоторым шагом и найдем для них x_1 и $f_1(w)$. Очевидно, если грузоподъемность самолета меньше 24 ед., то ни одного предмета первого вида погрузить нельзя. При грузоподъемности от 24 до 47 ед. можно погрузить 1 ед. предмета первого вида и т. д. Результаты представим в виде таблиц. Значения w , $f_1(w)$ и x_1 приведены в табл. 4.1. В таблицах принят более широкий предел для w — до 87 ед.

Таблица 4.1

w	$f_1(w)$	x_1	w	$f_1(w)$	x_1
0—23	0	0	48—71	192	2
24—47	96	1	72—87	288	3

Проведем оптимизацию на втором шаге. Рассмотрим эффективность груза, если загрузить самолет предметами первого и второго типов. Максимальную эффективность загрузки обозначим через $f_2(w)$. Если предметов второго вида взято x_2 , то вес предме-

тов первого вида должен быть не больше чем $\omega - P_2 x_2$. Максимальная эффективность предметов первого вида при этом

$$f_1(\omega - x_2 P_2) = \max_{x_1} [(\bar{\omega} - x_2 P_2) v_1],$$

а общая эффективность груза —

$$f_{12} = x_2 v_2 + f_1(\omega - x_2 P_2).$$

Тогда

$$f_2(\omega) = \max_{x_2} \{x_2 v_2 + f_1(\omega - x_2 P_2)\}; \quad 0 \leq x_2 \leq \left\lfloor \frac{\omega}{P_2} \right\rfloor. \quad (4.1)$$

Максимум этого выражения ищут только по x_2 . Но мы не знаем, какой вес ω могут занимать предметы второго вида. Поэтому мы должны рассмотреть выражение (4.1) для всевозможных значений ω от 0 до 83. При вычислении $f_1(\omega - x_2 P_2)$ воспользуемся полученными результатами (табл. 4.1).

Например, возьмем $\omega = 46$ ед. Для x_2 возможны значения 0, 1, 2; соответствующая эффективность от предметов второго вида 0, 85, 170, а для предметов первого вида составит 46, 24, 2 ед. Из табл. 4.1 для $\omega = 46, 24, 2$ ед. находим $f_1(\omega) = 96; 96; 0$ ед. Складываем соответствующие значения $0 + 96 = 96$ ед.; $85 + 96 = 181$ ед., $170 + 0 = 170$ ед. и выбираем наибольшее 181 ед. Оно получено для $x_2 = 1$. Для $\omega = 46$ заносим в табл. 4.2 $x_2 = 1, f_2(\omega) = 181$. Результаты вычисления $f_2(\omega)$ и x_2 приведены в табл. 4.2.

Т а б л и ц а 4.2

ω	$f_2(\omega)$	x_2	ω	$f_2(\omega)$	x_2
0—21	0	0	48—65	192	0
22—23	85	1	66—67	255	3
24—43	96	0	68—69	266	2
44—45	170	2	70—71	277	1
46—47	181	1	72—87	288	0

Из табл. 4.2 следует, что при грузоподъемности транспортного средства до 21 ед. ничего в него погрузить нельзя, при грузоподъемности 22—23 ед. можно погрузить только один предмет второго вида, при грузоподъемности 24—43 ед. — либо один предмет первого вида, либо один предмет второго вида. Максимальной эффективностью груза будет при погрузке предметами первого вида. При грузоподъемности 44—45 ед. можно погрузить либо один предмет первого вида, либо два предмета второго вида. Большая эффективность груза будет в последнем случае и $f_2(\omega) = 170$. При грузоподъемности 46—47 ед. можно погрузить один предмет первого вида, до двух предметов второго вида или же по одному предмету первого и второго видов. Эффективность груза в последнем варианте — максимальная, и т. д.

Приступим к оптимизации на третьем шаге. Будем загружать транспортное средство предметами первых трех видов. Требуется максимизировать по x_3

$$f_{123} \equiv x_3 v_3 + f_2(\omega - x_3 P_3); \quad f_3(\omega) = \max_{x_3} f_{123};$$

$$0 \leq x_3 \leq \left\lfloor \frac{\omega}{P_3} \right\rfloor.$$

Задаем значение ω и для каждого такого значения получаем максимум f_{123} по x_3 . Значения $f_2(\omega)$ берут в табл. 4.2. Например, пусть $\omega=38$ ед. Возможное значение $x_3=0; 1; 2$ ед. и соответствующая эффективность от предметов третьего вида 0; 50; 100 ед. На предметы первого и второго видов остается соответственно вес 38; 22; 6 ед. По табл. 4.2 находим $f_2(\omega)=96; 85; 0$ ед. Суммарная эффективность 96; 135; 100 ед. Максимальное значение эффективности при $\omega=38$ ед. равно 135 ед. при $x_3=1$. Результаты расчетов помещены в табл. 4.3.

Таблица 4.3

ω	$f_2(\omega)$	x_3	ω	$f_3(\omega)$	x_3
0—15	0	0	44—45	170	0
16—21	50	1	46—47	181	0
22—23	85	0	48—63	192	0
24—31	96	0	64—69	242	1
32—37	100	2	70—71	277	0
38—39	135	1	72—87	288	0
40—43	146	1			

Оптимизируем на последнем (четвертом) шаге. Получим оптимальный вес груза, который может принять самолет ω^* .

Зададимся значением ω , которое может быть занято четвертым грузом, и вычислим $x_4, f_4(\omega)$ (табл. 4.4).

Таблица 4.4

ω	$f_4(\omega)$	x_4	ω	$f_4(\omega)$	x_4
0—9	0	0	46—47	181	0
10—15	20	1	48—57	192	0
16—21	50	0	58—63	212	1
22—23	85	0	64—69	242	0
24—33	96	0	70—71	277	0
34—37	116	1	72—81	288	0
38—39	135	0	82—87	308	1
40—45	146	0			

Начинаем второй этап — нахождение оптимального решения поставленной задачи. Максимальное значение $f_4(\omega)$ — это и есть ω^* , соответствующее значение аргумента x_4 — количество груза четвертого вида, который берет самолет (x_4^*)

$$\omega^* = \max f_4(\omega) = 308; x_4^* = 1.$$

Вес, занятый предметами четвертого вида, будет $x_4^* P_4 = 10$ ед.

На остальные три вида груза остается $\omega - 10 = 73$ ед. Входим с этим значением в табл. 4.3 и получаем $x_3^* = 0$, а затем $x_2^* = 0$; $x_1^* = 3$.

Если еще раз вернуться к табл. 4.1—4.4, то увидим, что они содержат «заготовки» для оптимальной загрузки самолета любой грузоподъемности до ω ($\omega = 87$ ед.) указанными предметами. Таким образом, мы решили не только поставленную задачу, но и ряд родственных задач. С одной стороны это хорошо, а с другой — в памяти компьютера необходимо держать табл. 4.1—4.4: столбцы $f_1(\omega)$, $f_2(\omega)$, $f_3(\omega)$, $f_4(\omega)$ на протяжении следующего шага, а столбцы x_1 , x_2 , x_3 , x_4 и соответствующие им ω — на протяжении всего решения. Для сложных задач последнее обстоятельство имеет существенное значение и ограничивает возможности рассмотренного метода.

4.3. Задача о вкладе средств в производство

Задача. Планируется деятельность двух отраслей производства (например, швейного цеха и цеха по ремонту обуви) сроком на 5 лет ($m = 5$). Функция вклада в производство средств $\varphi(x) = 0,75x$; $\psi(y) = 0,3y$, т. е. если средства z вкладывать только в первую отрасль, то по годам средств будет вкладываться $z \cdot 0,75$; $z \cdot 0,75^2$; $z \cdot 0,75^3$, ... Соответственно, вкладывая средства z только во вторую отрасль, получаем $z \cdot 0,3$; $z \cdot 0,3^2$; $z \cdot 0,3^3$, ...

Функции дохода как функции от объема вкладываемых средств y и x имеют вид

$$f(x) = 1 - e^{-x}; g(y) = 1 - e^{-2y}.$$

Требуется распределить имеющиеся ресурсы $z = 2$ между отраслями (цехами) по годам так, чтобы получить максимальный доход.

Решение. Решение задачи не удастся получить полностью в аналитическом виде, решим ее графически. Пусть к началу пятого года количество средств равно z_4 и на пятый год выделено средств первому производству x_5 , тогда второму производству будет выделено $y_5 (y_5 = z_4 - x_5)$. Чтобы найти *условное оптимальное управление* на пятом шаге $x_5(z_4)$, нужно для каждого z_4 найти максимум функции:

$$f_5 = 1 - e^{-x_5} + 1 - e^{-2(z_4 - x_5)} = 2 - [e^{-x_5} + e^{-2(z_4 - x_5)}].$$

При фиксированном z_4 максимум функции f_5 достигается либо при $x_5=0$, либо внутри отрезка $(0, z_4)$. Максимум внутри отрезка будет в точке x_5^* , которую находим из условия

$$\frac{\partial f_5}{\partial x_5} = 0; e^{-x_5} - 2e^{-2(z_4-x_5)} = 0;$$

$$x_5 = \frac{1}{3}(2z_4 - \lg 2); x_5 \geq 0.$$

То есть при $z_4 > \ln 2/2 = 0,347$ максимум достигается в точке

$$x_5^*(z_4) = \frac{1}{3}(2z_4 - \ln 2).$$

При $z_4 \leq \ln 2/2$ максимум будет в точке $x_5^*(z_4) = 0$.

Итак, условное оптимальное управление на пятом шаге

$$x_5^*(z_4) = \begin{cases} 0 & \text{при } z_4 \leq \frac{\ln 2}{2}; \\ \frac{1}{3}(2z_4 - \ln 2) & \text{при } z_4 > \frac{\ln 2}{2}. \end{cases} \quad (4.2)$$

Построим на графике (рис. 4.1) зависимость $x_5^* = x_5^*(z_4)$ и функцию $j_5^* = f_5^*(z_4)$. При построении зависимости $f_5^*(z_4)$ пользуемся (4.2).

Перейдем к оптимизации на четвертом шаге. Запас средств после третьего шага обозначим z_3 . Определим, в каких пределах может изменяться z_3 . Наибольшее значение z_3 будет достигнуто, если все средства вложить в первую отрасль:

$$z_3; \max = z \cdot 0,75^3 = 2 \cdot 0,75^3 = 0,844.$$

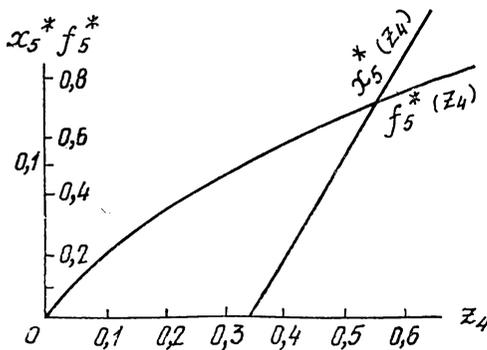


Рис. 4.1. Зависимость f_5^* и x_5^* от z_4

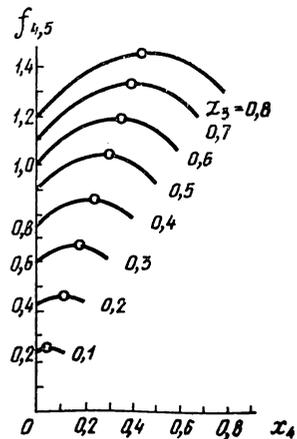


Рис. 4.2. Зависимость j_{45} от x_4

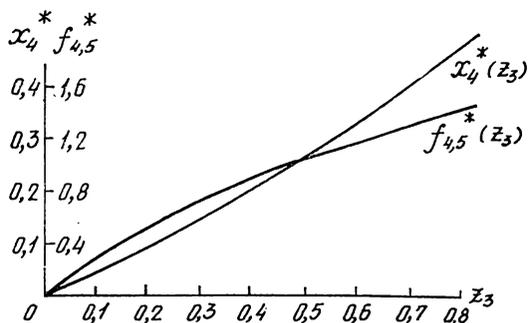


Рис. 4.3. Зависимость f_{45}^* и x_4^* от z_3

Наименьший запас средств соответствует случаю, когда все средства на трех шагах вложены во вторую отрасль:

$$z_3; \min = z \cdot 0,3^3 = 0,054.$$

Будем рассматривать теперь набор значений z_3 от 0,1 до 0,8 с шагом 0,1. Для каждого значения z_3 найдем *условное оптимальное управление* на четвертом шаге $x_4^*(z_3)$ и *условный оптимальный доход* на двух последних шагах $f_4^*(z_3)$.

Для этого построим серию кривых, показывающих выигрыш на двух последних шагах при любом управлении на четвертом и при оптимальном на пятом:

$$f_{45} = f_4(z_3, x_4) + f_5^*(z_4) = f_4(z_3, x_4) + f_5^*[0,75x_4 + 0,3(z_3 - x_4)];$$

$$f_4(z_3, x_4) = 2 - [e^{-x_4} + e^{-2(z_3 - x_4)}].$$

$f_5^*(z_4)$ берем из рис. 4.1.

Значения $f_{45}(x_4)$ приведены на рис. 4.2. Для каждого значения z_3 максимальная ордината определяет *условный максимальный выигрыш* на двух последних шагах $f_4^*(z_3)$, а абсцисса — *условное оптимальное управление* $x_4^*(z_3)$.

На рис. 4.3 построены графики зависимостей $f_{45}^*(z_3)$ и $x_4^*(z_3)$.

Аналогично решают задачу для третьего и второго шагов. Результаты приведены на рис. 4.4—4.7.

Спланируем первый шаг. Доход на пяти шагах (при любом управлении на первом шаге и оптимальном на последующих)

$$f_{12345}(x_1) = f_1(z, x_1) + f_{2345}^*(z_1) = 2 - [e^{-x_1} + e^{-2(z - x_1)}] + f_{2345}^*(z_1);$$

$$z_1 = 0,75x_1 + 0,3(z - x_1).$$

На первом шаге имеющиеся ресурсы известны: $z=2$. Поэтому $f_{12345}(x_1)$ опишется одной линией (рис. 4.8). Из рис. 4.8 видно, что *максимальный доход* будет $f_{12345}^* = 4,35$; *оптимальное управление*

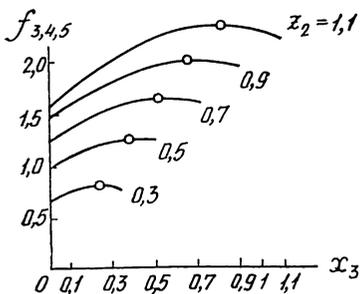


Рис. 4.4. Зависимость f_{345} от x_3

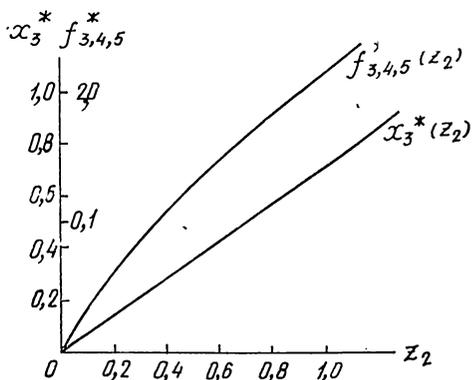


Рис. 4.5. Зависимость f_{345}^* и x_3^* от z_2

на первом шаге $x_1^* = 1,60$. По известному оптимальному управлению на первом шаге мы можем определить запас средств к концу этого шага, что в свою очередь определит оптимальное управление на следующем шаге.

Запас средств к концу первого шага:

$$z_1^* = 0,75 x_1^* + 0,3(z - x_1^*) = 1,32.$$

Оптимальное управление на втором шаге тогда (рис. 4.7) $x_2^* = 1,02$. Остаток средств к концу второго шага будет

$$z_2^* = 0,75 x_2^* + 0,3(z_1^* - x_2^*) = 0,86;$$

и далее $x_3^* = 0,62$, $z_3^* = 0,54$, $x_4^* = 0,30$, $z_4^* = 0,30$, $x_5^* = 0$.

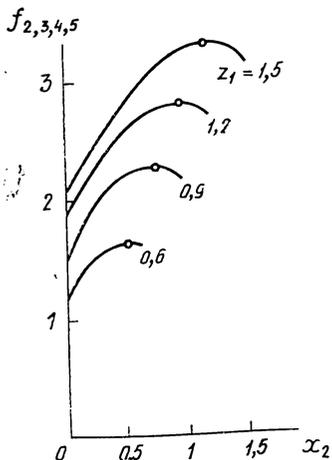


Рис. 4.6. Зависимость f_{2345} от x_2

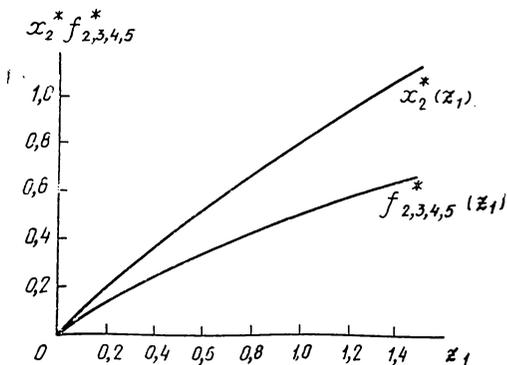


Рис. 4.7. Зависимость f_{2345}^* и x_2^* от z_1

Получим оптимальное управление, показывающее, сколько средств надо вкладывать в первую отрасль:

$$x^* = \{x_1^*, x_2^*, x_3^*, x_4^*, x_5^*\} = \\ = \{1,60; 1,02; 0,62; 0,30; 0\}.$$

Автоматически получаем количество средств, вкладываемых во вторую отрасль

$$y_1^* = z - x_1^* = 0,40;$$

$$y_2^* = z_1^* - x_2^* = 0,30;$$

$$y_3^* = z_2^* - x_3^* = 0,24; \quad y_4^* = z_3^* - x_4^* = 0,24;$$

$$y_5^* = z_4^* - x_5^* = 0,30;$$

Определим остаток средств: $0,30 \cdot 0,3 = 0,09$.

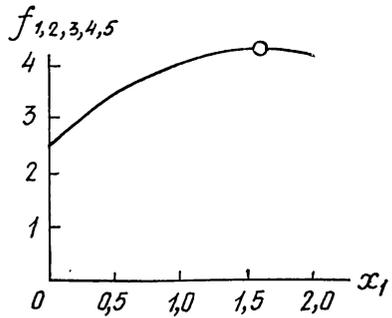


Рис. 4.8. Зависимость f_{12345} от x_1

4.4. Задача о распределении средств поражения

Рассмотрим еще одну задачу, которая аналогична задаче о резервировании средств, — о распределении средств поражения по обороняющимся целям.

Задача. Планируются боевые действия самолетами по обороняющимся целям. Цели эшелонированы по глубине территории на четырех параллельных рубежах. Перед тем, как выйти на данный рубеж, самолеты проходят зону действия огневых средств этого рубежа, где подвергаются обстрелу со стороны последних. Огневые средства каждого рубежа могут вести огонь не только по средствам поражения, направляющимся непосредственно на цели данного рубежа, но и по тем самолетам, которые проходят через зону действия, направляясь на следующие рубежи.

Вероятность поражения одного самолета, пролетающего зону действия орудий i -го рубежа,

$$v_i = 1 - e^{-\alpha_i \bar{N}_i},$$

где \bar{N}_i — среднее число орудий, сохранивших боеспособность на i -м рубеже; α_i — эффективность стрельбы орудий по самолетам.

Среднее число орудий i -го рубежа, пораженных волной самолетов, направленной на цели этого рубежа,

$$Q_i = N_i \left(1 - e^{-\frac{\bar{v}_j}{N_i} P_i} \right),$$

где N_i — число орудий на i -м рубеже; \bar{v}_j — среднее число самолетов в j -й волне, сохранивших боевые свойства до i -го рубежа;

P_i — средняя вероятность поражения одного орудия i -го рубежа атакующим его самолетом.
 Имеется N орудий на всех рубежах:

$$N = \sum_{i=1}^4 N_i, \quad i = \overline{1,4};$$

где N_i — число орудий на i -м рубеже: $N_1=10, N_2=12, N_3=15, N_4=10, N=47$.

Имеется $z_0=80$ самолетов, которые надо распределить на четыре волны так, чтобы сделать максимальным среднее число поражений на всех рубежах:

$$f(x) = \sum_{i=1}^4 f_i(x),$$

где $f_i(x)$ — среднее число поражений на i -м рубеже.

Волны распределены по времени так, чтобы к моменту подлета следующей волны предшествующая ей волна самолетов успела выполнить боевое задание.

Пусть

$$P_1 = 0,4, \quad P_2 = 0,5, \quad P_3 = 0,4, \quad P_4 = 1,0, \\ \alpha_1 = 0,05, \quad \alpha_2 = 0,04, \quad \alpha_3 = 0,04, \quad \alpha_4 = 0,05.$$

Решение. Будем пользоваться общей схемой решения задач методом динамического программирования. Оптимизируем на четвертом (последнем) шаге. К зоне действия орудий 4-го рубежа подойдет z_3 самолетов. Очевидно, что все они должны быть направлены на поражение орудий 4-го рубежа. Условное оптимальное управление на 4-м шаге

$$x_4^*(z_3) = z_3.$$

На 4-м рубеже N_4 орудий; он не подвергался бомбежке, т. е.

$$\bar{N}_4 = N_4.$$

Эти самолеты поразят на 4-м рубеже $f_4^*(z_3)$ орудий:

$$f_4^*(z_3) = N_4 \left[1 - e^{-\frac{\bar{v}_4}{N_4} P_4} \right],$$

где $\bar{v}_4(z_3) = z_3(1 - v_4)$; v_4 — вероятность поражения одного самолета на 4-м рубеже;

$$v_4 = 1 - e^{-\alpha_4 N_4},$$

т. е.

$$\bar{v}_4(z_3) = z_3 e^{-\alpha_4 N_4}$$

Задаваясь рядом значений z_3 , получаем $f_4^*(z_3)$ (рис. 4.9). Вообще-то следовало бы брать значения z_3 в интервале $(0; 80)$, но в этой задаче в этом нет необходимости.

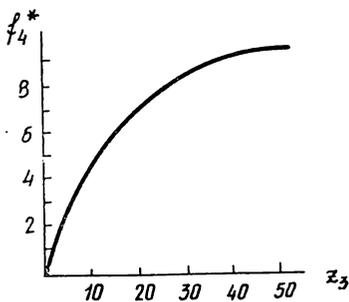


Рис. 4.9. Зависимость f_4^* от z_3

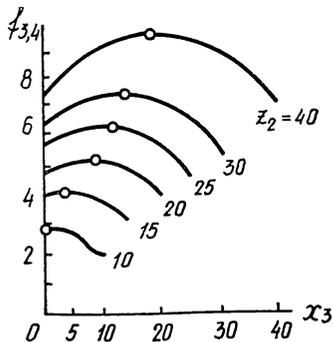


Рис. 4.10. Зависимость f_{34} от x_3

Оптимизируем на третьем шаге. Зададимся рядом значений z_2 самолетов, преодолевших 1-й и 2-й рубежи:

$$z_2 \in (10; 40).$$

Для каждого значения z_2 подсчитаем суммарный выигрыш: на третьем шаге при любом управлении, на четвертом — при оптимальном:

$$f_{34} = Q_3(x_3) + f_4^*(z_3).$$

Из z_2 самолетов надо выделить x_3 на подавление орудий 3-го рубежа, а $(z_2 - x_3)$ — направить на 4-й рубеж через зону огня 3-го рубежа. Условное оптимальное управление $x_3(z_2)$ найдем из условия максимального выигрыша на двух последних шагах.

Здесь $Q_3(x_3)$ — среднее число орудий, пораженных на 3-м рубеже выделенными для этой цели x_3 самолетами. При таком управлении до 4-го рубежа дойдут z_3 самолетов. Имеем

$$Q_3(x_3) = N_3 \left(1 - e^{-\frac{\bar{v}_3 P_3}{N_3}} \right),$$

$$\text{где } \bar{v}_3 = x_3(1 - v_3) = x_3(1 - 1 + e^{-\alpha_3 N_3}) = x_3 e^{-\alpha_3 N_3}.$$

Подсчитаем среднее число самолетов из оставшихся $(z_2 - x_3)$ самолетов, прошедших через огонь 3-го рубежа и дошедших до 4-го рубежа. На 3-м рубеже осталось в действии $\bar{N}_3 = N_3 - Q_3(x_3)$ орудий. Тогда среднее число дошедших самолетов до 4-го рубежа из полного числа $(z_2 - x_3)$ будет

$$z_3 = (z_2 - x_3) e^{-\alpha_3 \bar{N}_3},$$

$$\bar{N}_3 = \bar{N}_3(x_3).$$

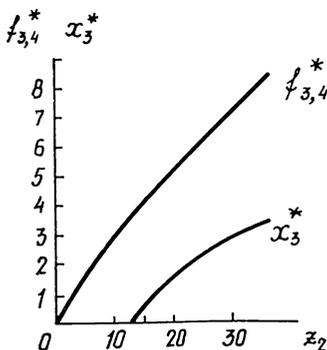


Рис. 4.11. Зависимость f_{34}^* и x_3^* от z_2

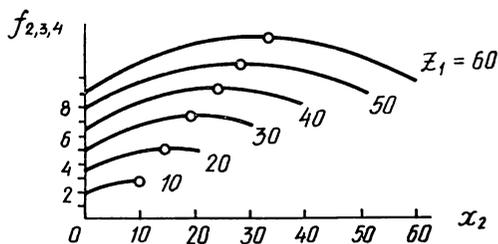


Рис. 4.12. Зависимость f_{234} от x_2

Следовательно,

$$f_{34}^*(z_2) = \max_{0 < x_3 < z_2} \left\{ N_3 \left[1 - e^{\frac{-x_3 e^{-a_3 N_3 P_3}}{N_3}} \right] + f_4^* [(z_2 - x_3) e^{-a_3 \bar{N}_3(x_3)}] \right\}.$$

Задавшись значениями z_2 , для каждого из них [и взяв значения $f_4^*(z_3)$ из рис. 4.9] получим $f_{34}(x_3)$ (рис. 4.10). Отмечая точки максимума $f_{34}(x_3)$, для каждого z_2 строим из рис. 4.10 зависимость f_{34}^* и x_3^* от z_2 (рис. 4.11).

Оптимизируем на втором шаге. Процедура аналогична оптимизации на третьем шаге.

Для разных значений z_1 (число самолетов, преодолевших 1-й рубеж) вычислим (рис. 4.12)

$$f_{234} = Q_2(x_2) + f_{34}^*(z_2).$$

Здесь

$$Q_2(x_2) = N_2 \left[1 - e^{\frac{-\bar{v}_2 P_2}{N_2}} \right];$$

$$\bar{v}_2 = x_2 e^{-a_2 N_2};$$

$$z_2 = (z_1 - x_2) e^{-a_2 \bar{N}_2};$$

$$\bar{N}_2 = N_2 - Q_2(x_2).$$

Величину $f_{34}^*(z_2)$ берем из рис. 4.11. Строим зависимость f_{234}^* и x_2^* от z_1 (рис. 4.13).

Оптимизация на первом шаге. Число самолетов, которые налетают на первый рубеж, задано $z_0=80$. Выигрыш составит

$$f_{1234}(x_1) = Q_1(x_1) + f_{234}^*(z_1);$$

$$Q_1(x_1) = N_1 \left(1 - e^{-\frac{\bar{v}_1}{N_1} P_1} \right);$$

$$\bar{v}_1 = x_1 e^{-\alpha_1 N_1};$$

$$z_1 = (z_0 - x_1) e^{-\alpha_1 \bar{N}_1};$$

$$\bar{N}_1 = N_1 - Q_1(x_1).$$

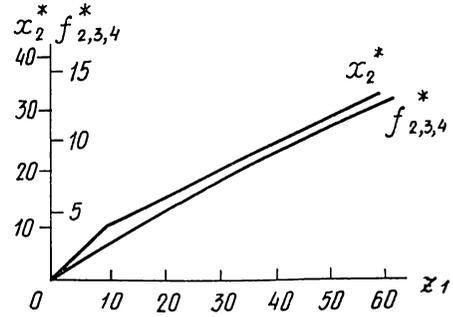


Рис. 4.13. Зависимость f_{234}^* и x_2^* от z_1

Величина f_{234}^* берется из рис. 4.13.

Строим зависимость f_{1234} от x_1 при $z_0=80$ (рис. 4.14). По точке максимума получаем, что $x_1^*=34$, а полное число пораженных орудий равно 14,1.

Итак, 34 самолета бомбят 1-й рубеж, 46 самолетов направлено дальше. К зоне действия орудий 2-го рубежа дошло z_1^* самолетов (прошли сквозь огонь оставшихся орудий 1-го рубежа из 46 самолетов):

$$z_1^* = 46 \cdot e^{-\alpha_1 \bar{N}_1^*};$$

$$\bar{N}_1^* = N_1 - Q_1(x_1^*);$$

$$Q_1(x_1^*) = N_1 \left(1 - e^{-\frac{\bar{v}_1^*}{N_1} P_1} \right);$$

$$\bar{v}_1^* = x_1^* e^{-\alpha_1 N_1}.$$
(4.3)

Получили $z_1^*=37$ и на 1-м рубеже поражено $Q_1(x_1^*)=5,6$ орудий. При $z_1^*=37$ из рис. 4.13 получили оптимальное управление на вто-

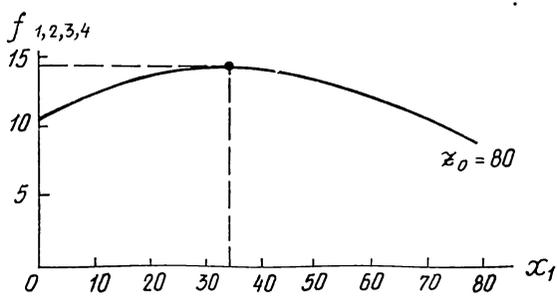


Рис. 4.14. Зависимость f_{1234} от x_1

ром шаге $x_2^* = 23$. То есть направить дальше следует 14 самолетов. Пользуясь формулами, аналогичными (4.3), получим

$$Q_2(x_1^*) = 5,4, \quad z_2^* = 10,7, \quad x_3^* = 0,$$

$$Q_3(x_3^*) = 0, \quad z_3^* = 5,9, \quad x_4^* = 5,9, \quad Q_4(x_4^*) = 3,1.$$

Итак, в первую волну войдут 34 самолета; $n_1 = 34$. Ко 2-му рубежу подойдут 37 самолетов, из них пойдут дальше только 14, а так как всего самолетов осталось 46, то во вторую волну следует включить $n_2 = 46(14/23) = 27$ самолетов, $n_3 = 0$ (так как $x_3^* = 0$) и в четвертую волну — оставшиеся 19 самолетов.

Анализ полученных результатов показывает, что оптимальнее осуществлять планирование на подступах к каждому рубежу, т. е. при этом решать задачу, аналогичную рассмотренной.

При более подробном рассмотрении убеждаемся, что эта задача является вырожденной задачей динамического программирования — планировать нужно каждый шаг отдельно, распределяя самолеты перед рубежом так, чтобы получать максимальное число самолетов, преодолевающих данных рубеж.

4.5. Вычислительные аспекты решения задач методом динамического программирования

Мы рассмотрели алгоритм решения задач, опирающийся на основное функциональное уравнение метода динамического программирования

$$f_m(S_0) = \max_{x_1} [f_1(S_0, x_1) + f_{m-1}(S_{m-1}(S_0, x_1))]; \quad m \geq 1 \quad (4.4)$$

Если $f_1(S_0, x_1)$ известно, то уравнение (4.4) представляет типичное рекуррентное соотношение. Чтобы вычислить $f_m(S_0)$, необходимо в запоминающем устройстве ЭВМ иметь значения $f_1(S_0, x_1)$, $f_{m-1}(S_{m-1})$; $S_{m-1}(S_0, x_1)$ или алгоритмы для образования этих значений.

Так как невозможно запомнить все значения $f_{m-1}(S_{m-1})$, то запоминается лишь дискретный набор значений $f_{m-1}[(S_{m-1})_i]$. Другие значения могут быть получены интерполяцией и экстраполяцией записанных данных. Чтобы найти максимум по x_1 , ограничимся дискретным набором значений $\{x_{1j}\}$; $j = 1, 2, \dots$. Одно или несколько значений x_{1j} , дающих максимум, опять запомним.

Повторяя эту процедуру на каждом шаге, получаем функцию дохода и функцию стратегии для данного шага. Рекуррентно вычислим последовательности $\{f_n(S_{n-1})\}$ и $\{x_n(S_{n-1})\}$. Видим, что процедура является итеративной, что облегчает ее программирование. Отметим, что как только $f_1(S_0)$ использована для вычисления $f_2(S_1)$ и $x_2(S_1)$, ее можно стереть из памяти и т. д. И чем меньше область возможных значений x_i , тем меньше требуется объем памяти, тем легче применить аппарат динамического программирования.

Для определенности рассмотрим случай, когда S_0 есть вектор размерности 2, $S_0 = f(x, y)$, и мы запоминаем значения S_0 в 100 точках по x и по y , т. е. имеем 10^4 чисел. Этот случай трудностей не вызывает. Если размерность вектора более высокая, скажем 4, то получим 10^8 чисел.

Для изучения таких процессов на современных ЭВМ применяют более тонкие алгоритмы. К примеру, вместо простого запоминания набора значений $\{f(S_0)_i\}$; $i=1, 2, \dots$ хранят в памяти метод воспроизведения $f(S_0)$. Наиболее простой прием такого воспроизведения функции — это представление ее в виде полинома.

Методы динамического программирования лучше «подходят» к задачам с дискретными переменными, чем с непрерывными. Основное функциональное уравнение имеет один и тот же вид и для дискретных, и для непрерывных переменных. В отдельных случаях при наличии непрерывных переменных удается сократить расчеты применением дифференциального исчисления; в большинстве же случаев задача сводится к задаче с дискретными переменными.

Использование процедуры динамического программирования позволяет найти все оптимальные решения, сколько бы их ни было, для любой *аддитивной* функции многих переменных, определенной на любом множестве D , заданном условиями-ограничениями.

Глава 5. О РАЗВИТИИ МЕТОДОВ РЕШЕНИЯ ЗАДАЧ МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

5.1. Основные направления развития методов решения задач математического программирования

Можно построить математические модели для достаточно широкого круга явлений. Однако мы выяснили, что не всякую математическую задачу мы в состоянии решить. Во-первых, по причине большой ее размерности, во-вторых из-за наличия нелинейных целевых функций и условий-ограничений. Кроме того, нам хотелось бы, чтобы, не решая заново задачу, иметь инструмент, который позволял бы с небольшими затратами изменить решения при небольших изменениях исходных данных (параметров целевой функции и условий-ограничений). Поскольку такие изменения исходных данных являются случайными величинами, то следовало бы определить и доверительные границы для решений реальной задачи. Но тогда мы выйдем за рамки этой книги — это область *стохастического программирования*.

Тем не менее удается, хотя достаточно и громоздкими методами, в задачах линейного программирования установить, как будет изменяться решение задачи, если известны функциональные зависимости, описывающие изменение исходных данных. Подобные задачи решают в разделе, называемом *«параметрическое программирование»*.

Несмотря на обилие разработанных методов решения сетевых задач трудности остаются. В гл. 2 мы упоминали, что из-за большой размерности некоторые сетевые задачи строгими методами решить не удастся; в подобных случаях применяют *эвристические методы*, которые дают хотя и не строго оптимальное решение, но приемлемое в практических приложениях.

В гл. 3 мы рассмотрели сетевые задачи для однородного потока. В реальных условиях по сети требуется транспортировать (передать) различные «продукты»: например, телефонные разговоры, телеграммы, телевизионные передачи и т. п., т. е. необходимо иметь алгоритмы, которые позволяли бы решать задачи о *многопродуктовых потоках* в сетях. Методы решения *многопродуктовых*

сетевых задач сложны и поэтому выбираемые алгоритмы позволяют решить задачу только приближенно.

Естественно стремление исследователей тем не менее создать в некотором роде универсальные алгоритмы, которые были бы применимы к достаточно широкому кругу задач математического программирования. К ним относятся алгоритмы, реализующие *методы штрафных (барьерных) функций: метод внутренней точки, метод внешней точки и комбинированный метод внутренней и внешней точки*. Методами штрафных функций можно решать задачи как линейного, так и нелинейного программирования. Хотя вряд ли экономно их использовать для решения линейных задач.

5.2. Понятие о параметрическом программировании

Исходные данные, необходимые для численной постановки реальных задач математического программирования, определяют в большинстве случаев неточно, приближенно. Это связано не только с наличием погрешностей измерения, но и с желанием описать в математической постановке возможное изменение исходных данных, чтобы в дальнейшем для оптимизации решения использовать наилучшие их значения.

Исследование чисто математических проблем, таких как анализ чувствительности решения при вариации исходных данных, оценка устойчивости решения, также требует разработки методов, учитывающих неопределенность исходных данных.

Как было показано, наиболее развиты к настоящему времени и наиболее доступны для изучения задачи линейного программирования. Алгоритм учета неопределенности исходных данных мы рассмотрим на примере задач линейного программирования, так как в других случаях учет неопределенности может оказаться крайне трудной проблемой. Раздел математического программирования, в котором может быть решена данная проблема, называют *параметрическим программированием*. Параметрическое программирование изучает в основном задачи, которые являются естественным обобщением задач линейного программирования, когда исходные данные (коэффициенты) в целевой функции и в условиях-ограничениях предполагаются не постоянными величинами, а функциями, зависящими определенным образом (чаще всего линейно) от некоторых параметров. Всякой задаче параметрического программирования можно поставить в соответствие некоторую задачу линейного программирования, называемую *исходной*. От того, как именно трактуется исходная задача, зависит трактовка параметрической задачи. Введение параметра обычно отражает некоторую реальную ситуацию.

Приведем несколько постановок задач параметрического программирования, которые наиболее естественным образом могут быть сопоставлены с принятой исходной задачей. Рассмотрим, помимо этого, одну из интерпретаций задачи параметрического программирования.

Пусть коэффициенты целевой функции исходной задачи линейного программирования (2.2) зависят от одного параметра. В задаче (2.2) коэффициенты целевой функции представляют собой цену единицы количества некоторого продукта, а координаты векторов-ограничений могут быть истолкованы как запасы различных ресурсов.

Целевая функция рассматриваемой задачи параметрического программирования может иметь (в простейшем случае) следующий вид:

$$f(x) = (a_{01} + b_1 t)x_1 + \dots + (a_{0n} + b_n t)x_n \rightarrow \max,$$

где a_{01}, \dots, a_{0n} — исходные (старые) коэффициенты задачи линейного программирования; b_1, \dots, b_n — новые коэффициенты; t — параметр, $t \in R^1$.

Зависимость коэффициентов этой функции от параметра t можно понимать как зависимость цены единицы продукта от времени. Различные новые коэффициенты b_1, \dots, b_n отражают индивидуальный характер зависимости от параметра t цен разных продуктов. Значение целевой функции исходной задачи равно стоимости выпущенной продукции, а значение целевой функции соответствующей параметрической задачи показывает, чему равна стоимость выпускаемой продукции при условии изменения цен единицы продукции, когда закон изменения этих цен (от времени, от качества продукции и т. п.) задан.

Рассмотрим случай, когда от параметра зависят координаты системы ограничений. Можно учесть зависимость этих показателей от времени, способа выработки ресурса, района размещения объекта, модель которого рассматривается в задаче и т. п. Условия ограничения примут вид

$$\begin{aligned} (a_{11} + c_{11}t)x_1 + \dots + (a_{1n} + c_{1n}t)x_n &\leq a_{10} + d_{1t}; \\ (a_{m1} + c_{m-1}t)x_1 + \dots + (a_{mn} + c_{mn}t)x_n &\leq a_{m0} + d_{mt}; \\ x_1, \dots, x_n &\geq 0; t \in R^1. \end{aligned}$$

Буквами $a_{..}$ с соответствующими индексами обозначены исходные коэффициенты в условиях-ограничениях задачи, а буквами $c_{..}$ и $d_{..}$ — новые коэффициенты, определяющие зависимость исходных коэффициентов от параметра t .

Задача параметрического программирования с s независимыми переменными t_1, \dots, t_s , или s -параметрическая задача, записывается следующим образом:

максимизировать

$$f(x) = (a_{01} + b_{11}t_1 + \dots + b_{1s}t_s)x_1 + \dots + (a_{0n} + b_{n1}t_1 + \dots + b_{ns}t_s)x_n + e_1t_1 + \dots + e_st_s$$

при ограничениях

$$\left\{ \begin{array}{l} (a_{11} + c_{111}t_1 + \dots + c_{11s}t_s)x_1 + \dots + (a_{1n} + c_{1n1}t_1 + \dots + c_{1ns}t_s)x_n \leq \\ \leq a_{10} + d_{11}t_1 + \dots + d_{1s}t_s; \\ (a_{m1} + c_{m11}t_1 + \dots + c_{m1s}t_s)x_1 + \dots + (a_{mn} + c_{mn1}t_1 + \dots + \\ + c_{mns}t_s)x_n \leq a_{m0} + d_{m1}t_1 + \dots + d_{ms}t_s; \\ x_1, \dots, x_n \geq 0; t \in R^1. \end{array} \right.$$

Рассмотрим метод решения частной задачи линейного параметрического программирования, в которой все коэффициенты целевой функции линейно зависят от некоторого действительного параметра t на конкретном примере:

$$f(x) = (2 + t)x_1 + (3 - t)x_2 + t \rightarrow \max;$$

$$\left\{ \begin{array}{l} -x_1 + 2x_2 \leq 4 \\ x_1 + x_2 \leq 5 \\ 2x_1 - x_2 \leq 8 \\ x_1 \geq 0; x_2 \geq 0; t \in R^1, \end{array} \right. \quad (5.1)$$

В матрично-векторной форме

$$\max f(x) = (a_0 + bt)^T x;$$

$$Ax \leq a_0;$$

$$x \geq 0; t \in R^1,$$

$$\text{где } a_0 = \{2; 3; 0\}^T; b = \{1; -1; 1\}^T; x = \{x_1; x_2; 1\}^T;$$

$$a_0 = \{4, 5, 8\}^T;$$

$$A = \begin{pmatrix} -1 & 2 \\ 1 & 1 \\ 2 & -1 \end{pmatrix}$$

Для каждого фиксированного значения t задача (5.1) становится задачей линейного программирования, которую называют *принадлежащей этому значению t* .

Решением параметрической задачи (5.1) называют явным образом заданную функцию

$$\hat{f}(t) = \max\{f(x) = (a_0 + bt)^T x \mid AX \leq a_0; X \geq 0\},$$

решающую функцию задачи (5.1), вместе с набором *решающих отношений* $\hat{x}_1(t), \dots, \hat{x}_n(t)$, каждое из значений которых при данном значении t равно значениям переменных x_1, x_2, \dots, x_n , образующих *оптимальное решение задачи*, принадлежащей данному значению t

(если это решение существует). Доказано, что r -я критическая область K^r задачи (5.1), $r=1, \dots, \infty$, определяемая условием

$$K^r = \{t \mid a_{0j}^r(t) = a_{0j}^r + b_j^r t \geq 0, j = \overline{1, n+m}\},$$

является отрезком (замкнутым интервалом). Область определения Q решающей функции $\hat{f}(t)$ выпукла; решающая функция $\hat{f}(t)$ выпукла в своей области определения.

При любом $r=1, 2, \dots$ для всех значений t из критической области K^r решающая функция линейна и в силу этого непрерывна в области Q .

При любом $r=1, 2, \dots$ внутри критической области K^r множества значений отношений $\hat{x}_j(t)$, $j = \overline{1, n}$ ограничены постоянными величинами; сами эти отношения полунепрерывны сверху.

Для построения решающей функции будем пользоваться симплекс-методом в следующей его модификации. Под последней строкой первой из симплекс-таблиц, полученной при $t=t_0=0$ и содержащей некоторое допустимое базисное решение, присписывают еще одну строку коэффициентов $b_j^{(1)}t = -b_j t$, $j = \overline{1, n+m}$, где $b_j=0$ для $j = n+1; n+m$. Полученная строка подвергается тем же преобразованиям, что и остальные (табл. 5.1; $r=1$). Естественно, что в задаче (5.1) мы предвзительно перешли к ограничениям — равенствам и ввели новые переменные x_3, x_4, x_5 . За три итерации ($r = \overline{1, 3}$) мы получили оптимальное решение при $t=0$ (в табл. 5.1 для каждой итерации выделены разрешающие элементы). Коэффициенты в целевой функции для выбора разрешающего элемента задают двумя последними строками при фиксированном (заданном) значении $t=t_0$. Для данного допустимого базисного решения надо построить соответствующую критическую область — множество всех значений t , при которых это решение будет оптимальным. Это множество будет замкнутым интервалом. Надо найти границы этого интервала. Их находят по формулам

$$m_{-1}^r = \max \left\{ -\frac{a_{0j}^r}{b_j^r}, -\infty \mid b_j^r > 0, j = \overline{1, n+m} \right\};$$

$$m_{+1}^r = \min \left\{ \frac{a_{0j}^r}{b_j^r}, +\infty \mid b_j^r < 0, j = \overline{1, n+m} \right\}.$$

Если нижняя (верхняя) граница этого интервала есть $-\infty$ ($+\infty$), то найдена нижняя (верхняя) граница области Q . Для любой границы этого интервала, такой, что координата ее есть число, по крайней мере один из коэффициентов целевой функции $a_{0j}^r(t) = a_{0j}^r + b_j^r t$ равен нулю. При значениях $t=t_p$, обращающих выражение $a_{0j}^r + b_j^r t$ в нуль и называемых *критическими*, решения задачи определены неоднозначно. Если существует второе базисное решение, то изменяя базис, можно задавать соседнюю крити-

Симплекс-таблицы для линейной задачи параметрического программирования

Номер итерации	$-x_1$	$-x_2$	$-x_3$	$-x_4$	$-x_5$	a_0	
$r = 1$	-1	$\overline{2}$	1	0	0	4	x_3
	1	1	0	1	0	5	x_4
	2	-1	0	0	1	8	x_5
	-2	-3	0	0	0	0	$-a_0$
	$-t$	t	0	0	0	t	$-b_j t$
$r = 2$	1/2	1	1/2	0	0	2	x_2
	$\overline{3/2}$	0	-1/2	1	0	3	x_4
	3/2	0	1/2	0	1	10	x_5
	-7/2	0	3/2	0	0	6	$-a_0$
	$-t/2$	0	$-t/2$	0	0	$-t$	$-b_j t$
$r = 3$	0	1	1/3	1/3	0	3	x_2
	1	0	-1/3	$\overline{2/3}$	0	2	x_1
	0	0	$\overline{1}$	-1	1	7	x_5
	0	0	1/3	7/3	0	13	
	0	0	$-2t/3$	$t/3$	0	0	
$\pi = -1$							
$r = 4$	1/2	1	1/2	0	0	2	x_2
	3/2	0	-1/2	1	0	3	x_4
	3/2	0	1/2	0	1	10	x_5
	-7/2	0	3/2	0	0	6	
	$-t/2$	0	$-t/2$	0	0	$-t$	
$\pi = +1$							
$r = 4$	0	1	0	$\overline{2/3}$	-1/3	2/3	x_2
	1	0	0	1/3	1/3	13/3	x_1
	0	0	1	-1	1	7	x_3
	0	0	0	8/3	-1/3	32/3	
	0	0	0	$-t/3$	$2t/3$	$14t/3$	
$r = 5$	0	3/2	0	1	-1/2	1	x_4
	1	-1/2	0	0	1/2	4	x_2
	0	1	1	0	1/2	8	x_3
	0	-8/2	0	0	1	8	
	0	$t/2$	0	0	1/2	$5t$	

ческую область значений t . Если же при попытке заменить базис обнаружена неограниченность решения, то гиперплоскость, соответствующая целевой функции при рассматриваемом ее значении, оказывается параллельной одной из граней выпуклого многогранника, определяемого системой ограничений задачи. Такая грань содержит точки, хотя бы одна из координат которых сколь угодно велика по абсолютному значению. В этом случае достигнута конечная нижняя (верхняя) грань области Q . Пусть при построении обнаружилось, что некоторая точка полученного интервала имеет координату, меньшую (большую), чем координата верхней (нижней) границы интервала, полученного на предыдущем шаге построения. Говорят в таком случае, что эта *верхняя (нижняя) граница превзойдена*. Тогда сама эта верхняя (нижняя) граница является и нижней (верхней) границей строящегося интервала. Определим критическое значение $t_p = m_{\pi}^{(r)}$. Индекс $\pi = -1$ применяют при отыскании нижней границы области Q , индекс $\pi = +1$ — при отыскании верхней границы. В нашем случае $r=3$; отыскиваем нижнюю границу области Q , т. е. полагаем $\pi = -1$. Получим, положив $p = \pi = -1$ (для $b_j > 0; j = \overline{1, n+m}$),

$$t_{-1} = m_{-1}^3 = \max \left\{ -\frac{7/3}{1/3}; -\infty \right\} = -7.$$

Из табл. 5.1 при $r=3$ следует, что при $t=7$ $\hat{f}(t) = 13$, $\hat{x}_1 = 2(1-\lambda)$, $\hat{x}_2 = 2\lambda + 3(1-\lambda)$, $0 \leq \lambda \leq 1$. Это критическое значение может быть превзойдено. В табл. 5.1 при $r=3$ выбран разрешающий элемент, равный $2/3$ при $j=4$. Общая формула для выбора этой небазисной переменной имеет вид

$$\tau_p \in \{ \tau | a_{0\tau}^r + b_{\tau p}^r t_p = 0; x_{\tau} \text{ — небазисная переменная} \}.$$

Для нас $\tau=4$. Получим следующую таблицу при $\pi = -1$ и $r=4$. Она совпала с таблицей, полученной при $r=2$. Так как все b_n^4 отрицательны, то $t_{-2} = -\infty$ и является нижней границей области Q . Из табл. 5.1 при $r=4$ получаем, что при $t < -7$ $\hat{f}(t) = 6-t$, $\hat{x}_1 = 0$, $\hat{x}_2 = 2$.

Отыскиваем верхнюю границу области Q , полагаем $\pi = +1$, $p = \pi$, $p = +1$, $r=3$. Тогда для $b_j < 0, j = \overline{1, n+m}$,

$$t_{+2} = m_1^3 = \min \left\{ -\frac{1/3}{-2/3}; +\infty \right\} = +1/2.$$

При $t = 1/2$ $\hat{f}(t) = 13$, $\hat{x}_1 = 2\lambda + 13(1-\lambda)/3$, $\hat{x}_2 = 3\lambda + 2(1-\lambda)/3$, $0 \leq \lambda \leq 1$. Выбираем разрешающий элемент из столбца для x_3 и получаем новую таблицу при $\pi = +1$ и $r=4$. Из этой таблицы найдем

новое t_{+2} , превышающее критическое значение $t_{+2}=1/2$. Новое значение t_{+2} при $-7 < t < 1/2$ $\hat{f}(t)=13$, $\hat{x}_1=2$, $\hat{x}_2=3$

$$t_{+2} = m_{+1}^4 = \min \left\{ -\frac{8/3}{-1/3}; +\infty \right\} = 8.$$

При $t = 8$ $\hat{f}(t) = 1/3(13 + 14 \cdot 8) = 48$,

$$\hat{x}_1 = \frac{13}{3}\lambda + 4(1 - \lambda); \hat{x}_2 = \frac{2\lambda}{3};$$

при $\frac{1}{2} < t < 8$ $\hat{f}(t) = (32 + 14t)/3$, $\hat{x}_1 = 13/3$, $\hat{x}_2 = 2/3$.

Строим следующую таблицу при $r=5$ и $\pi=+1$. Получаем $t > 8$.

Окончательные результаты приведены в табл. 5.2 и на рис. 5.1, где показано «вращение» целевой функции, поведение решающей функции $\hat{f}(t)$ и геометрические образы отношений $\hat{x}_1(t)$ и $\hat{x}_2(t)$.

Таблица 5.2

Решение параметрической задачи ($0 \leq \lambda \leq 1$)

t	$\hat{f}(t)$	\hat{x}_1	\hat{x}_2
$t < -7$	$6 - t$	0	2
$t = -7$	13	$2(1 - \lambda)$	$2\lambda + 3(1 - \lambda)$
$-7 < t < 1/2$	13	2	3
$t = 1/2$	13	$2\lambda + 13(1 - \lambda)/3$	$3\lambda + 2(1 - \lambda)/3$
$1/2 < t < 8$	$(32 + 14t)/3$	13/3	2/3
$t = 8$	48	$13\lambda/3 + 4(1 - \lambda)$	$2\lambda/3$
$t > 8$	$8 + 5t$	4	0

Алгоритм решения линейной параметрической задачи в случае зависимости от параметра коэффициентов целевой функции в предположении, что допустимая область, характеризующаяся ограничениями задачи, не пуста и что для некоторого значения параметра $t=t_0$ существует оптимальное решение, имеет следующий вид:

1. При $t=t_0$ r -я симплекс-таблица является оптимальной. Полагаем $\pi=-1$ и $p=\pi$, переходим к п. 2.

2. Определяем критическое значение $t_p = m_{\pi}^r$.

а. Если $|t_p| = \infty$, то достигнута граница области Q .

а1. Если $\pi=-1$, то отыскиваем нижнюю границу области Q . Полагаем $\pi=+1$; $p=\pi$, заменяем r на $r+p+1$ и переходим снова к п. 2.

а2. Если $\pi=+1$, то отыскиваем верхнюю границу области Q . Переходим к п. 3.

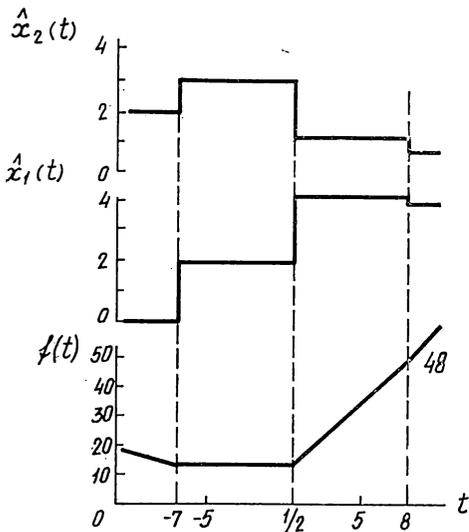
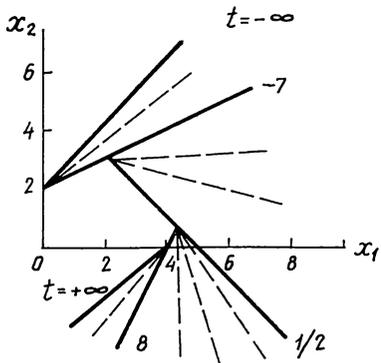


Рис. 5.1. Решение задачи параметрического линейного программирования

б Если $|t_p| < \infty$, то определяем возможность превзойти критическое значение t_p . С помощью выражения для τ_p^r , иногда подбором из нескольких возможных значений, определяем новую базисную переменную. В этом случае следует принимать во внимание наличие множества различных решений.

б1. Если $a_i^r \tau \leq 0$, $i = \overline{1, m}$, то достигнута граница области Q . Переходим к п. 2а1 или к п. 2а2.

б2. В противном случае строим новую симплекс-таблицу. Заменяя r на $r+1$, p на $p+\pi$, снова начинаем вычисления с п. 2.

3. Вычисления прекращаем. Для каждой критической области значение решающей функции $\hat{f}(t)$ берем из соответствующей таблицы. Там же определены оптимальные значения решающих соотношений $\hat{x}_1(t), \dots, \hat{x}_n(t)$.

В практических приложениях часто возникают следующие частные случаи параметрической задачи:

1) значение параметра t принадлежит некоторому заранее выбранному интервалу $L \subset R^1$;

2) требуется отыскать только минимум (максимум) функции $\hat{f}(t)$ для всех $t \in R^1$, т. е. требуется определить те значения t , при которых возможно достижение этого минимума (максимума);

3) требуется отыскать интервал возможных изменений какого-то одного из коэффициентов целевой функции.

Особый интерес представляет задача нахождения наибольшего интервала $t_1 \leq t \leq t_2$, такого, что оптимальное базисное решение любой из задач, принадлежащих значениям t из этого интервала,

совпадает с оптимальным базисным решением задачи, принадлежащей значению $t=0$.

Если область определения Q решающей функции $\hat{f}(t)$ ограничена заранее интервалом L , то вычисления, предписываемые п. 2 алгоритма, заканчивают, когда будут достигнуты границы интервала L . Если интервал $t_{-1} \leq t \leq t_{+1}$ не имеет общих точек с интервалом L , то полагают $\pi = -1$ или $\pi = +1$ в зависимости от того, будут ли значения $t \in L$ меньше t_{-1} или больше t_{+1} .

Если в задаче необходимо установить только минимум (максимум) значения решающей функции $\hat{f}(t)$, то вычисления, предписываемые п. 2, проводят при $\pi = -1$ или $\pi = +1$ в зависимости от того, в каком направлении уменьшается значение функции $\hat{f}(t)$.

Если требуется только определить возможность вариации одного из коэффициентов, то сперва устанавливают, существует ли решение при $t=0$ и находят это решение. Вводя затем параметр t , сводят проблему к определению значений t_{-1} (равных t_1) и t_{+1} (равных t_2).

5.3. Многопродуктовые потоки в сетях

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 расположены в городах Санта-Барбара, Бейкерсфилд и Сакраменто. В период уборки урожая из этих садов поставляют в различных количествах апельсины, лимоны и лаймы (табл. 5.3).

Т а б л и ц а 5.3

Производительность (ящики в неделю)

Город	Апельсины	Лимоны	Лаймы
Санта-Барбара	800	700	700
Бейкерсфилд	1000	800	500
Сакраменто	500	1000	500

По договорному соглашению требуется доставлять в оптовые магазины, расположенные в Денвере, Сиэтле и Канзас-Сити, такое количество фруктов, которое указано в табл. 5.4.

Перевозка осуществляется по железной дороге, однако в каждом поезде для фруктов отведено ограниченное место. Пропускные способности путей между различными пунктами отправления и

Таблица 5.4

Величина спроса (ящики в неделю)

Город	Апельсины	Лимоны	Лаймы
Денвер	900	900	700
Сиэтл	700	1000	500
Канзас-Сити	700	600	500

пунктами назначения приведены в табл. 5.5. 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

Таблица 5.5

Пропускная способность железных дорог (ящики в неделю)

Из	В		
	Денвер	Сиэтл	Канзас-Сити
Санта-Барбара	1200	900	1000
Бейкерсфилд	1200	1000	1100
Сакраменто	950	1300	1000

соответствующие транспортные затраты зависят от вида упакованных в него фруктов. Затраты на транспортировку одного ящика приведены в табл. 5.6.

Задача определения схемы перевозки фруктов, минимизирующей транспортные затраты, является *задачей о многопродуктовом потоке*. Характерная особенность задач о многопродуктовом потоке состоит в том, что по дугам сети протекает несколько неоднородных потоков. При этом суммарный поток всех продуктов по дуге ограничен ее пропускной способностью. Если бы это было не так, то можно было бы для каждого продукта независимо решить минимизационную транспортную задачу. Однако указанные выше ограничения на пропускные способности дуг существенно усложняют решение задачи.

Задачи о многопродуктовом потоке, как и об однопродуктовом потоке, могут быть сформулированы в виде задач линейного программирования. Задача о транспортировке фруктов является примером многопродуктовой транспортной задачи. Обозначим через x_{ij}^k поток k -го продукта из i -го источника в j -й сток, а через c_{ij}^k — стоимость транспортировки единицы этого продукта. Пусть далее a_i^k и b_j^k — это соответственно предложение узла- i и спрос

Транспортные затраты (центы на ящик)

Из	В		
	Денвер	Сиятл	Канзас-Сити
<i>Санта-Барбара</i>			
Апельсины	3,2	2,5	5,6
Лимоны	2,4	1,9	4,3
Лаймы	2,3	1,7	4,0
<i>Бейкерсфилд</i>			
Апельсины	3,0	2,1	5,5
Лимоны	2,3	1,7	4,4
Лаймы	2,0	1,4	4,3
<i>Сакраменто</i>			
Апельсины	3,1	2,0	5,7
Лимоны	2,2	1,6	4,8
Лаймы	2,0	1,5	4,3

узла j для k -го продукта, а u_{ij} — пропускная способность дуги (i, j) . Математическая постановка многопродуктовой транспортной задачи выглядит следующим образом:

минимизировать

$$\sum_{k=1}^r \sum_{i=1}^m \sum_{j=1}^n c_{ij}^k x_{ij}^k \quad (5.2)$$

при условии, что

$$\sum_i x_{ij}^k = b_j^k \quad \text{для всех } j, k;$$

$$\sum_j x_{ij}^k = a_i^k \quad \text{для всех } i, k;$$

$$\sum_k x_{ij}^k \leq u_{ij} \quad \text{для всех } i, j;$$

$$x_{ij}^k \geq 0 \quad \text{для всех } i, j, k.$$

Как и в однопродуктовой транспортной задаче, предполагается, что для каждого продукта суммарное предложение равно суммарному спросу, т. е.

$$\sum_i a_i^k = \sum_j b_j^k \quad \text{для всех } k.$$

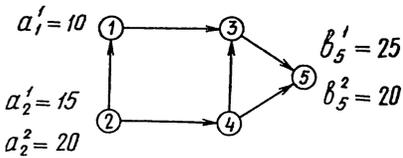


Рис. 5.2. Сеть в многопродуктовой задаче о перевозках

Во многих задачах, называемых *многопродуктовыми задачами о перевозках*, вводят промежуточные узлы, т. е. такие узлы, которым не приписывают ни спрос, ни предложение, а только требуется, чтобы для них выполнялось условие сохранения потока.

В качестве одного из таких примеров рассмотрим сеть, изображенную на рис. 5.2. Узлы 1 и 2 являются источниками 1-го продукта, а узел 2 — источником 2-го продукта. 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:

минимизировать
$$\sum_{k=1}^r \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k$$

при условии, что

$$\sum_j x_{ij}^k - \sum_j x_{ji}^k = a_i^k, \text{ если узел } i \text{ является источником } k\text{-го продукта,}$$

$$\sum_j x_{ij}^k - \sum_j x_{ji}^k = 0, \text{ если узел } i \text{ является промежуточным узлом,}$$

$$\sum_i x_{ij}^k - \sum_i x_{ji}^k = -b_j^k, \text{ если узел } j \text{ является стоком } k\text{-го продукта,}$$

$$\sum_k x_{ij}^k \leq u_{ij} \text{ для всех } (i, j) \in A,$$

$$x_{ij}^k \geq 0 \text{ для всех } k \text{ и } (i, j) \in A.$$

Через A здесь обозначается множество дуг сети.

Одна из трудностей решения задачи о многопродуктовом потоке вызвана тем, что оптимальные решения могут быть нецелочисленными. В качестве примера рассмотрим двухпродуктовую транспортную задачу, сетевая формулировка которой задана на рис. 5.3. Дугам сети приписаны числа c_i^1 , c_{ij}^2 , u_{ij} . Оптимальное решение соответствующей задачи линейного программирования представлено в табл. 5.7.

Нецелочисленные решения возникают по той причине, что матрицы ограничений в задачах линейного программирования, вообще говоря, не *унимодулярны*. Напомним, что условие унимодуляр-

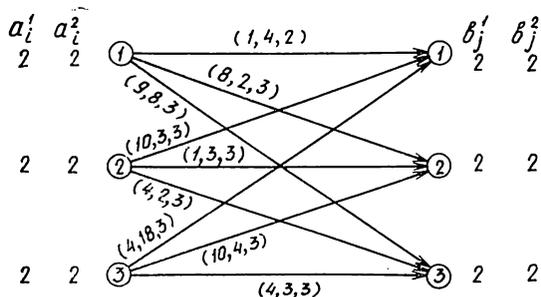


Рис. 5.3. Двухпродуктовая транспортная задача

ности матрицы ограничений 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, намного быстрее, чем операции, использующие числа (десятичные) с плавающей точкой. Благодаря этому создают программы, позволяющие очень быстро решать задачи большой размерности.

Для задач о многопродуктовом потоке были разработаны специальные алгоритмы, в которых учтены специфика структуры и свойства данного класса задач. Было показано, что эти алгоритмы являются более быстрыми, чем процедуры решения общей задачи линейного программирования, но значительно более медленными, чем соответствующие алгоритмы 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.

Таблица 5.7

Оптимальное решение задачи линейного программирования

Дата	Продукт 1	Продукт 2
(1, 1)	3/2	1/2
(1, 2)	0	3/2
(1, 3)	1/2	0
(2, 1)	0	3/2
(2, 2)	2	0
(2, 3)	0	1/2
(3, 1)	1/2	0
(3, 2)	0	1/2
(3, 3)	3/2	3/2

5.4. 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 любого алгоритма решения задачи о потоке минимальной стоимости.

Для многопродуктовой транспортной задачи 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, если $m \leq 2$ или $n \leq 2$ независимо от числа продуктов. Очевидно, что если m или n равно 1, то решение тривиально. Однако, когда и m , и n превосходят 1, то возникает более общая задача линейного программирования. Покажем, что при этом существует более простой метод решения.

Введя слабые переменные s_{ij} в ограничения на пропускные способности дуг, переформулируем многопродуктовую транспортную задачу (5.2) следующим образом:

$$\text{минимизировать } \sum_{k=1}^r \sum_{i=1}^m \sum_{j=1}^n c_{ij}^k x_{ij}^k$$

при условии, что

$$\sum_i x_{ij}^k = b_j^k \quad \text{для всех } j, k;$$

$$\sum_j x_{ij}^k = a_i^k \quad \text{для всех } i, k;$$

$$\sum_k x_{ij}^k + s_{ij} = u_{ij} \quad \text{для всех } i, j;$$

$$x_{ij}^k, s_{ij} \geq 0 \quad \text{для всех } i, j, k.$$

При $m=2$ данная задача сводится к следующей задаче линейного программирования:

минимизировать

$$\sum_{k=1}^r \sum_{j=1}^n [(c_{2j}^k - c_{1j}^k) x_{2j}^k + c_{1j}^k b_j^k] \quad (5.3)$$

при условии, что

$$\sum_k x_{2j}^k + s_{2j} = u_{2j} \text{ для всех } j; \quad (5.4)$$

$$-\sum_j x_{2j}^k = -a_2^k \text{ для всех } k; \quad (5.5)$$

$$-\sum_j s_{2j} = -\sum_j u_{2j} + \sum_k a_2^k, \quad (5.6)$$

$$x_{2j}^k + x_{1j}^k = b_j^k \text{ для всех } j, k; \quad (5.7)$$

$$s_{2j} + s_{1j} = u_{2j} + u_{1j} - \sum_k b_j^k \text{ для всех } j; \quad (5.8)$$

$$x_{ij}^k s_{ij} \geq 0 \text{ для всех } i, j, k. \quad (5.9)$$

Нетрудно заметить, что в равенствах (5.4)—(5.6) каждая из переменных x_{2j}^k и s_{2j} появляется дважды: один раз со знаком плюс и один раз со знаком минус. Отметим также, что переменные x_{1j}^k и s_{1j} входят только в равенства (5.7) и (5.8) (со знаком плюс). Поэтому можно рассматривать эти переменные как слабые и исключить их, записав (5.7) и (5.8) в виде

$$x_{2j} \leq b_j^k \text{ для всех } j, k;$$

$$s_{2j} \leq u_{2j} + u_{1j} - \sum_k b_j^k \text{ для всех } j.$$

Величины, стоящие в правых частях равенств (5.4)—(5.6), представляют собой предложение (если они положительные) и спрос (если отрицательные). Равенства (5.4)—(5.6) описывают транспортную модель, сетевая структура которой показана на рис. 5.4. Правые части равенств (5.7) и (5.8) представляют собой пропускные способности дуг, соответствующих переменным x_{2j}^k и s_{2j} . Для решения данной задачи определяют оптимальные потоки x_{2j}^k и s_{2j} . Затем из (5.7) и (5.8) могут быть найдены величины x_{1j}^k и s_{1j} .

Интересно отметить, что исходная задача линейного программирования не имела форму сетевой задачи. Однако с помощью преобразования ограничений удалось свести ее к сетевой и тем самым существенно упростить решение.

5.5. 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

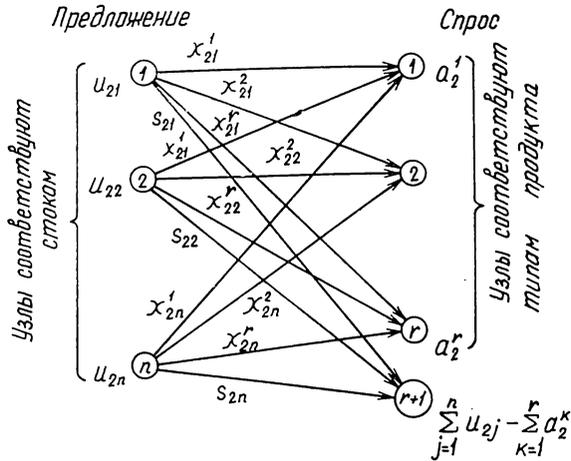


Рис. 5.4. Сеть в многопродуктовой транспортной задаче, сведенной к задаче об однопродуктовом потоке

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 множества 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.

Будем рассматривать многопродуктовую транспортную задачу с m источниками n стоками и r продуктами. Сведем ее к сетевой задаче с двумя источниками, разбив множество источников на два подмножества S_1 и S_2 . Теперь необходимо определить новые предложения агрегированных узлов, решить вопрос о стоимостях и пропускных способностях агрегированных дуг и найти способ построения допустимого решения исходной задачи с помощью решения агрегированной задачи.

Поскольку каждый из двух агрегированных узлов представляет собой совокупность источников в исходной задаче, то величины предложения этих двух узлов \bar{a}_i^k определяют как

$$\bar{a}_i^k = \sum_{i \in S_l} a_i^k, \quad l = 1, 2,$$

где a_i^k — предложение k -го продукта из i -го узла исходной задачи.

Пусть y_{ij}^k — поток k -го продукта из узла l в узел j в агрегированной сети. Поскольку дуга (l, j) получена в результате агрегирования дуг, ведущих из узлов $i \in S_l$ в сток j , то

$$y_{ij}^k = \sum_{i \in S_l} x_{ij}^k,$$

где x_{ij}^k — поток k -го продукта из узла i в узел j в исходной задаче.

Для определения стоимостей воспользуемся понятием *взвешенного агрегирования*. Стоимости взвешивают пропорционально величинам предложения источников, представленных агрегированными узлами:

$$\bar{c}_{ij}^k = \sum_{i \in S_l} c_{ij}^k (a_i^k / \bar{a}_i^k).$$

Чтобы определить пропускные способности агрегированных дуг \bar{u}_{lj} , найдем величины

$$\delta_i = \min_k [\bar{a}_i^k / a_i^k] \text{ для } i \in S_l.$$

Тогда $\bar{u}_{lj} = \min_{i \in S_l} [\delta_i u_{ij}^k]$.

Если принять \bar{u}_{lj} как сумму пропускных способностей исходных дуг u_{ij} из источников $i \in S_l$ в сток j , то могут возникнуть трудности при построении допустимого решения исходной задачи.

Сформулируем агрегированную задачу в виде следующей задачи линейного программирования:

$$\text{минимизировать } \sum_{k=1}^r \sum_{l=1}^2 \sum_{j=1}^n \bar{c}_{lj}^k y_{lj}^k$$

при условии, что

$$\sum_l y_{lj}^k = b_j^k \text{ для всех } j, k;$$

$$\sum_j y_{lj}^k = \bar{a}_l^k \text{ для всех } l, k;$$

$$\sum_k y_{lj}^k \leq \bar{u}_{lj} \text{ для всех } l, j;$$

$$y_{lj}^k \geq 0 \text{ для всех } l, j, k.$$

Теперь необходимо построить решение исходной задачи, используя оптимальное решение агрегированной задачи. Определим величины

$$x_{ij}^k = y_{ij}^k (a_i^k / \bar{a}_i^k) \text{ для } i \in S_l.$$

Откуда следует, что

$$\sum_{i \in S_l} x_{ij}^k = y_{ij}^k \left(\sum_{i \in S_l} a_i^k | \bar{a}_i^k \right) = y_{ij}^k (\bar{a}_i^k | \bar{a}_i^k) = y_{ij}^k .$$

Эту процедуру называют *дезагрегированием с фиксированными весами*. Кроме того,

$$\sum_k \sum_t \sum_j c_{ij}^k x_{ij}^k = \sum_k \sum_t \sum_j \bar{c}_{ij}^k y_{ij}^k .$$

Иными словами, значение целевой функции в результате дезагрегирования с фиксированными весами не изменяется. Однако сле-

Т а б л и ц а 5.8

Параметры агрегированной задачи

Предложения

Город	Апельсины	Лимоны	Лаймы
Санта-Барбара	800	700	700
{ Бейкерсфилд	1500	1800	1000
{ Сакраменто			

Стоимости

Из	В		
	Денвер	Сиэтл	Канзас-Сити
<i>Санта-Барбара</i>			
Апельсины	3,200	2,500	5,600
Лимоны	2,400	1,900	4,300
Лаймы	2,300	1,700	4,000
{ <i>Бейкерсфилд</i>	3,033	2,067	5,567
{ <i>Сакраменто</i>			
Апельсины			
Лимоны	2,244	1,644	4,622
Лаймы	2,000	1,450	4,300

Пропускные способности

Из	В		
	Денвер	Сиэтл	Канзас-Сити
Сата-Барбара	1200	900	1000
{ Бейкерсфилд	1500	1000	1000
{ Сакраменто			

дует помнить, что в силу определения величин \bar{u}_{ij} в агрегированной задаче может не существовать допустимого решения, даже если в исходной задаче оно существует. В этом случае можно попытаться воспользоваться другими методами агрегирования.

Продемонстрируем процедуру агрегирования на конкретном примере. Перейдем к решению задачи о транспортировке фруктов, сформулированной в § 5.3. Агрегируем источники 2 и 3. Величины предложения, стоимости и пропускные способности для агрегированной задачи содержатся в табл. 5.8.

Стоимость в оптимальном решении исходной задачи равна 18 570 долл. Стоимость в оптимальном решении агрегированной задачи равна 18 799,10 долл. и на 0,26% превосходит стоимость действительного оптимального решения.

5.6. 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.

Составление расписания движения судов. Одной из наиболее часто возникающих задач в области планирования перевозок является задача составления оптимального расписания движения транспортных средств. Предположим, что известно число разнотипных судов, которые отличаются друг от друга скоростью передвижения, 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 оптимальных маршрута и 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. Судя k -го типа первоначально располагают в источнике s^k , и дуги (s^k, i) , таким образом, соответствуют началу их эксплуатации. Далее, вводят 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:

максимизировать

$$f(x) = \sum_k \sum_i \sum_j c_{ij}^k x_{ij}^k \quad (5.10)$$

при условии, что

$$\sum_k \sum_{A_v} r_{ij}^k x_{ij}^k \leq b_v, \quad (5.11)$$

$$\sum_i x_{ij}^k - \sum_j x_{ji}^k = \begin{cases} d^k, & i = s^k; \\ 0, & i \neq s^k, t; \\ -d^k, & i = t. \end{cases} \quad (5.12)$$

$$0 \leq x_{ij}^k \leq u_{ij}^k. \quad (5.13)$$

Здесь через c_{ij}^k обозначена полезность, соответствующая отдельному судну, перевозимому грузу и маршруту; A_v — это множество допустимых маршрутов для данного груза; r_{ij}^k — грузоподъемность судна k -го типа на маршруте (i, j) . Суммарный перевозимый груз не может превышать спроса на него b_v . Ограничения (5.12) описывают условие сохранения потока по числу судов d^k , а величина x_{ij}^k равна 0 или ∞ в зависимости от того, может ли быть использовано на дуге (i, j) судно k -го типа или нет. В качестве примера рассмотрим задачу, описанную в табл. 5.9. Число судов каждого типа равно 2. Соответствующая сеть изображена на рис. 5.5. Отметим, что в рассмотренном классе задач узлы служат для обозначения места и времени.

Таблица 5.9

Груз	Минимально допустимый момент начала перевозки (через n дней)	Допустимые сроки доставки
1	0	4, 5
2	2	6, 7, 8
3	3	7, 8

Проектирование городской транспортной сети. Многопродуктовые сетевые модели используют и 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 способ-

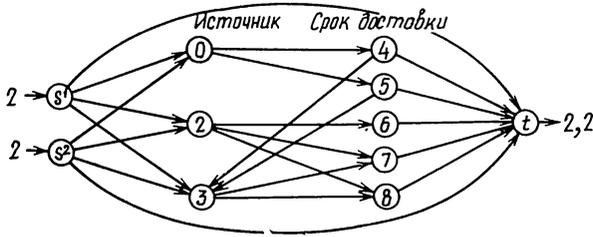


Рис. 5.5. Сеть в задаче составления расписания движения судов

ность 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, предназначенных на улучшение дорог, равна B . «Продуктами» в данной модели являются потоки из каждого источника во все пункты назначения. Пусть x_{ij}^k — 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:

$$f_{ij} \left(\sum_k x_{ij}^k \right).$$

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:

минимизировать
$$\sum_i \sum_j f_{ij} \left(\sum_k x_{ij}^k \right)$$

при условии, что

$$\sum_j x_{ji}^k - \sum_i x_{ij}^k = d_{ki},$$

$$\sum_k x_{ij}^k \leq u_{ij} + y_{ij}, \tag{5.14}$$

$$\sum_i \sum_j c_{ij} y_{ij} \leq B,$$

$$x_{ij}^k, y_{ij} \geq 0.$$

В задаче (5.14) было сделано одно важное допущение, касающееся поведения водителя. Оно основано на классическом принципе

распределения транспортных средств: водители выбирают маршруты таким образом, что суммарное время поездок для всей системы является минимальным. Как правило, целевая функция задачи (5.14) является нелинейной и выпуклой.

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 максимального 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 вводят матрицу, элемент 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:

$$\text{минимизировать } \sum_k \sum_i \sum_j e_{ij} x_{ij}^k + \sum_i \sum_j c_{ij} y_{ij} .$$

при условии, что

$$\sum_j x_{ji}^k - \sum_i x_{ij}^k = d_{ij} .$$

$$\sum_k x_{ij}^k \leq u_{ij} + y_{ij} ,$$

$$y_{ij} \leq b_{ij} ,$$

$$x_{ij}^k , y_{ij} \geq 0 .$$

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 полагают равными 0, $u_{ij}=0$, а величины b_{ij} выбирают достаточно большими. Другим примером использования этой модели является задача минимизации стоимости передачи информации при фиксированных пропускных способностях дуг. В этом случае $b_{ij}=0$.

5.7. Эвристический алгоритм решения задачи синтеза сети связи

Как уже отмечалось, сетевые задачи большой размерности решают в основном приближенными эвристическими методами. Рассмотрим один из эвристических методов на примере задачи синтеза некоторой сети связи.

Одна из основных задач при разработке схемы сети связи —

это определение наивыгоднейшей конфигурации и распределения каналов на сети. При этом могут решаться и другие задачи:

1) выбор типа линий связи (кабельные, радиорелейные, воздушные и т. п.) с соответствующими системами передачи;

2) последующее распределение стандартных каналов на сети.

Структура сети — ее *топология* — это совокупность пунктов (узлов, станций и т. п.) и соединяющих их линий связи или каналов в их взаимном расположении. Она показывает потенциальные возможности обеспечения связью отдельных пунктов этой сети.

Экономически оптимальный выбор сетки линий, их емкостей и плана распределения каналов при заданном наборе пунктов сети называют *задачей оптимального синтеза структуры сети*. Ее решение существенно зависит от того, насколько тщательно отобраны допустимые направления.

В качестве основных критериев оптимальности построения сети обычно рассматривают объем капитальных затрат, расход кабельной продукции и т. д. Минимум расхода кабельной продукции обеспечивается при минимуме общей протяженности трасс кабельных и радиорелейных линий связи. Иногда в качестве критерия оптимальности может рассматриваться и объем строительно-монтажных работ.

Реальная сеть должна отвечать одновременно в той или иной мере всем критериям оптимальности, т. е. задача определения наивыгоднейшей конфигурации и распределения каналов сети является *многокритериальной*. Но в связи со сложностью решения многокритериальных задач обычно в качестве оптимальной принимают сеть, отвечающую одному, какому-либо заранее заданному, критерию оптимальности. Очевидно, что сеть будет зависеть от выбранного критерия оптимальности. И в общем случае будет различной в зависимости от используемого критерия оптимальности.

Рассмотрим задачу синтеза сети при следующих требованиях:

1. Каждый пункт сети должен иметь два независимых пути к центральному узлу (ЦУ) (прямой и обходной пути). Прямой путь должен обеспечивать передачу по 70—75% каналов от их расчетного числа, а обходной путь — соответственно по 25—30% каналов. Это соотношение может измениться, а поэтому должно задаваться на входе задачи.

2. Должен иметься путь, соединяющий каждый пункт сети с дополнительным узлом связи (ДУ), не совпадающим с ЦУ. На некоторых участках этот путь может совпадать с прямым и обходными путями.

В сетях связи накладывают дополнительные требования:

3. На линиях связи пункт — ЦУ число транзитов по высокой частоте (ВЧ) должно быть не более двух в случае применения аналоговых систем передачи.

4. Суммарная емкость (число каналов) на любом участке зоны сети должна быть кратна 12 для аналоговых систем передачи и 30 для цифровых систем.

5. Иногда требуется предусмотреть возможность того, чтобы через некоторые пункты проходило минимальное число путей. Эти пункты отмечают в исходных данных.

Задача синтеза структуры сети может быть наиболее адекватно представлена (в силу нелинейного ступенчатого изменения целевых функций, в частности функции стоимости линии связи) как задача целочисленного нелинейного программирования в различных формах.

Рассмотрим задачу выбора оптимального варианта построения сети с двумя независимыми путями к единственному узлу (центральный узел), к которому передается информация со всех других узлов сети. В качестве целевой функции выступает функция стоимости сети.

Задан неориентированный граф (сеть) G с $N+1$ вершинами. Все вершины, кроме $(N+1)$ -й, являются источниками информации. Вершина $N+1$ является стоком (центральным узлом). Путем μ^k от вершины k к стоку $N+1$ будем называть последовательность вершин $(k, i, j, \dots, N+1)$, через которые проходит данный путь. Для каждой вершины k ($k = \overline{1, N}$) необходимо выбрать два пути к стоку, не имеющих общих вершин, кроме k и $N+1$. Заданы потоки информации P_1 и P_2 , которые требуется передать от вершины k к стоку соответственно по первому и второму путям. Дугам графа G приписаны веса, равные их длинам. Для общности рассмотрения заданный граф G дополним до полного дугами с достаточно большим весом вместо отсутствующих.

Цель задачи — выбор топологии сети и маршрутов передачи информации на ней, минимизирующих общую стоимость сети. Общая стоимость сети определяется как сумма стоимостей входящих в нее дуг. Стоимость дуги c_{ij} является функцией от длины дуги (i, j) и суммарного потока информации на ней — суммы потоков от вершины i к вершине j и от вершины j к вершине i . Конкретный вид функции стоимости не имеет принципиального значения и может быть произвольным. В дальнейшем предполагается, что функция стоимости возрастает с ростом расстояния и не убывает с ростом суммарного потока. Известно, что оптимальная сеть при различных минимизируемых величинах может быть получена по одному и тому же алгоритму при соответствующем выборе величины c_{ij} .

Введем следующие обозначения:

$x_{ij}^{1,k}$ — булева переменная, равная 1, если вершина j непосредственно следует за вершиной i на первом пути μ_1^k от вершины k к стоку, и равная 0 в противном случае ($i = \overline{1, N}, j = \overline{1, N+1}, i \neq j; k = \overline{1, N}$).

$x_{ij}^{2,k}$ — булева переменная, равная 1, если вершина j непосредственно следует за вершиной i на втором пути μ_2^k от вершины k к стоку, и равная 0 в противном случае ($i = \overline{1, N}, j = \overline{1, N+1}, i \neq j; k = \overline{1, N}$).

Q_{ij} — суммарный поток информации на дуге (i, j) , определяемый по формуле

$$Q_{ij} = \sum_{k=1}^N \sum_{l=1}^2 P_l^k (x_{ij}^{lk} + x_{ji}^{lk}); \quad (5.15)$$

$$i = \overline{1, N}; j = \overline{1, N+1}; i < j.$$

Тогда рассматриваемую задачу можно сформулировать следующим образом. Необходимо найти значения булевых переменных x_{ij}^{lk} , минимизирующих общую стоимость сети:

$$S = \sum_{i=1}^N \sum_{j=i+1}^{N+1} c_{ij} Q_{ij}, \quad (5.16)$$

где Q_{ij} — вычисляются по формуле (5.15), при ограничениях:

$$\sum_{j=1, j \neq k}^{N+1} x_{kj}^{lk} = 1, \quad \sum_{i=1, i \neq k}^N x_{ki}^{lk} = 0; \quad (5.17)$$

$$\sum_{i=1}^N x_{i, N+1}^{lk} = 1, \quad k = \overline{1, N}; l = 1, 2; \quad (5.18)$$

$$\sum_{i=1, i \neq r}^N x_{ir}^{lk} = \sum_{j=1, j \neq r}^{N+1} x_{rj}^{lk} \leq 1, \quad r = \overline{1, N}, k = \overline{1, N}, l = 1, 2. \quad (5.19)$$

$$\sum_{l=1}^2 \sum_{i=1, i \neq r}^N x_{ir}^{lk} \leq 1, \quad r = \overline{1, N}, k = \overline{1, N}. \quad (5.20)$$

Ограничение (5.17) означает, что началом как первого, так и второго путей от вершины k к стоку должна быть вершина k . Ограничение (5.18) предопределяет, что все пути должны оканчиваться в вершине $N+1$. Ограничение (5.19) указывает на условия сохранения потока (равенство) и на запрет на петли в путях от вершины к стоку (неравенство). Заметим, что если функция стоимости монотонно возрастает с ростом суммарного потока, то ограничение (5.19) в виде неравенства можно опустить. Ограничения (5.20) реализуют условие непересечения первого и второго путей от каждой вершины к стоку по вершинам.

Как видно, данная математическая постановка предполагает наличие $2N^3$ булевых переменных и $5N^2 + 6N$ ограничений, что в практически интересных случаях ($N > 20$) не позволяет рассчитывать на точное решение.

Исключение составляет случай, когда функция стоимости линейна относительно суммарного потока. Поскольку целевая функция (5.16) становится линейной по переменным x_{ij}^{lk} , это позволяет

осуществить декомпозицию исходной задачи на N независимых задач *булевого программирования* с целевой функцией

$$S_k = \sum_{i=1}^N \sum_{j=i+1}^{N+1} c'_{ij} P_l^k x_{ij}^{lk}, \quad (5.21)$$

где c'_{ij} — стоимость единицы потока по дуге (i, j) , и ограничениями (5.17) — (5.20) при фиксированных значениях $k (k = \overline{1, N})$.

Последняя задача (при фиксированном k) может быть сформулирована достаточно просто: найти два пути от вершины k к вершине $N+1$, не имеющих общих вершин, кроме k и $N+1$, и обеспечивающих минимум суммарной стоимости путей. При этом стоимость пути равна стоимости входящих в него дуг, а стоимость дуги (i, j) равна $c'_{ij} P_l^k$ для l -го пути ($l = 1, 2$).

Математическую постановку задачи с двумя независимыми путями к центральному узлу и одним путем к ДУ (дополнительному узлу) можно записать в следующем виде.

Пусть вершина с номером N является ДУ. Через μ_k будем обозначать путь от k вершины к ДУ ($k = \overline{1, N-1}$). При этом $\forall k = \overline{1, N-1}$ путь μ_k не должен содержать вершину $N+1$. Пусть x_{ij}^{3k} — булева переменная, равная 1, если вершина j связана с вершиной i на пути μ_k от вершины k к ДУ, и равная 0 в противном случае. Через P_3^k будем обозначать требуемые величины потоков информации от вершины k к ДУ ($k = \overline{1, N-1}$).

Тогда, придерживаясь прежних обозначений, суммарный поток на дуге (i, j) определяем по формуле

$$Q_{ij} = \sum_{k=1}^N \sum_{l=1}^2 P_l^k (x_{ij}^{lk} + x_{ji}^{lk}) + \sum_{k=1}^{N-1} P_3^k (x_{ij}^{3k} + x_{ji}^{3k}). \quad (5.22)$$

Для дуг $(i, N+1)$ ($i = \overline{1, N}$) суммарный поток, как и раньше, вычисляют по формуле (5.15).

Рассматриваемую задачу можно сформулировать следующим образом. Найти значения булевых переменных x_{ij}^{lk} ($l = \overline{1, 3}$), минимизирующих общую стоимость сети, вычисляемую по формуле (5.16) при ограничениях (5.17) — (5.20) и ограничениях:

$$\sum_{j=1, j \neq k}^N x_{kj}^{3k} = 1; \quad \sum_{i=1, i \neq k}^{N-1} x_{ik}^{3k} = 0; \quad k = \overline{1, N-1}; \quad (5.23)$$

$$\sum_{i=1}^{N-1} x_{iN}^{3k} = 1; \quad k = \overline{1, N-1}; \quad (5.24)$$

$$\sum_{i=1, i \neq r}^{N-1} x_{ir}^{3k} = \sum_{j=1, j \neq r}^{N-1} x_{rj}^{3k} \leq 1, \quad r = \overline{1, N-1}; \quad k = \overline{1, N-1}; \quad r \neq k. \quad (5.25)$$

Ограничения (5.23)—(5.25) аналогичны ограничениям (5.17)—(5.19). Аналог ограничения (5.20) отсутствует, поскольку к ДУ необходимо построить только один путь от каждой вершины.

Как легко видеть, требование наличия путей к ДУ увеличивает число переменных на $(N-1)^2$ и число ограничений на $(N-1)(N+2)$.

Но, как и ранее, в случае, когда функция стоимости дуги линейна от величины суммарного потока, задача распадается на две независимые задачи:

1) определение двух непересекающихся путей к центральному узлу;

2) определение путей к вершине N , имеющих минимальную суммарную стоимость.

Математическая постановка первой из этих задач уже рассматривалась, а вторая, как легко видеть, приводит к задаче построения кратчайших путей от каждой вершины к N -й.

В обеих постановках случайными величинами являются значения P_i^k , так как они получаются в результате прогнозов и в крайнем случае должны задаваться в виде интервалов с соответствующими значениями вероятностей. Другой случайной величиной может быть доля информации, передаваемой по прямому (или обратному) пути.

Полученная нелинейная целочисленная задача может быть решена на современных ЭВМ лишь для сети малых размеров: число пунктов 10 при 20—30 допустимых трассах. Размеры реальных сетей, как правило, больше. Поэтому в некоторых работах проводят линеаризацию данной задачи. В результате задачу сводят к целочисленной линейной или просто к задаче линейного программирования. Но в этом случае для получаемого решения не гарантируется определенная степень приближения критерия затрат к минимуму.

Большая размерность описанных задач и стохастическая природа исходных данных приводит к необходимости разработки эвристических алгоритмов решения. Кроме того, сама специфика структуры сетевых задач позволяет находить более эффективные алгоритмы поиска решения, чем алгоритмы в точных методах решения.

В литературе описан ряд эвристических методов, учитывающих специфику поставленной задачи. Но в этих алгоритмах для решения задачи используют, хотя и в разной степени, дуги минимальной длины. Следовательо, они ориентированы в некоторой степени на сеть минимальной длины, что не всегда позволяет найти наилучший результат, тем более этот недостаток будет сказываться при построении коротких сетей, в которых расстояния между узлами малы, так как в этом случае стоимость сети в основном определяется стоимостью аппаратуры и мало зависит от длины трасс.

Алгоритм синтеза сети без ДУ. Алгоритм для решения общей задачи (5.15) — (5.20) состоит из трех этапов:

I этап — построение первых путей от всех вершин к стоку;

II этап — построение вторых путей от всех вершин к стоку;

III этап — улучшение сети, полученной на первых двух этапах.

Этап I.

Шаг 1. Строят начальную звездообразную сеть, в которой каждая вершина напрямую, без промежуточных узлов, соединена со стоком, и подсчитывают стоимость такой сети согласно заданной функции стоимости. Если для какой-то вершины связь со стоком отсутствует в исходном графе, то стоимость такой несуществующей дуги полагают равной достаточно большому числу.

Шаг 2. Для каждой вершины k , путь от которой является радиальным путем

$$\mu_1^k = (k, N+1),$$

вычисляют экономию T_k^r , которая будет получена при замене пути μ_1^k на путь

$$\mu_1^k(r) = (k, r, \mu_1^k)$$

и равна разности их стоимостей. При этом промежуточная вершина пробегает по всем вершинам, инцидентным вершине k (т. е. связанным с вершиной k разрешенными дугами), и по пути μ_1^k . Пути от вершины до стока не содержат несуществующих дуг. Затем из всех экономий T_k^r выбирают максимальную.

На рис. 5.6 приведен пример такой замены. Экономия достигается за счет ликвидации дуги $(k, N+1)$, но при этом может увеличиться стоимость дуг пути μ_1^r .

Шаг 3. Выбирают максимальное значение T_{k^*} из всех экономий T_k , $k=1, N$. Если $T_{k^*} \leq 0$, то этап I считают завершенным.

Шаг 4. Для выбранного T_{k^*} осуществляют замену пути $\mu_1^{k^*} = (k^*, N+1)$ на путь $\mu_1^{k^*} = (k^*, r, \mu_1^{r^*})$, на котором достигалась

максимальная экономия T_{k^*} . Осуществляют замену путей для всех вершин $V(k^*)$, пути от которых проходили по дуге $(k^*, N+1)$, т. е. вместо пути $\mu_1^{V(k^*)} = (V(k^*), \dots, k^*, N+1)$, формируют новый путь $\mu_1^{V(k^*)} = (V(k^*), \dots, k^*, r, \mu_1^{r^*})$.

На рис. 5.6 такими вершинами являются вершины j_1 и j_2 . Новый путь для вершины j_1 , например, будет $\mu_1^{j_1} = (j_1, k, r, i_1, i_2, N+1)$ вместо $\mu_1^{j_1} = (j_1, k, N+1)$.

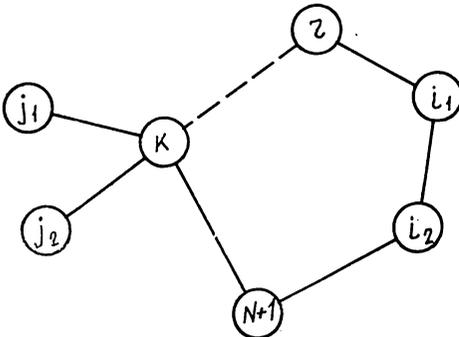


Рис. 5.6. Пример замены дуги $[\mu_1^r = (r, i_1, i_2, N+1)$; $\mu_1^k = (k, N+1)$; $\mu_1^k(r) = (k, r, i_1, i_2, N+1)$]

Шаг 5. Экономию T_k полагают равной 0 для всех $k = \overline{1, N}$. Переход к шагу 2.

Шаги 2—5 выполняют до тех пор, пока может быть достигнута экономия при замене какого-либо радиального пути на путь через промежуточную вершину.

Пересчет экономий T_k осуществляется после каждой замены радиального пути.

Поскольку каждый раз происходит уменьшение на единицу числа радиальных путей, то этап I конечен.

На этапе II осуществляют построение вторых путей от каждой вершины к стоку. Идея алгоритма на этапе II та же — замена радиального второго пути на путь через промежуточную вершину.

Шаг 1 аналогичен шагу 1 этапа II. Отличия: если первый путь от некоторой вершины после завершения этапа I оставался радиальным, то радиальный второй путь прокладывают по несуществующей второй дуге к стоку и его стоимость полагают вдвое большей, чем стоимость просто несуществующей дуги. Смысл этого в том, что такие пути выгоднее всего заменять, а значит, они будут заменены в первую очередь. Кроме того, при подсчете стоимости сети учитывают первые пути от всех вершин к стоку.

Шаг 2 аналогичен шагу 2 этапа I. Только теперь существует две возможности замены радиального второго пути от k -й вершины на путь через промежуточную вершину r , а именно:

$$\mu_2^{k,1}(r) = (k, r, \mu_1^r), \mu_2^{k,2}(r) = (k, r, \mu_2^r),$$

т. е. новый путь идет от вершины k к вершине r , а затем либо по первому, либо по второму пути от вершины r к стоку (конечно, при этом второй путь μ_2^r должен быть «реальным»).

Обозначим через $V(k)$ множество вершин, вторые пути которых проходят по дуге $(k, N+1)$, при этом $k \in V(k)$. Тогда путь μ можно заменить на путь $\mu_2^{k,1}$ или $\mu_2^{k,2}$ только в том случае, если $\forall j \in V(k) \mu_1^j \cap \mu_2^{k,1} \in \{k, N+1\}$ или $\mu_1^j \cap \mu_2^{k,2} \in \{k, N+1\}$ соответственно. Схематично вышесказанное показано на рис. 5.7.

Шаги 3—5 аналогичны шагам I этапа.

Шаги 2—5 выполняются до тех пор, пока замена какого-либо радиального пути на путь через промежуточную вершину может дать положительную экономию.

После выполнения I и II этапов мы получим сеть с размеченными на ней маршрутами первых и вторых путей от каждой вершины к стоку. Эта сеть с маршрута-

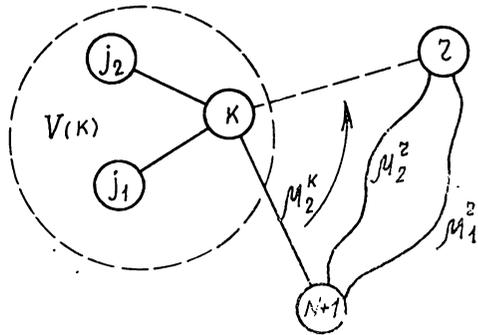


Рис. 5.7. Пример замены пути

ми удовлетворяет всем условиям задачи, т. е. является допустимой. Кроме того, она обладает следующими особенностями.

1. Каждая вершина $k = \overline{1, N}$ имеет не менее двух инцидентных вершин сети, поскольку осуществляет два непересекающихся по вершинам, а следовательно, и по дугам пути. Инцидентными вершинами называют вершины, связанные разрешенными дугами.

2. По каждой дуге сети (i, j) согласно выбранному маршрутам обоих путей происходит передача информации либо только в одну сторону, например от вершины i к вершине j , либо в обе стороны. В первом случае дугу (i, j) ориентируем от i к j , а во втором случае дуга (i, j) будет ориентирована в обе стороны. После ориентации сети описанным образом получим, что каждая вершина сети имеет ровно две выходящих из нее дуги (кроме, конечно, стока). Входящих же дуг может быть произвольное число. Это следует из алгоритма замены радиального пути на путь через промежуточную вершину.

3. Если же по дуге (i, j) от вершины i к вершине j идет поток информации P_{ij} , то в дальнейшем он полностью проходит по первому или второму пути от вершины j к стоку. Этот факт также следует из алгоритмов I и II этапов.

Особенности сети, отмеченные в пп. 2 и 3, облегчают реализацию III этапа.

Если после выполнения первых двух этапов каждая вершина сети имеет ровно две инцидентные вершины (кроме, разумеется, стока), то полученная сеть не будет обладать избыточностью. В этом случае возможность улучшения сети заключается только в попытке заменить какой-либо радиальный первый путь. Если же некоторые вершины имеют более двух инцидентных вершин, то полученная сеть обладает известной избыточностью. На III этапе осуществляют попытку улучшить сеть с точки зрения ее общей стоимости в обоих случаях.

Шаг 1. Составляют множество дуг — претендентов на замену. В него включают все дуги (i, j) , ориентированные лишь в одну сторону ($i = \overline{1, N}$, $j = \overline{1, N+1}$, $i \neq j$). Обозначим это множество через D . Для каждой дуги $(i, j) \in D$ составляют множество вершин $V(i, j)$, пути от которых проходят по дуге (i, j) , и множество путей $L(i, j)$ [первые или вторые пути от вершин множества $V(i, j)$, проходящие по дуге (i, j)].

Шаг 2. Для каждой дуги $(k, m) \in D$ (рис. 5.8) вычисляют экономию, которая может быть получена при замене дуги (k, m) на дугу (k, r) с соответствующим изменением маршрутов первых или вторых путей для всех вершин $j \in V(k, m)$ на путь $\mu_l^{j,1} = (j, \dots, k, r, \mu_l^r)$ или путь $\mu_l^{j,2} = (j, \dots, k, r, \mu_l^r)$ ($l=1, 2$), т. е. новый путь идет, как и раньше, до вершины k , а далее через вершину r и по первому или второму пути от вершины r к стоку. Такую замену можно осуществить лишь в том случае, когда:

1) для любой вершины $j \in V(k, m)$ другой путь не имеет общих вершин с μ_l^r , кроме $(N+1)$ -й вершины;

2) сам путь μ_i^1 не содержит вершин из $V(k, m)$, т. е.

$$\mu_i^1 \cap V(k, m) = \emptyset.$$

Экономии при этом достигают за счет ликвидации дуги (k, m) , а также за счет уменьшения потока на $P_{k,m}$ для остальных дуг пути μ_i^k . Но при этом может произойти увеличение стоимости сети от включения в сеть дуги (k, r) или увеличения на $P_{k,m}$ потока на дуге (k, r) и на дугах путей μ_1^r или μ_2^r .

Шаг 3. Выбирают дугу $(k, m) \in D$ и соответствующую вершину r , на которых достигается максимальная экономия. Если максимальная экономия положительна, то выполняют шаг 4, в противном случае — завершение III этапа.

Шаг 4. Осуществляют модификацию сети, маршрутов путей, множеств D , V и L в соответствии с выбранной на шаге 3 дугой (k, m) , вершиной r и путем от вершины r к стоку. После этого — переход к шагу 2.

После завершения III этапа работа эвристического алгоритма считается законченной.

Алгоритм синтеза сети с ДУ. Схема эвристического алгоритма для решения задачи (5.15)—(5.25) приведена на рис. 5.9.

На каждой итерации осуществляют последовательное выполнение вышеописанных трех этапов эвристического алгоритма построения двух непересекающихся путей к централь-

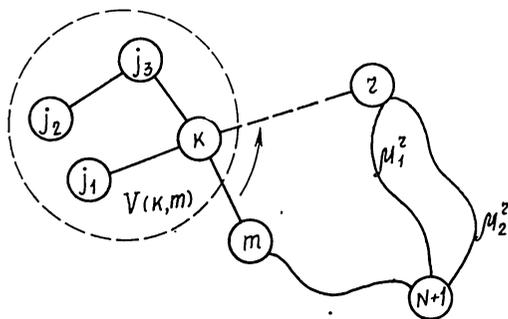


Рис. 5.8. Пример замены дуг с изменением маршрутов первых или вторых путей

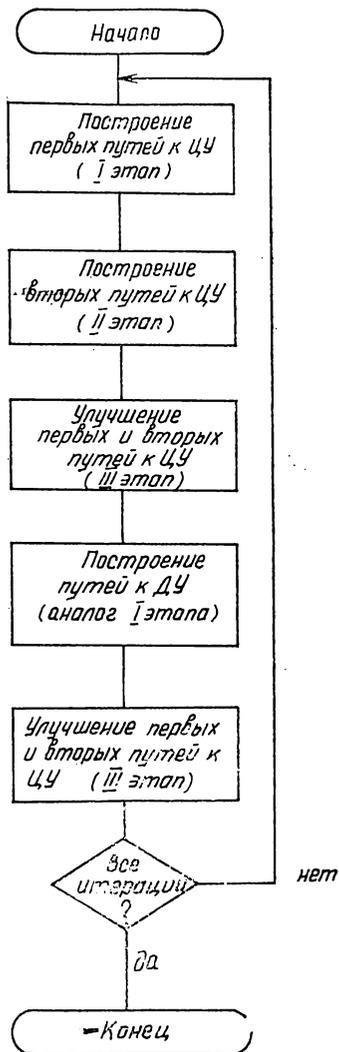


Рис. 5.9. Схема эвристического алгоритма

ному узлу, после чего осуществляют построение путей к ДУ с помощью III этапа эвристического алгоритма и проводят попытку **уменьшить стоимость сети** путем замены некоторых маршрутов путей к центральному узлу с учетом новых путей к ДУ с помощью III этапа эвристического алгоритма. На каждой итерации (кроме первой) при построении первых и вторых путей к центральному узлу учитывают потоки информации, протекающие по сети согласно построенным на предыдущей итерации путям к ДУ. А при построении путей к ДУ учитывают потоки информации, протекающие по сети согласно построенным ранее первым и вторым путям к центральному узлу. Для учета существующих на сети потоков информации в эвристическом алгоритме изменяют лишь подсчет экономии при замене одного пути на другой.

Итерационный процесс продолжают до тех пор, пока осуществление очередной итерации приводит к изменению сети и потоков на ней или пока не будет исчерпан заданный лимит итераций. В качестве решения задачи принимают тот вариант сети, при котором достигается минимальное значение общей стоимости сети.

Как показал опыт многих экспериментальных расчетов, требуемое для достижения минимума число итераций зависит от соотношения величин $\min[P_k^1, P_k^2]$ и P_k^3 . Чем ближе эти значения, тем больше требуется итераций (но их число, например, не превосходило 5 при $N=25$). Если же потребности путей к запасному узлу существенно меньше потребностей к центральному узлу, то лучший результат, как правило, достигается на первой итерации.

Построение сети с учетом ограничений по транзитам. Доработку изложенного алгоритма для построения сети и распределения каналов с учетом ограничения по транзитам осуществляют следующим образом.

Под транзитностью пути в настоящей работе понимают число дуг в этом пути. Ограничение на транзитность формулируют так: построить сеть, в которой число дуг в маршрутах прямых и обходных путей к ЦУ и путей к дополнительному узлу ДУ не будет превышать некоторого заданного числа (обозначим его τ). Для этого достаточно на шаге 2 этапов I—III описанного алгоритма рассматривать в качестве новых только пути, удовлетворяющие условию транзитности.

Для i -й вершины необходимо знать:

1) число дуг на участке пути между i -й вершиной и вершиной, наиболее удаленной от i -й по числу дуг, путь от которой проходит через прямой или обходной путь от i -й вершины к ЦУ и путь от i -й вершины к ДУ. Обозначим эти числа соответственно $A_1(i)$, $A_2(i)$, $A_3(i)$;

2) число дуг в прямом, обходном пути к ЦУ и пути к ДУ от i -й вершины; обозначим эти числа соответственно $B_1(i)$, $B_2(i)$, $B_3(i)$.

В процессе учета ограничения по транзитам при построении прямых путей к ЦУ при замене пути $\mu_1^k = (k, N+1)$ на путь

$\mu_1^k(r) = (k, r, \mu_1^r)$ в качестве r берем лишь те вершины, для которых выполняется неравенство

$$A_1(k) + B_1(r) < \tau.$$

Если от k -й вершины, которая соединена с центром фиктивной дугой, нет маршрута, удовлетворяющего ограничению по транзитности, то путь от k -й вершины строят без ограничения на транзитность после того, как построены все маршруты прямых путей, удовлетворяющие ограничению на транзитность.

При построении нового маршрута обходного пути с учетом ограничения по транзитности для k -й вершины в качестве вариантов замены рассматривают лишь такие вершины r , для которых выполняются требования:

1) если новый маршрут пройдет через прямой путь от r , то

$$A_2(k) + B_1(r) < \tau;$$

2) если новый маршрут пройдет через обходной путь от r , то

$$A_2(k) + B_2(r) < \tau.$$

Если вершина k соединена с центром фиктивной дугой и нет маршрута, удовлетворяющего требованию транзитности, то маршрут от k строят без ограничения на транзитность после построения всех маршрутов обходных путей, которые удовлетворяют ограничению на транзитность.

Учет ограничения на транзитность на этапе оптимизации построения прямых и обходных путей показан на рис. 5.10. Здесь W — множество вершин, пути от которых проходят по дуге (i, k) . Замена дуги (i, k) на дугу (i, j) на этом этапе происходит при следующих условиях:

1) транзитность пути от k -й вершины (старый путь) не меньше транзитности пути от j -й вершины (новый путь); замену производят, если при этом есть выгода (если экономия положительна);

2) транзитность пути от j -й вершины превышает транзитность пути от вершины k ;

3) если ранее построенный путь от i -й вершины не удовлетворял ограничению на транзитность, то замена происходит в том случае, когда она выгодна;

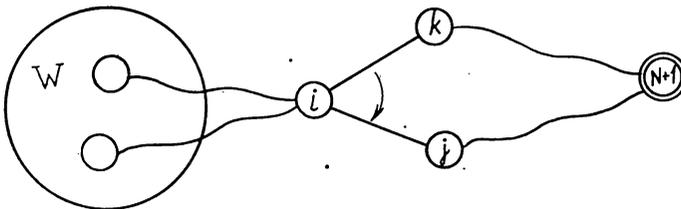


Рис. 5.10. Учет ограничения на транзитность

4) если путь от i -й вершины удовлетворял ограничению на транзитность, но путь хотя бы от одной вершины из множества W (рис. 5.10) многотранзитный, то замену производят тогда и только тогда, когда, во-первых, она выгодна и, во-вторых, если при этом не появляются новые недопустимые по транзитности маршруты (прежние могут оставаться многотранзитными);

5) если путь от вершины i и путь от любой другой вершины из множества W удовлетворяют ограничению на транзитность, то замену производят, когда она выгодна и не создает недопустимых по транзитности маршрутов.

При построении путей к ДУ учет ограничения по транзитам проводится следующим образом.

При замене $\mu_3^k = (k, N)$ на путь $\mu_3^k(r) = (k, r, \mu_3^r)$ проверяют выполнение неравенства

$$A_3(k) + B_3(r) < \tau.$$

После каждого изменения маршрутов производят пересчет чисел A и B у тех вершин, в маршруты которых были внесены изменения.

Рассмотренный эвристический алгоритм не накладывает серьезных ограничений на функцию стоимости дуг и может быть использован для решения широкого круга задач:

построение сети кратчайшей длины или минимальной стоимости;

построение сети минимальной по величине задействованных каналов-километров;

распределение маршрутов на сети при наличии ограничений пропускных способностей дуг с целью достижения максимальной суммарной свободной емкости.

Кроме того, алгоритм позволяет легко учесть дополнительные требования по качеству передачи информации, например ограничение на транзитность.

Примеры синтеза сети с детерминированными исходными данными. Для иллюстрации работы эвристического алгоритма рассмотрим граф, который приведен на рис. 5.11, где изображены допустимые трассы прокладки кабеля с указанием протяженности каждой дуги графа в километрах. Центральным узлом (ЦУ) считают узел 1, а дополнительным узлом (ДУ) — узел 2. Потребности в каналах первого и второго путей к ЦУ и пути к ДУ для каждого узла сети приведены в табл. 5.10. Стоимость 1 км дуги зависит от стоимости системы передачи, установленной на дуге, и составляет 2 тыс. руб. при емкости 120 каналов, 4 тыс. руб. при емкости 240 каналов и 5 тыс. руб. при емкости 600 каналов. Такой вид функции стоимости линий связи в общем виде соответствует реальной ситуации. Кроме того, в проектируемой сети между узлами 1 и 2 уже имеется система передачи емкостью 480 каналов, между узлами 1 и 3 — 600 каналов, между узлами 2 и 8, 3 и 15 — по 120 каналов.

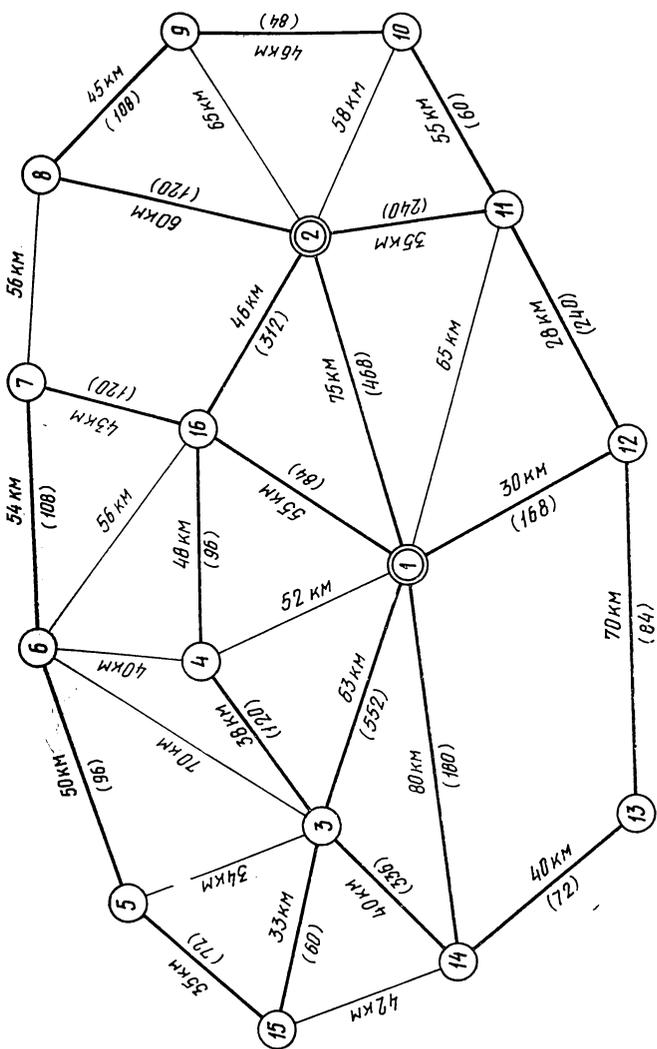


Рис. 5.11. Допустимые трассы прокладки кабеля

Исходные данные для задачи синтеза сети

Наименование пути	Потребности узлов (в каналах)														
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Первый путь к ЦУ	60	120	220	24	24	12	24	36	24	96	36	12	240	12	48
Второй путь к ЦУ	36	60	72	12	24	12	24	24	12	60	36	12	120	12	48
Путь к ДУ	0	24	24	12	12	12	12	12	12	24	12	12	36	12	48

В табл. 5.11 приведены результаты расчетов по описанному алгоритму. Суммарное число свободных каналов на сети вычисляются как суммарную разность между емкостью установленной системы передач и потоком информации на каждой дуге. Дуги, включенные в оптимальную сеть, изображены на рис. 5.11 жирными линиями, в скобках указано число используемых каналов на дуге. Время счета примера составляет около 1 мин на ЭВМ ЕС1033.

Таблица 5.11

Решение задачи синтеза сети

Параметр	Эвристический алгоритм
Стоимость сети, тыс. руб.	2280
Число дуг, штук	22
Протяженность сети, км	1069
Суммарная свободная емкость каналов	1260

На рис. 5.12 приведена сеть, полученная в результате работы эвристического алгоритма построения двух независимых путей к ЦУ без ДУ. Стоимость полученной сети составила 2124 тыс. руб. Маршруты первых и вторых путей сети существенно отличаются от маршрутов первых и вторых путей на рис. 5.11, т. е. при выполнении итерационного процесса маршруты первых и вторых путей претерпевают значительные изменения. Следует отметить, что требование наличия путей к ДУ увеличивает на 15,7% общую потребность в линиях связи по первому и второму путям к ЦУ, а суммарную стоимость сети увеличивает лишь на 7,3%.

На тех же исходных данных с ЦУ и ДУ был проведен расчет в случае, когда стоимость дуги не зависит от расстояния, а определяется только используемой системой передачи. Стоимости системы передач были взяты следующие: емкостью 120 каналов —

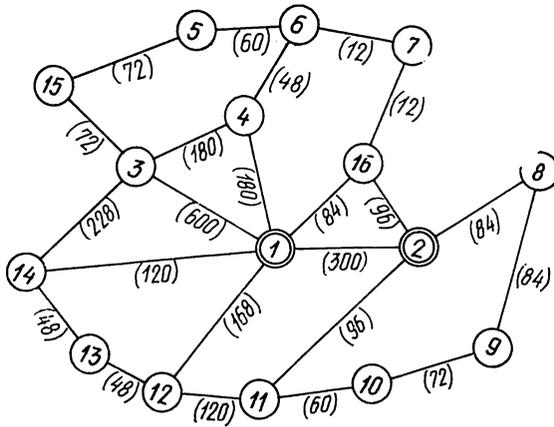


Рис. 5.12. Синтез сети согласно эвристическому алгоритму (сеть без ДУ — два независимых пути к ЦУ)

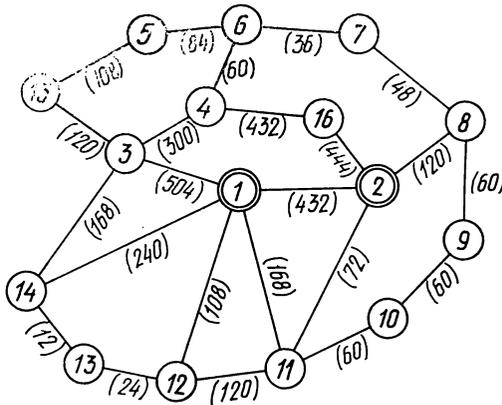


Рис. 5.13. Синтез сети согласно эвристическому алгоритму, когда стоимость дуги не зависит от расстояния

2 тыс. руб.; 240 каналов — 4 тыс. руб.; 600 каналов — 5 тыс. руб. Сеть, полученная в результате работы предложенного алгоритма, приведена на рис. 5.13. Число дуг сети 23. Протяженность сети 1132 км. Суммарная свободная емкость 2460 каналов.

5.8. Методы внутренней точки для задачи математического программирования

Рассмотрим общую задачу математического программирования, не содержащую ограничений в виде равенств, т. е.

минимизировать $f(x)$

при ограничениях $g_i(x) \geq 0; i = \overline{1, m}$.

Пусть вблизи локального минимума этой задачи x^* существует окрестность, где есть такая точка x^0 , в которой $g_i(x^0) \geq 0$ ($i = \overline{1, m}$) и выполняются условия строгой дополняющей нежесткости: $u_i(x^*) > 0$, если $g_i(x^*) = 0$, $i = \overline{1, m}$.

Видоизменим достаточные условия локального минимума в точке x^* , сформулированные в § 1.5. Предположим, что при малом $r > 0$ в точке $[x(r), u(r)]$ вблизи точки (x^*, u^*) выполняются условия

$$\begin{cases} g_i(x) \geq 0, & i = \overline{1, m}; \\ u_i g_i(x) = r > 0, & i = \overline{1, m}; \\ u_i \geq 0, & i = \overline{1, m}; \\ \nabla f(x) - \sum_{i=1}^m u_i \nabla g_i(x) = 0. \end{cases}$$

Из второго условия выразим u_i и подставим это выражение в последнее равенство:

$$\nabla f[x(r)] - \sum_{i=1}^m \frac{r}{g_i[x(r)]} \nabla g_i[x(r)] = 0.$$

Непосредственной проверкой легко установить, что левая часть этого выражения — градиент функции

$$L(x, r) = f(x) - r \sum_{i=1}^m \ln g_i(x),$$

обращающийся в нуль в точке $x(r)$. Причем $x(r)$ стремится к x^* при r , стремящемся к нулю. Функцию $L(x, r)$ в таком виде называют *логарифмической штрафной функцией*.

Аналогично получаем другой вид штрафной функции $L_1(x, r)$, полагая $u_i = \lambda_i^2$. Тогда $\lambda_i g_i(x) = r > 0$, $i = \overline{1, m}$,

$$\nabla f[x(r)] - \sum_{i=1}^m \frac{r^2}{g_i^2[x(r)]} \nabla g_i[x(r)] = 0,$$

и функция $L_1(x, r)$ примет вид

$$L_1(x, r) = f(x) + r^2 \sum_{i=1}^m \frac{1}{g_i(x)}.$$

Задавая последовательность значений $\{r_k\}$, стремящуюся к нулю, получаем последовательность $\{x(r_k)\}$, сходящуюся к x^* . С помощью новых функций $L(x, r)$ мы свели задачу на условный экстремум (задачу математического программирования) к задаче поиска безусловного экстремума функции $L(x, r)$. Точнее, задачу математического программирования заменили семейством функций, зависящих от параметра r и обладающих следующими свойствами:

1) в окрестности оптимальной точки они близки к заданной минимизируемой функции;

2) каждая функция из построенного семейства достаточно быстро возрастает при приближении к границе допустимой области из «внутренней» части допустимой области.

К минимизируемой функции исходной задачи мы добавили ряд слагаемых, называемых *штрафными (барьерными) функциями*, зависящими от параметра r и функции одного из ограничений. При фиксированном значении параметра r второе слагаемое стремится к бесконечности при стремлении к нулю его аргумента. Каждую функцию семейства подвергают безусловной минимизации, и этот процесс не может вывести x^* за пределы допустимой области. Подобные методы названы *методами внутренней точки*.

В задаче математического программирования при наличии ограничений-равенств допустимой области (в виде области) нет, и «метод внутренней точки» не применим.

Рассмотрим некоторые примеры перехода от задачи математического программирования к задаче безусловной минимизации «методом внутренней точки».

Пример. Минимизировать $f(x) = x_1 + x_2$
при ограничениях

$$\begin{aligned} g_1(x) &= -x_1^2 + x_2 \geq 0, \\ g_2(x) &= x_1 \geq 0. \end{aligned}$$

Решение. Построим логарифмическую штрафную функцию

$$L(x, r) = x_1 + x_2 - r \ln(-x_1^2 + x_2) - r \ln x_1.$$

Определим точки минимума $L(x, r)$ аналитически, так как $L(x, r)$ дифференцируема в рассматриваемой области. Для нахождения $x_1(r)$ и $x_2(r)$ получим систему уравнений

$$\begin{cases} \frac{\partial L}{\partial x_1} = 1 + \frac{r \cdot 2x_1}{-x_1^2 + x_2} - \frac{r}{x_1} = 0; \\ \frac{\partial L}{\partial x_2} = 1 - \frac{r}{-x_1^2 + x_2} = 0. \end{cases}$$

Откуда найдем

$$x_1(r) = \frac{-1 \pm \sqrt{1 + 8r}}{4};$$

здесь оставим только знак «+», так как $x_1 \geq 0$;

$$x_2(r) = \frac{(-1 \pm \sqrt{1 + 8r})^2}{16} + r.$$

Заметим, что значения $x_1(r)$ и $x_2(r)$ удовлетворяют условию положительной определенности матрицы $\nabla^2 L$. Дадим последовательность значений r : 1,0; 0,5; 0,25; 0,1. Соответственно получим последовательности значений

$$x_1(r) = 0,5; 0,309; 0,183; 0,085;$$

$$x_2(r) = 1,25; 0,595; 0,283; 0,107,$$

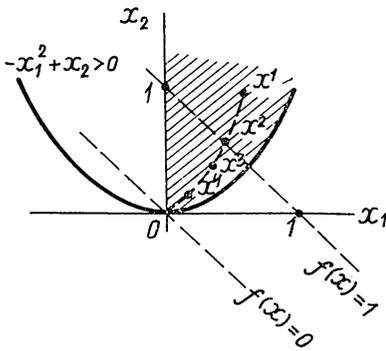


Рис. 5.14. Графическое решение задачи математического программирования методом внутренней точки

Решение. Построим логарифмическую штрафную функцию

$$L(x, r) = x_1 - r \ln x_1.$$

Необходимое условие существования минимума:

$$\frac{dL}{dx_1} = \frac{\partial L}{\partial x_1} = 1 - \frac{r}{x_1} = 0.$$

Откуда $x_1(r) = r$.

Из условия $\frac{d^2L}{dx_1^2} > 0$ следует, что точка минимума будет при $r < 1$. Для $r = 1$ в любой точке $x_1 \geq 1$ будет минимум, для $r > 1$ минимум будет только при $x_1 = +\infty$.

5.9. Методы внешней точки для задачи математического программирования

Попытаемся приблизиться к оптимальной точке из *недопустимой* области. Для чего преобразуем достаточные условия локального минимума задачи математического программирования следующим образом:

1) рассмотрим ограничения в ослабленной форме:

$$g_i(x) \geq -r; \quad i = \overline{1, m}; \quad r > 0; \quad (5.26)$$

2) преобразуем условие дополняющей нежесткости $u_i g_i(x^*) = 0$ так, чтобы оно имело смысл для отрицательных значений $g_i(x)$ и сводилось к исходному условию при $r \rightarrow 0$ (это означает, что $u_i(x^*) = 0$ при $g_i(x^*) \neq 0$; если $g_i(x^*) = 0$, то $u_i(x^*)$ принимает любые значения), т. е.

$$u_i(r) = -\min[0, g_i(x)]. \quad (5.27)$$

Откуда если $r > 0$ и мало, $x(r)$ удовлетворяет (5.26) и $g_i[x(r)] = 0$ для некоторых i , то $u_i \equiv 0$ и $\lim_{r \rightarrow 0} u_i(r) = u_i(x^*) = 0$. Если

сходящиеся при $r \rightarrow 0$ к точке $(0, 0)$. Графическое решение данной задачи представлено на рис. 5.14.

В общем случае в задачах со многими локальными минимумами (при слабых условиях регулярности) существует последовательность безусловных локальных минимумов, сходящихся к каждому из условных локальных минимумов.

Рассмотрим второй пример: решение задачи линейного программирования методом внутренней точки.

Пример. Минимизировать x_1 при условии $g_1(x) = x_1 > 0$.

$g_1[x(r)] < 0$, то $\lim_{r \rightarrow 0} [-\min(0, g_i[x(r)])] = 0$, так как из (5.26) следует, что точка $x(r)$ при $r \rightarrow 0$ находится в допустимой области. В силу (5.26) $g_i[x(r)] \geq -r$, а из (5.27) вытекает, что $\lim_{r \rightarrow 0} u_i(r) \cdot r = 0$, поэтому $\lim_{r \rightarrow 0} u_i(r) \cdot g_i[x(r)] = 0$.

Условие дополняющей нежесткости в данном случае выполняется в пределе (из (5.27) следует, что $u_i \geq 0, i = \overline{1, m}$).

Аналогично преобразуют другие достаточные условия:

$$u_i(r) \geq 0, i = \overline{1, m};$$

$$\nabla f[x(r)] - \sum_{i=1}^m u_i(r) \nabla g_i[x(r)] = 0 \quad (5.28)$$

и для каждого вектора y , для которого $y^T \nabla g_i[x(r)] = 0$ при всех $i \in D^* = \{i | u_i^* > 0\}$, справедливо неравенство

$$y^T \left\{ \nabla^2 f[x(r)] - \sum_{i \in D^*} u_i(r) \nabla^2 g_i[x(r)] \right\} y > 0.$$

Подставим (5.27) в (5.28):

$$\nabla f[x(r)] + \sum_{i=1}^m \frac{1}{r} \min\{0, g_i[x(r)]\} \nabla g_i[x(r)] = 0.$$

Опять непосредственной проверкой убеждаемся, что функция, для которой выполняются названные условия, имеет вид

$$T(x) = f(x) + \sum_{i=1}^m \frac{1}{2r} \{ \min\{0, g_i(x(r))\} \}^2.$$

Это и есть функция, минимизируемая методом внешней точки. В нее могут входить ограничения и в виде неравенств, и в виде равенств. Можно доказать, что все необходимые условия минимума этой функции выполняются, например $\nabla^2 T$ есть положительно определенная матрица. Причем для ограничений-неравенств

$$(\min\{0, g_i[x(r)]\})^2 \equiv \left[\frac{g_i - |g_i|}{2} \right]^2.$$

Для ограничений-равенств $h_j[x(r)]$ можно записать

$$\begin{aligned} g_{j1}(x) &= h_j(x) \geq 0; \\ g_{j2}(x) &= -h_j(x) \geq 0. \end{aligned}$$

Откуда $g_{j1} = -g_{j2}$, или

$$\begin{aligned} \frac{1}{r} \left\{ \left[\frac{g_{j1} - |g_{j1}|}{2} \right]^2 + \left[\frac{-g_{j1} - |-g_{j2}|}{2} \right]^2 \right\} &= \\ &= \frac{g_{j1}^2}{r} = \frac{\{h_j[x(r)]\}^2}{r}. \end{aligned}$$

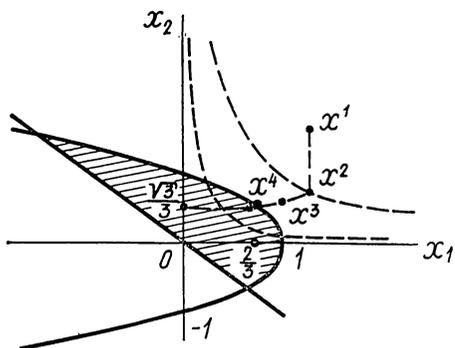


Рис. 5.15. Графическое решение задачи математического программирования методом внешней точки

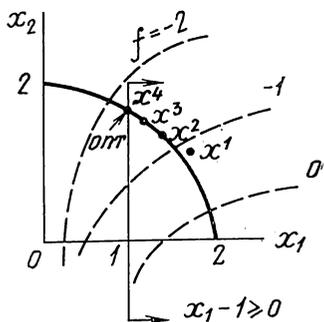


Рис. 5.16. Графическое решение задачи математического программирования с помощью комбинированного метода

Рассмотрим пример применения *метода внешней точки* для решения задачи математического программирования.

Пример. Минимизировать $f(x) = -x_1x_2$ при ограничениях

$$g_1(x) \equiv -x_1 - x_2^2 + 1 \geq 0;$$

$$g_2(x) \equiv x_1 + x_2 \geq 0.$$

Решение. Составим функцию штрафа для метода внешней точки:

$$T(x, r) = -x_1x_2 + \frac{(-x_1 - x_2^2 + 1 | x_1 - x_2^2 + 1 |)^2}{4r} + \frac{(x_1 + x_2 - |x_1 + x_2|)^2}{4r}.$$

Из необходимого условия

$$\frac{\partial T}{\partial x_1} = 0, \quad \frac{\partial T}{\partial x_2} = 0$$

определим последовательность значений x_1 и x_2 , сходящуюся к решению.

Зададим последовательность значений r : 1,0; 0,5; 1/3; 0,1; получим соответствующие последовательности значений

$$x_1(r) : 0,89; 0,77; 0,73; 0,67;$$

$$x_2(r) : 0,64; 0,62; 0,61; 0,58.$$

Последовательность значений x_1 сходится к $2/3$, а последовательность значений x_2 к $\sqrt{3}/3$ (рис. 5.15).

5.10. Комбинированный метод внутренней и внешней точек

Введем новый параметр $t = 1/r$ и рассмотрим комбинированную функцию

$$V(x, r, t) = f(x) + s(r)L(x) + p(t)T(x),$$

где $s(r)$ — функция от $r (r \rightarrow 0)$ для метода внутренней точки; $L(x)$ — функция штрафа для метода внутренней точки; $p(t)$ — функция от $t (t \rightarrow \infty)$ для метода внешней точки; $T(x)$ — функция штрафа для метода внешней точки.

Покажем на примере, как применяется комбинированная функция $V(x, r, t)$ для решения задачи математического программирования.

Пример. Минимизировать $f(x) = \ln x_1 - x_2$
при ограничениях

$$g(x) \equiv x_1 - 1 \geq 0;$$

$$h(x) = x_1^2 + x_2^2 - 4 = 0; \quad x_2^2 = 4 - x_1^2.$$

Решение. Построим комбинированную функцию

$$V(x, r, t = r^{-1}) = \underbrace{\ln x_1 - x_2}_{f(x)} - \underbrace{r \ln(x_1 - 1)}_{s(r)L(x)} + \underbrace{r^{-1}(x_1^2 + x_2^2 - 4)^2}_{p(t)T(x)}.$$

С помощью комбинированной функции мы избежали использования модулей $|g(x)|$, которые появляются в методе внешней точки. Необходимые условия минимума функции $V(x, r, r^{-1})$ дают

$$\frac{\partial V}{\partial x_1} = \frac{1}{x_1} - \frac{r}{x_1 - 1} + 4r^{-1}(x_1^2 + x_2^2 - 4)x_1 = 0;$$

$$\frac{\partial V}{\partial x_2} = -1 + 4x_2r^{-1}(x_1^2 + x_2^2 - 4) = 0.$$

Откуда

$$4r^{-1}(x_1^2 + x_2^2 - 4) = \frac{1}{x_2} = \frac{1}{\sqrt{4 - x_1^2}}.$$

Зададим последовательность значений $r : 1,0; 1/4; 1/16; 1/64; 1/256$ и получим соответствующие ей последовательности значений:

$$x_1(r) : 1,553; 1,159; 1,040; 1,010; 1,002;$$

$$x_2(r) : 1,334; 1,641; 1,711; 1,727; 1,731;$$

$$V : -0,2648; -1,0285; -1,4693; -1,6447; -1,7048.$$

Последовательности значений x_1 и x_2 дают оптимальное решение $(1, \sqrt{3})$. Графическое решение задачи математического программирования приведено на рис. 5.16.

СПИСОК ЛИТЕРАТУРЫ

1. Акулич И. Л. Математическое программирование в примерах и задачах. — М.: Высшая школа, 1986. — 317 с.
2. Беллман Р. Динамическое программирование: Пер. с англ./Под ред. Н. Н. Воробьева. — М.: ИЛ, 1960. — 400 с.
3. Беллман Р., Дрейфус С. Прикладные задачи динамического программирования: Пер. с англ./Под ред. А. А. Первозванского. — М.: Наука, 1965. — 458 с.
4. Вентцель Е. С. Исследование операций. — М.: Сов. радио, 1972. — 551 с.
5. Вильямс Н. Н. Параметрическое программирование в экономике (методы оптимальных решений). — М.: Статистика, 1976. — 96 с.
6. Гольштейн Е. Г., Юдин Д. Б. Новые направления в линейном программировании. — М.: Сов. радио, 1966. — 524 с.
7. Зангвилл У. И. Нелинейное программирование: Пер. с англ./Под ред. Е. Г. Гольштейна. — М.: Сов. радио, 1973. — 312 с.
8. Зуховицкий С. И., Авдеева Л. И. Линейное и выпуклое программирование (справочное руководство). — М.: Наука, 1964. — 348 с.
9. Исследование операций. Методологические основы и математические методы: Пер. с англ./Под ред. И. М. Макарова, И. М. Бескровного, т. 1. — М.: Мир, 1981. — 712 с.
10. Исследование операций. Модели и применение: Пер. с англ./Под ред. И. М. Макарова, И. М. Бескровного, т. 2. — М.: Мир, 1981. — 677 с.
11. Калихман И. Л. Сборник задач по математическому программированию. — М.: Высшая школа, 1975. — 270 с.
12. Карпелевич Ф. И., Садовский Л. Е. Элементы линейной алгебры и линейного программирования. — М.: Наука, 1965. — 275 с.
13. Лазарев В. Г., Лазарев Ю. В. Динамическое управление потоками информации в сетях связи. — М.: Радио и связь, 1983. — 216 с.
14. Мартин Дж. Системный анализ передачи данных: Пер. с англ./Под ред. В. С. Лапина. — М.: Мир, 1975, т. 2. — 431 с.
15. Монаков В. М., Беляева Э. С., Краснер Н. Я. Методы оптимизации. Пособие для учителя. — М.: Просвещение, 1978. — 175 с.
16. Муртаф Б. Современное линейное программирование: Теория и практика. Пер. с англ./Под ред. И. А. Станевичуса. — М.: Мир, 1984. — 224 с.
17. Рокафеллор Р. Выпуклый анализ: Пер. с англ./Под ред. А. Д. Иоффе, В. М. Тихомирова. — М.: Мир, 1973. — 469 с.
18. Сухарев А. Г., Тимохов А. В., Федоров В. В. Курс методов оптимизации. — М.: Наука, Физматлит, 1986. — 326 с.
19. Ху Т. Целочисленное программирование и потоки в сетях: Пер. с англ./Под ред. А. А. Фрийдмана. — М.: Мир, 1974. — 419 с.
20. Фиакко А., Мак-Кормик Г. Нелинейное программирование. Методы последовательной безусловной минимизации: Пер. с англ./Под ред. Е. Г. Гольштейна. — М.: Мир, 1972. — 240 с.
21. Филлипс Д., Гарсиа-Диас А. Методы анализа сетей: Пер. с англ./Под ред. Б. Г. Сушкова. — М.: Мир, 1984. — 496 с.
22. Юдин Д. Б., Гольштейн Е. Г. Линейное программирование. Теория и конечные методы. — М.: Физматгиз, 1963. — 775 с.

СПИСОК МАТЕМАТИЧЕСКИХ СИМВОЛОВ

- = равно
- ≡ тождественно равно
- ∞ бесконечность
- ≠ не равно
- ≈ приближенно равно
- ∝ (∞) пропорционально
- < меньше
- > больше
- ∝ подобно
- ≤ меньше или равно
- ≥ больше или равно
- ≧ знак больше или меньше
- ⊂ знак содержания для множеств
- ∈ знак принадлежности
- ∉ знак непринадлежности
- ∪ объединение двух множеств
- ∩ пересечение двух множеств
- ∅ пустое множество
- < = > эквивалентно
- ⇒ следует
- ∀ квантор общности
- ! знак факториала: $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$
- Σ знак суммы:

$$\sum_{i=1}^n a_i \equiv a_1 + a_2 + \dots + a_n .$$

ОГЛАВЛЕНИЕ

Введение	3
Глава 1. Введение в математическое программирование	5
1.1. Общие положения математического программирования	5
1.2. Общая запись задачи математического программирования и ее виды	15
1.3. Некоторые сведения об экстремуме функции, частных производных, градиенте и производной по направлению	16
1.4. Особенности нахождения оптимальных решений в задачах математического программирования	20
1.5. Необходимые и достаточные условия оптимума в задачах математического программирования	23
1.6. Теория двойственности и недифференциальные условия оптимальности в задаче выпуклого программирования	27
1.7. Графическое решение задач математического программирования	29
1.8. Простейшая оптимизационная задача	32
Глава 2. Линейное программирование	33
2.1. Математическая постановка задачи линейного программирования	33
2.2. Симплекс-метод — основной метод решения задач линейного программирования	35
2.3. Метод полного исключения Жордана для решения систем линейных алгебраических уравнений	40
2.4. Как спланировать выпуск продукции швейному предприятию	42
2.5. Двойственность в задачах линейного программирования	48
2.6. Как оптимально организовать поставку грузов от поставщиков к потребителям	53
2.7. Задача о перевозках с перегрузкой	59
2.8. Целочисленное линейное программирование	61
2.9. Постановка задачи об оптимальном раскрое материалов (о минимизации отходов)	67
2.10. Задача о наилучшем использовании посевной площади	68
2.11. Задача о закреплении самолетов за воздушными линиями	69
2.12. Задача о назначениях (проблема выбора)	72
2.13. Задача об оптимальном распределении самолетов между войсками и учебными полигонами	75
2.14. Задача о рациональном соотношении между различными типами бронебойных снарядов	76
2.15. Задача о покрытии множества	78
Глава 3. Сетевые (потокосные) задачи	81
3.1. Основные определения и приложения потокосных моделей	81
3.2. Задача о покупке автомобиля	87
3.3. Задача о многополюсной кратчайшей цепи	91

3.4. Анализ сложности алгоритмов поиска кратчайших путей	96
3.5. Задача о назначениях	97
3.6. Задача размещения производства	101
3.7. Задача о максимальном потоке	103
3.8. Задача о многополюсном максимальном потоке	107
3.9. Задача коммивояжера	111
3.10. Задача о многополюсной цепи с максимальной пропускной способностью	120
Глава 4. Основы динамического программирования	124
4.1. Условия применимости динамического программирования	124
4.2. Задача об оптимальной загрузке транспортного средства неделимыми предметами	127
4.3. Задача о вкладе средств в производство	131
4.4. Задача о распределении средств поражения	135
4.5. Вычислительные аспекты решения задач методом динамического программирования	140
Глава 5. О развитии методов решения задач математического программирования	142
5.1. Основные направления развития методов решения задач математического программирования	142
5.2. Понятие о параметрическом программировании	143
5.3. Многопродуктовые потоки в сетях	151
5.4. Специальный класс целочисленных задач о многопродуктовом потоке	156
5.5. Приближенное решение многопродуктовой транспортной задачи методом агрегирования	157
5.6. Приложения задач о многопродуктовом потоке	161
5.7. Эвристический алгоритм решения задачи синтеза сети связи	164
5.8. Методы внутренней точки для задачи математического программирования	179
5.9. Методы внешней точки для задачи математического программирования	182
5.10. Комбинированный метод внутренней и внешней точек	185
Список литературы	186
Список математических символов	187

МГТУ им. Н. Э. БАУМАНА
НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ
«РАДИОЭЛЕКТРОНИКА И ЛАЗЕРНАЯ ТЕХНИКА»

предлагает

«КАСКАД» — аппарат для электромагнитной терапии
(разработчики *В. И. Козинцев* и *С. И. Шукин*)

Области применения: — травматология и ортопедия;
— спортивная медицина;
— физиотерапия;
— медицина катастроф;
— микрохирургия и трансплантология.

Результативность: — улучшение репаративных процессов в твердых и мягких тканях;
— ремиссия отечных явлений;
— улучшение регионарного кровотока;
— улучшение реологических и биохимических свойств крови.

В ходе фундаментальных и прикладных медико-технических исследований взаимосвязанных биоэлектрических и биомеханических процессов в твердых и мягких тканях организма определены превалирующие физические механизмы, ответственные за биоэлектrogenез как при нормальном функционировании организма, так и при широком классе заболеваний, связанных с нарушением функций опорно-двигательного аппарата и сосудистой системы человека. Установлено, что восстановление с помощью внешнего электромагнитного воздействия утраченной электрической активности биотканей приводит к стимуляции восстановительных процессов в пораженных тканях, системах и органах. В результате этих исследований был разработан и успешно апробирован в клиниках аппарат «Каскад».

«Каскад» создает в костной ткани верхних и нижних конечностей такие же токи, которые в них возникают при выполнении ими своих функций (опорный период шага для нижних конечностей и движение типа сгибания-разгибания для верхних). Воздействие осуществляется электромагнитным полем со специальной

амплитудно-временной характеристикой. Особенностью аппарата является функциональность параметров воздействия в зависимости от режимов.

В комплект поставки аппарата «Каскад» входят:

- программно-методическое пособие по НОУ-ХАУ технологии применения аппарата в вышеперечисленных областях;
- обучающая программа работе на аппарате;
- программа экспресс-оценки эффективности воздействия по данным неинвазивного измерения кровотока и индивидуальной оптимизации его режимов.

Программные средства реализованы на флоппи-дисках в среде MS-DOS. Пользователи, не располагающие ЭВМ типа IBM PC XT/AT (или совместимыми с ними), могут получить твердые копии указанных документов в виде справочных печатных материалов.

Эксплуатационные характеристики:

- габариты 360×160×400 мм;
- масса не более 12 кг;
- потребляемая мощность не более 300 ВА.

В настоящее время освоен серийный выпуск аппарата.

За справками обращаться по адресу:

107005, Москва, Б-5, 2-я Бауманская ул., д. 5, к. 389А.

Тел. 267-44-38

БИБЛИОТЕКА МГТУ им. Н. Э. БАУМАНА

КОМПЬЮТЕРНАЯ ТЕХНОЛОГИЯ В ИНФОРМАЦИОННОМ ОБСЛУЖИВАНИИ ЧИТАТЕЛЕЙ

* Библиотека МГТУ им. Н. Э. Баумана предлагает новую форму информационного обслуживания — информационные материалы о новых поступлениях в библиотеку университета по современным направлениям науки и техники в виде рефератов и (или) библиографических описаний. Их поставляют на дискетах, ознакомление с ними возможно на компьютерах заказчика.

* Вместе с информацией передается программа, обеспечивающая чтение, поиск и выдачу на принтер сведений о новых книгах или статьях в периодических изданиях. Программа имеет автоматический перевод на русский или английский языки для управления поиском информации.

* Отобранная и выведенная на принтер информация может являться заказом на книгу или журнал, которые будут переданы заказчику по системе межбиблиотечного абонемента. По желанию заказчика библиотека обеспечит оперативное копирование подобранных материалов.

* Компьютерная информационная технология предполагает использование 5-дюймовых дискет любого формата и компьютеров, программно-совместимых с IBM PC XT/AT.

* Тематика компьютерных информационных изданий может определяться заказчиком.

Справки по телефону: 261-95-86.

40 коп.