МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Нижегородский государственный университет им. Н.И. Лобачевского

А.А. Тюхтина

МЕТОДЫ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ

Часть 1

Учебно-методическое пособие

Рекомендовано методической комиссией механико-математического факультета для студентов ННГУ, обучающихся по направлениям подготовки 080500 «Бизнес – информатика» и 080100 «Экономика»

Нижний Новгород
2014

УДК 519.854
ББК 22.18
Т 98

Т 98 Тюхтина А.А. Методы дискретной оптимизации: Часть 1: Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2014. – 62 с.

Рецензент: доцент, кандидат физ.-мат. наук **А.В. Калинин**

В учебно-методическом пособии рассматриваются методы отсечений и методы ветвей и границ для решения задач целочисленного линейного программирования.

Данное пособие предназначено для студентов Нижегородского госуниверситета, обучающихся по направлениям «Бизнес – информатика» (профиль подготовки «Информатика и математика в анализе экономических систем и бизнеса») и «Экономика» (академическая магистерская программа «Математические методы анализа экономики»).

Ответственный за выпуск:
председатель методической комиссии механико-математического факультета ННГУ, канд. физ.-мат. наук, доцент **Н.А. Денисова**

УДК 519.854
ББК 22.18

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ

1.1 Основные типы задач дискретной оптимизации

Задача дискретной оптимизации – это задача поиска максимума или минимума функции *f*, определенной на конечном или счетном множестве *D*:

$$f(x) \to extr,\ x \in D. \qquad (1.1)$$

Функция *f* называется целевой функцией, элементы множества *D* – допустимыми решениями. Если множество *D* задается системой ограничений

$$g_i(x) \le 0,\ i = 1,...,m_1,\ g_i(x) = 0,\ i = m_1 + 1,...,m,$$

$$x = (x_1,...,x_n) \in R^n,\ x_j \in \Omega_j \subset R,\ j = 1,...,n_1,\ n_1 \le n,$$

где каждое Ω_j – либо конечное множество, содержащее не менее двух элементов, либо счетное множество, то задача (1.1) называется задачей частично дискретного (или дискретного, если $n_1 = n$) математического программирования. Если $\Omega_j \subseteq Z$, $j = 1,...,n_1$, задача (1.1) называется задачей частично целочисленного (целочисленного при $n_1 = n$) программирования. В пособии будут рассматриваться, в основном, задачи целочисленного линейного программирования, которые формулируются следующим образом:

$$\sum_{j=1}^{n} c_j x_j \to \min$$

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

$$x_j \ge 0,\ j = 1,...,n,$$

$$x_j \in Z,\ j = 1,...,n_1,\ n_1 \le n.$$

Обычно предполагается, что параметры задачи – коэффициенты целевой функции c_j, $j = 1,...,n$, элементы матрицы системы ограничений $A = (a_{ij})$ и вектора правых частей $b = (b_1,...,b_m)$ – целые числа.

Выделяют следующие основные классы задач дискретного программирования: транспортная задача и ее варианты, задачи с неделимостями, экстремальные комбинаторные задачи, задачи на неклассических областях, задачи с разрывной целевой функцией.

Транспортная задача линейного программирования при целочисленных исходных данных всегда обладает целочисленным оптимальным планом. Поэтому для решения дискретных задач, которые можно сформулировать как транспортные задачи, дополненные условиями целочисленности переменных, можно применять обычные методы линейного программирования. Самая известная задача этого класса – задача о назначениях.

Задачи с неделимостями – это математические модели прикладных задач, переменные в которых представляют физически неделимые величины.

Такие задачи описывают, например, планирование выпуска неделимых видов продукции или использования неделимых производственных факторов, это задачи распределения ресурсов и капиталовложений, сетевого планирования и управления, производственного планирования и т.п.

В *комбинаторных задачах* оптимизируется функция, заданная на конечном множестве, элементами которого служат выборки (перестановки) из *n* объектов. Из комбинаторных задач, имеющих большое прикладное значение, следует отметить задачу о коммивояжере и задачи теории расписаний. При постановке комбинаторных задач в виде задач целочисленного математического программирования часто вводятся булевы переменные $x_j \in \{0,1\}$, носящие логический характер: $x_j = 1$, если выполняется некоторое условие и $x_j = 0$ в противном случае. Так, например, обстоит дело, когда решение сводится к выбору одного из возможных вариантов действий, включая и вариант отказа от каких бы то ни было действий.

Задачи на неклассических областях представляют собой задачи нахождения экстремума линейной функции на невыпуклой или несвязной области, задаваемой, например, с использованием логических условий вида «либо-либо». В *задачах с разрывными целевыми функциями*, напротив, допустимое множество – выпуклый многогранник, но целевая функция не является непрерывной. Например, под неоднородной разрывной линейной функцией понимается функция $\sum_{j=1}^{n} c_j(x_j)$, где $c_j(x_j) = \begin{cases} 0 & при \quad x_j = 0 \\ c_j x_j + d_j & при \quad x_j > 0. \end{cases}$

К появлению подобных целевых функций приводит, в частности, учет в моделях постоянных затрат, которые должны быть произведены независимо от объема производства. Задачи последних двух классов могут быть сведены к задачам частично целочисленного линейного программирования введением дополнительных, как правило, булевых переменных.

Приведем примеры постановки некоторых задач дискретной оптимизации.

1.2 Задача о рюкзаке

Задачей о рюкзаке или ранце называется задача целочисленного линейного программирования с одним ограничением:

$$\sum_{j=1}^{n} c_j x_j \to \max \tag{1.2}$$

$$\sum_{j=1}^{n} a_j x_j \le b, \tag{1.3}$$

$$x_j \ge 0 \text{ и целые, } j = 1,...,n. \tag{1.4}$$

Числа c_j, a_j ($j = 1,...,n$), b можно считать положительными и целыми, $a_j \le b$.

Своё название задача получила благодаря следующей её интерпретации. Имеется n видов неделимых предметов со стоимостями c_1, …, c_n и весами a_1, …, a_n. Требуется так упаковать ими рюкзак, чтобы его вес не превышал b, а суммарная стоимость упакованных предметов была максимальной. Вводя переменные x_j, $j = 1,...,n$, для количества упакованных предметов каждого типа, получаем задачу (1.2) – (1.4).

Иногда рассматривается задача о рюкзаке с ограничением типа равенства

$$\sum_{j=1}^{n} a_j x_j = b,$$

что соответствует требованию полной загрузки рюкзака грузоподъёмностью b. Задача о рюкзаке (1.2) – (1.4) может быть сведена к задаче с ограничением типа равенства введением дополнительной неотрицательной целочисленной переменной

$$x_{n+1} = b - \sum_{j=1}^{n} a_j x_j .$$

Если считать, что имеется ровно по одному предмету каждого типа, то ограничение (1.4) следует заменить условием

$$x_j \in \{0,1\},\ j = 1,...,n, \tag{1.5}$$

то есть $x_j = 1$, если j-й предмет кладется в рюкзак и $x_j = 0$, если нет. Полученная задача (1.2), (1.3), (1.5) называется задачей о бинарном или 0-1 рюкзаке. При этом предполагается дополнительно, что $\sum_{j=1}^{n} a_j > b$, то есть все предметы в рюкзак упаковать нельзя. Множество D в этой задаче – множество n-мерных булевых векторов с компонентами 0, 1, удовлетворяющих условию (1.3). Очевидно, что $|D| = 2^n$.

Обобщением задачи о рюкзаке является задача о многомерном рюкзаке

$$\begin{gathered} \sum_{j=1}^{n} c_j x_j \to \max \\ \sum_{j=1}^{n} a_{ij} x_j \le b_i,\ i = 1,...,m, \\ x_j \in \{0,1\},\ j = 1,...,n, \end{gathered} \tag{1.6}$$

имеющая следующую экономическую интерпретацию. Пусть есть n проектов, ожидаемая прибыль от реализации которых составляет c_1, …., c_n. Задан вектор ресурсов $b = (b_1,...,b_m)$, $b_i > 0$, $i = 1,...,m$; количество единиц ресурса типа i, необходимое для реализации проекта с номером j, равно $a_{ij} > 0$. Для любого ресурса i выполнено условие $\sum_{j=1}^{n} a_{ij} > b_i$, то есть реализация всех проектов

невозможна. Требуется выбрать набор проектов с максимальной суммарной прибылью. Переменная x_j, $j = 1,...,n$, принимает значение 1, если проект j реализуется и 0 в противном случае. Множество допустимых решений этой задачи — это множество булевых векторов $x = (x_1,...,x_n)$, удовлетворяющих условиям (1.6).

Задачи о рюкзаке с ограничениями типа неравенства всегда имеют допустимое решение (например, нулевое). Кроме того, допустимое решение можно получить, если отбросить условие целочисленности, для задач о бинарном рюкзаке – заменить условия (1.5) ограничениями

$$0 \le x_j \le 1,\ j = 1,...,n,$$

решить полученную задачу линейного программирования и округлить нецелые значения переменных x_j в меньшую сторону.

1.3 Задача об упаковке

Задачу об упаковке в контейнеры можно сформулировать следующим образом: задано число $Q>0$ и набор n положительных действительных чисел $L = \{w_1,..., w_n\}$, требуется разбить множество L на наименьшее возможное число подмножеств так, чтобы сумма чисел в каждом подмножестве не превосходила Q. Без ограничения общности можно считать, что

$$Q=1,\ w_i \in (0,1] \text{ при } i = 1,...,n \text{ и } \sum_{i=1}^{n} w_i > 1.$$

Задача интерпретируется как проблема упаковки предметов с размерами w_1, …, w_n в единичный контейнер. Она возникает как частный случай задачи маршрутизации, когда целью является минимизация количества используемых транспортных средств, а также во многих других приложениях.

Сформулируем задачу об упаковке как задачу целочисленного линейного программирования. Так как допустимым является решение, при котором в каждый контейнер помещается один предмет, можно считать, что имеется n контейнеров. Введем переменные y_j, $j = 1,...,n$, принимающие значение 1, если j-й контейнер занят, и 0 в противном случае, и x_{ij}, $i, j = 1,...,n$, равные 1, если i-й предмет помещен в j-й контейнер и 0, если нет. Задача тогда примет вид

$$\sum_{j=1}^{n} y_j \to \min$$

$$\sum_{i=1}^{n} w_i x_{ij} \le y_j,\ j = 1,...,n, \qquad (1.7)$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ i = 1,...,n, \qquad (1.8)$$

$$x_{ij} \in \{0,1\},\ y_j \in \{0,1\},\ \ i, j = 1,..., n\,.$$

Условия (1.7) – ограничения на вместимость контейнера, равенства (1.8) гарантируют, что каждый предмет помещен ровно в один контейнер.

1.4 Транспортная задача с фиксированными доплатами

Рассмотрим математическую модель следующей задачи организации перевозок однородного груза. Имеется *m* пунктов отправления (поставщиков) и *n* пунктов назначения (потребителей), предложение *i*-го поставщика составляет $a_i,\ i = 1,..., m$, спрос *j*-го потребителя – $b_j,\ j = 1,..., n$, при этом $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. Затраты на транспортировку *x* единиц груза от поставщика *i* до потребителя *j* составляют

$$c_{ij}(x) = \begin{cases} 0, & x = 0, \\ c_{ij}x + d_{ij} & x > 0. \end{cases}$$

Числа $c_{ij} \geq 0$ интерпретируются как стоимость перевозки единицы груза и определяются, например, расстоянием между пунктами, постоянные затраты $d_{ij} \geq 0$ могут представлять собой оплату аренды транспортных средств или другие издержки, не зависящие от объема перевозок. Требуется организовать перевозку груза от поставщиков потребителям так, чтобы совокупные затраты на транспортировку были минимальны.

Обозначив через x_{ij} количество единиц груза, перевозимого из *i*-го пункта производства в *j*-й пункт потребления, получим следующую задачу математического программирования с неоднородной разрывной линейной целевой функцией:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}(x_{ij}) \to \min \tag{1.9}$$

$$\sum_{j=1}^{n} x_{ij} = a_i\ ,\ i = 1,..., m\,, \tag{1.10}$$

$$\sum_{i=1}^{m} x_{ij} = b_j\ ,\ j = 1,..., n\,, \tag{1.11}$$

$$x_{ij} \geq 0\,,\ i = 1,..., m\,,\ \ j = 1,..., n\,. \tag{1.12}$$

Задача (1.9)–(1.12) называется транспортной задачей с фиксированными доплатами или неоднородной транспортной задачей. Очевидно, если все $d_{ij} = 0,$ то (1.9)–(1.12) – классическая транспортная задача.

Сведем задачу (1.9)–(1.10) к задаче частично целочисленного линейного программирования. Для этого обозначим

$$M_{ij} = \min\{a_i, b_j\},\ i = 1,...,m,\ j = 1,...,n,$$

и введем переменные

$$y_{ij} \in \{0,1\},\ i = 1,...,m,\ j = 1,...,n, \qquad (1.13)$$

принимающие значение 1, если $x_{ij} > 0$ и 0 в противном случае. Тогда задача

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\left(c_{ij}x_{ij} + d_{ij}y_{ij}\right) \to \min$$

при ограничениях (1.10) – (1.13) и дополнительных ограничениях

$$x_{ij} \le M_{ij}y_{ij},\ i = 1,...,m,\ j = 1,...,n,$$

будет эквивалентна исходной.

Аналогично к задаче частично целочисленного программирования можно привести любую задачу математического программирования с неоднородной разрывной линейной целевой функцией, если переменные в задаче ограничены, то есть

$$0 \le x_j \le M_j,\ j = 1,...,n.$$

1.5 Задачи коммивояжера

Классическая задача коммивояжера заключается в следующем. Коммивояжер, находящийся в родном городе, должен посетить несколько других населенных пунктов, побывав в каждом по одному разу, и вернуться обратно так, чтобы общая длина его пути была как можно меньше.

Пусть все пункты (включая начальный) перенумерованы и их общее количество равно n. Расстояние от пункта i до пункта j обозначается c_{ij}. Полагая $c_{ii} = \infty$ (коммивояжеру запрещается оставаться на месте), определяем $n \times n$ матрицу $C = (c_{ij})$. Естественно считать, что $c_{ij} \ge 0$.

Любой маршрут коммивояжера полностью определяется порядком посещенных пунктов и имеет, поэтому, вид $z = (i_1, i_2, ..., i_n, i_1)$. Длина маршрута z - сумма соответствующих элементов матрицы C, то есть

$$l(z) = \sum_{k=1}^{n} c_{i_k, i_{k+1}},$$

где $i_{n+1} = i_1$. Обозначив множество допустимых маршрутов через D, получаем задачу комбинаторной оптимизации

$$l(z) \to \min,\ z \in D.$$

Очевидным практическим приложением задач коммивояжера является планирование маршрутов перевозки грузов. Помимо этого, решение задачи коммивояжера требуется при расчете авиационных линий, оптимизации конвейерного производства, составлении расписаний работы оборудования и т.п. Под «пунктами» могут пониматься, например, этапы производственного процесса или различные операции, выполняемые на одном оборудовании.

Соответственно, c_{ij} может представлять собой не только расстояние, но и время, издержки, другой измеритель, служащий для определения «выгодности» маршрута. В общем случае, c_{ij} – стоимость следования j непосредственно после i.

Пусть $G=(V,A)$ – полный граф со множеством вершин $V=\{1,...,n\}$ и множеством дуг A, длина дуги (i,j) в котором равна c_{ij}. Контур (ориентированный цикл), включающий каждую вершину графа ровно один раз, называется *гамильтоновым контуром*. Таким образом, задача коммивояжера – это задача поиска гамильтонова контура, имеющего минимальную длину.

В силу замкнутости маршрута можно считать, что начальный пункт i_1 задан, поэтому количество допустимых маршрутов равно $(n-1)!$. Если $c_{ij}=c_{ji}$ для всех $i,j=1,...,n$ (матрица C симметричная), граф G является неориентированным, и говорят о гамильтоновых циклах. Поскольку перестановки $(i_2,...,i_n)$ и $(i_n,...,i_2)$ определяют один маршрут, количество различных замкнутых маршрутов составит $(n-1)!/2$.

Сформулируем задачу коммивояжера на языке математического программирования. Пусть $c_{ii}=M$, где M – произвольно большое число. Введем переменные x_{ij}, $i,j=1,...,n$, принимающие значение 1, если дуга (i,j) содержится в оптимальном маршруте, и 0 в противном случае. Тогда суммарная длина маршрута имеет вид

$$\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij}\,. \tag{1.14}$$

Набору переменных $\{x_{ij}\}$ соответствует граф с n вершинами, дуга (i,j) в котором существует в том и только в том случае, если $x_{ij}=1$. Ясно, что для допустимого решения задачи этот граф должен представлять собой цикл, то есть быть связным, и степень каждой вершины должна равняться двум.

Выполнение второго требования гарантируют условия однократного посещения каждого пункта:

$$\sum_{j}x_{ij}=1,\ i=1,...,n, \tag{1.15}$$

$$\sum_{i}x_{ij}=1,\ j=1,...,n \tag{1.16}$$

Один из возможных вариантов обеспечения связности маршрута следующий. Для каждого собственного подмножества $S\subset V$, $\overline{S}=V\setminus S$ потребуем, чтобы

$$\sum_{i\in S,\,j\in\overline{S}}x_{ij}\geq 1,$$

то есть должна существовать по крайней мере одна дуга, идущая из S в $\overline{S}$. Всего таких ограничений столько же, сколько собственных подмножеств V, то есть 2^n-2.

В более краткой формулировке каждой вершине $i = 2,...,n$ ставится в соответствие переменная u_i и требуется выполнение следующих условий:

$$u_i - u_j + nx_{ij} \le n-1,\ i, j = 2,...,n,\ i \ne j. \qquad (1.17)$$

Нетрудно показать (см., например, [3]), что ограничения (1.17) исключают наличие циклов длиной меньше *n* и переменные u_i можно считать целочисленными и неотрицательными. Вспоминая, что

$$x_{ij} \in \{0,1\},\ i, j = 1,...,n, \qquad (1.18)$$

получаем, что задача коммивояжера принимает вид задачи целочисленного линейного программирования, в которой имеется $n^2 + n - 1$ переменных и $n^2 - n + 2$ ограничений:

Минимизировать функцию (1.14) при ограничениях (1.15) – (1.18).

Существует довольно много разновидностей и обобщений задачи коммивояжера [8]. Рассмотрим некоторые из них.

Если критерием оптимальности маршрута коммивояжера является не общая длина пути, а протяженность самого длинного перехода, задача коммивояжера называется *минимаксной* или задачей на узкое место. Требуется найти гамильтонов контур, имеющий минимальную длину наибольшей дуги:

$$\max c_{i_k, i_{k+1}} \to \min,\ (i_k, i_{k+1}) \in z,\ z \in D.$$

Минимаксная задача коммивояжера может быть сформулирована также следующим образом:

$$\max_{x_{ij}=1} c_{ij} x_{ij} \to \min$$

при условиях (1.15) – (1.18).

Снимем теперь ограничение однократного посещения каждой вершины. Задача поиска контура наименьшей длины, включающего каждую вершину графа хотя бы один раз, называется *общей* задачей коммивояжера. Постановка общей задачи коммивояжера удобна, если граф не является полным, то есть запрещен непосредственный переход из некоторых пунктов i в некоторые пункты j. Обычно отсутствующим дугам графа приписывается бесконечный вес: $c_{ij} = \infty$. Гамильтонов контур в графе может не существовать, то есть задача коммивояжера может не иметь допустимых решений. Решение общей задачи коммивояжера существует, если граф сильно-связный (любые две вершины можно соединить ориентированным путем).

В *открытой задаче коммивояжера* возврат в начальный пункт не предполагается: коммивояжер начинает свой путь в некотором городе и посещает каждый оставшийся город в точности один раз. При этом конечные пункты маршрута могут быть произвольными либо фиксированными. Путь в графе, проходящий через каждую вершину ровно один раз, называется *гамильтоновым путем*. Таким образом, открытая задача коммивояжера – это задача поиска гамильтонова пути, имеющего минимальную длину. Любую открытую задачу коммивояжера можно свести к обычной (закрытой) или наоборот [8].

Любое допустимое решение открытой задачи коммивояжера можно отождествить с перестановкой $z=(i_1,i_2,...,i_n)$ чисел 1,…, *n*. Длина маршрута *z* равна $l(z)=\sum_{k=1}^{n-1} c_{i_k,i_{k+1}}$. Количество открытых маршрутов коммивояжера составляет, очевидно, $n!$, если начальный и конечный пункты произвольны, $(n-1)!$ или $(n-2)!$, если фиксированы один или оба концевых пункта соответственно.

1.6 Сетевая интерпретация задач целочисленного программирования

Множество практически важных оптимизационных задач, в том числе дискретных, допускает естественную постановку в терминах графов и сетей. К этому классу задач относятся, в частности, транспортные задачи, дискретные задачи размещения, задачи маршрутизации, задачи сетевого планирования и управления, задачи теории расписаний. Это позволяет применять для решения таких задач специальные сетевые алгоритмы, нередко более эффективные, чем обычные методы математического программирования [8,10,13]. Оказывается, произвольную задачу целочисленного линейного программирования можно свести к задаче поиска кратчайшего пути в некотором ациклическом графе.

Рассмотрим следующую задачу

$$\langle c,x\rangle \to \min, \tag{1.19}$$

$$Ax=b, \tag{1.20}$$

$$x\ge 0,\ x\in Z^n, \tag{1.21}$$

где $c=(c_1,...,c_n)$, $b=(b_1,...,b_m)$, A – матрица с элементами a_{ij}, $i=1,...,m$, $j=1,...,n$, все коэффициенты c_j, b_i, a_{ij} целые.

Построим граф, каждой вершине которого соответствует вектор $v=Ay$ для некоторого *n*-мерного вектора *y* с целыми неотрицательными компонентами. Вершины $v=Ay$ и $v'=Ay'$ соединены дугой длиной c_j, если вектора *y* и *y'* отличаются только *j*-й компонентой, причем $y'_j=y_j+1$, то есть $v'=v+a^j$, где a^j - *j*-й столбец матрицы *A*. Граф содержит, в частности, вершину, которой соответствует нулевой вектор (при нулевом *y*) и, если множество допустимых решений задачи (1.19)–(1.21) непустое, вершину, которой соответствует вектор *b*.

Пусть *p* – ориентированный путь из вершины 0 в вершину *b*. Обозначим через x_j количество дуг длиной c_j в этом пути. Тогда длина этого пути равна $\sum_{j=1}^{n} c_j x_j$ и $b=\sum_{j=1}^{n} x_j a^j = Ax$. С другой стороны, если $x=(x_1,...,x_n)$ – допустимое решение задачи (1.19)–(1.21), в графе *G* имеется путь *p*, проходящий через вершины

$$0,\ a^1,\ 2a^1,\ldots,\ x_1a^1,\ x_1a^1+a^2,\ \ldots,\ x_1a^1+x_2a^2,\ldots,\ b$$

длиной $l(p)=\sum_{j=1}^{n}c_jx_j$. Таким образом, любому допустимому решению задачи (1.19)–(1.21) соответствует путь из вершины 0 в вершину b в графе G, длина которого равна значению целевой функции на этом решении, и наоборот. В частности, кратчайший путь из вершины 0 в вершину b соответствует оптимальному решению задачи (1.19)–(1.21).

Аналогично, задача целочисленного линейного программирования на максимум сводится к задаче поиска самого длинного пути

К сожалению, для многих задач целочисленного программирования количество вершин в построенном графе настолько велико, что алгоритмы поиска кратчайших путей не могут быть эффективно применены, требуется предварительное преобразование графа, позволяющее уменьшить его размер [9].

Пример 1.1. Рассмотрим следующую задачу

$$3x_1+x_2\to\max$$
$$x_1+2x_2\le 5,$$
$$2x_1+x_2\le 5,$$
$$x_1\ge 0,\ x_2\ge 0 \text{ и целые.}$$

Вводя дополнительные неотрицательные целочисленные переменные

$$x_3=5-x_1-2x_2,\ x_4=5-2x_1-x_2,$$

и умножив целевую функцию на –1, получим задачу вида (1.20) – (1.22) с матрицей системы ограничений

$$A=\begin{pmatrix}1 & 2 & 1 & 0\\ 2 & 1 & 0 & 1\end{pmatrix}.$$

Граф G содержит 36 вершин, на рис. 1.1 показан его фрагмент и один из 12 путей длиной 7 из вершины 0 в вершину b=(5,5), соответствующих оптимальному решению (2,1,1,0) задачи.

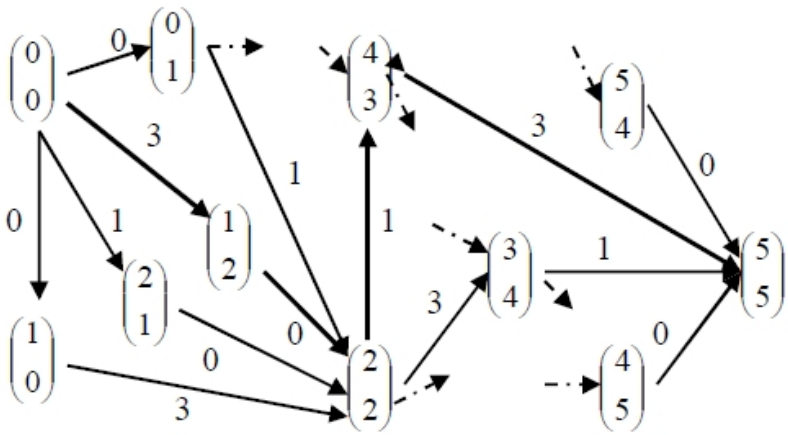

Рис. 1.1. Решение задачи из примера 1.1

Сведем указанным образом к задаче поиска самого длинного пути в ациклическом графе задачу о рюкзаке с ограничением типа равенства. Граф G имеет множество вершин $V=\{0,1,...,b\}$. Вершины i и k соединены дугой длиной c_j, если $k-i=a_j$ для некоторого $j=1,...,n$. Любому допустимому решению $x=(x_1,...,x_n)$ задачи соответствует ориентированный путь из вершины 0 в вершину b и наоборот, если p - ориентированный путь из вершины 0 в вершину b, то обозначая через x_j количество дуг длиной c_j в этом пути, $j=1,...,n$, получим допустимое решение задачи. Так как значение целевой функции на допустимом решении равно длине соответствующего пути, оптимальному решению задачи отвечает самый длинный путь.

Пример 1.2. Рассмотрим задачу

$$5x_1+4x_2+8x_3+2x_4 \to \max$$
$$2x_1+4x_2+3x_3+x_4=7,$$
$$x_1,\ x_2,\ x_3,\ x_4 \geq 0 \text{ и целые.}$$

Граф содержит 8 вершин (0,1,…7), дуги (0,2), (1,3), …, (5,7) длиной 5, (0,4), (1,5), (2,6), (3,7) длиной 4, (0,3), (1,4), …, (4,7) длиной 8, (0,1), (1,2), …, (6,7) длиной 2 (см. табл. 1.1). Самым длинным путям 0–3–6–7, 0–1– 4–7, 0–3–4–7 соответствует решение (0,0,2,1), путям 0–3–5–7, 0–2–5–7, 0–2–4–7 – решение (2,0,1,0). Максимальное значение целевой функции равно 18.

Таблица 1.1

Матрица длин дуг

	1	2	3	4	5	6	7
0	2	5	8	4			
1		2	5	8	4		
2			2	5	8	4	
3				2	5	8	4
4					2	5	8
5						2	5
6							2

Основные подходы к решению задач целочисленного программирования основаны либо на их «регуляризации», то есть замене решения исходной задачи решением последовательности непрерывных задач (методы отсечения), либо на переборе допустимых решений (комбинаторные методы – алгоритмы ветвей и границ, динамическое программирование и т.п.). Кроме того, на практике часто используются эвристические или приближенные методы, позволяющие получить хорошее допустимое решение задачи дискретной оптимизации за приемлемое время.

2 МЕТОД ОТСЕЧЕНИЙ

2.1 Основные понятия линейного программирования

В главе рассматриваются методы решения задач целочисленного линейного программирования, подразумевающие временное отбрасывание условий дискретности и решение соответствующих непрерывных задач. В связи с этим, вспомним сначала основные понятия и методы линейного программирования.

Задача линейного программирования в канонической форме имеет вид

$$\langle c, x \rangle \to \min \tag{2.1}$$

$$Ax = b, \tag{2.2}$$

$$x \geq 0, \tag{2.3}$$

где $A = (a_{ij})$ - матрица размером mxn, $m<n$, $c \in R^n$, $b \in R^m$. Будем считать, что ранг матрицы A равен m. Столбцы матрицы A обозначаем a^j, j=1,...,n.

Допустимым решением или *планом* задачи называется решение $x \in R^n$ системы уравнений (2.2), удовлетворяющее условиям неотрицательности (2.3). Множество допустимых решений

$$X = \{x \in R^n : Ax = b, x \geq 0\}.$$

Решение системы (2.2) $x \in R^n$ называется базисным, если система столбцов a^j матрицы A, соответствующих ненулевым координатам вектора x, линейно независима. Базисное решение системы линейных уравнений (2.2) называется *опорным вектором* задачи линейного программирования (2.1)–(2.3).

Вектор $x \in X$ называется *опорным планом*, если система столбцов a^j матрицы A, соответствующих положительным координатам вектора x, линейно независима. Таким образом, опорный план задачи (2.1)–(2.3) – это опорный вектор, удовлетворяющий условию неотрицательности (2.3), то есть являющийся планом задачи.

Теорема 2.1. *Если задача линейного программирования допустима ($X \neq \varnothing$), то у неё существует опорный план.*

Теорема 2.2. *Если задача линейного программирования имеет решение, то среди её опорных планов найдется оптимальный план.*

Рассмотрим далее методы поиска оптимального опорного плана.

Любая линейно независимая система из m столбцов матрицы A является базисом пространства R^m. *Базисом* опорного вектора x называется базис, включающий все столбцы, соответствующие ненулевым компонентам x. Если количество таких столбцов равно m, опорный вектор называется *невырожденным*.

Пусть x^0 – опорный вектор, $\{a^{j_1},....,a^{j_m}\}$ – его базис. Положим $\sigma = \{j_1,..., j_m\}$ – список номеров базисных переменных, $\omega = \{1,...,n\} \setminus \sigma$ – множество номеров небазисных (свободных) переменных, $A_\sigma = (a^{j_1},...,a^{j_m})$ –

невырожденная *mxm* матрица, составленная из базисных столбцов. Для векторов $x, c \in R^n$ полагаем $x_\sigma = (x_{j_1},...,x_{j_m})$, $c_\sigma = (c_{j_1},...,c_{j_m})$. Аналогично определяются x_ω, c_ω и матрица A_ω.

Поскольку $Ax = \sum_{j=1}^{n} x_j a^j = A_\sigma x_\sigma + A_\omega x_\omega$, вектор x_σ^0 образован коэффициентами разложения вектора *b* по данному базису,

$$x_\sigma^0 = A_\sigma^{-1} b,\ x_j^0 = 0, \tag{2.4}$$

Обратно, пусть множество векторов $\{a^{j_1},....,a^{j_m}\}$ линейно независимо. Очевидно, вектор x^0, удовлетворяющий (2.4) – опорный вектор задачи (2.1)–(2.3). Умножим систему (2.2) на матрицу A_σ^{-1}:

$$A_\sigma^{-1} Ax = A_\sigma^{-1} b. \tag{2.5}$$

Обозначим

$$\Lambda = A_\sigma^{-1} A,\ \Lambda = (\lambda_{ij}),\ i \in \sigma, \tag{2.6}$$

тогда система (2.5) примет вид

$$\Lambda x = x_\sigma^0.$$

Так как $\Lambda x = \sum_{j=1}^{n} x_j \lambda^j = A_\sigma^{-1}(A_\sigma x_\sigma + A_\omega x_\omega) = x_\sigma + A_\sigma^{-1} A_\omega x_\omega$, если $j \in \sigma$, то $\lambda_{ij} = \delta_{ij}$,

$$x_i = x_j^0 - \sum_{j \in \omega} \lambda_{ij} x_j,\ i \in \sigma. \tag{2.7}$$

Используя (2.7), получаем также

$$\langle c, x \rangle = \langle c_\sigma, x_\sigma \rangle + \langle c_\omega, x_\omega \rangle = \sum_{i \in \sigma} c_i x_i^0 - \sum_{i \in \sigma} c_i \sum_{j \in \omega} \lambda_{ij} x_j + \sum_{j \in \omega} c_j x_j.$$

Обозначим

$$\Delta_j = \sum_{i \in \sigma} c_i \lambda_{ij} - c_j = \langle c_\sigma, \lambda^j \rangle - c_j,\ j = 1,..., n,\ \Delta = (\Delta_1,..., \Delta_n). \tag{2.8}$$

Тогда

$$\langle c, x \rangle = \langle c, x^0 \rangle - \langle \Delta, x_\omega \rangle.$$

Очевидно, $\Delta_j = 0$ при $j \in \omega$. Компоненты вектора *Δ* – коэффициенты целевой функции, выраженной через свободные переменные, взятые со знаком минус, называются *приведенными значениями переменных* или *оценками замещения*.

Двойственная к (2.1)–(2.3) задача линейного программирования имеет вид

$$\langle b, y \rangle \to \max \tag{2.9}$$

$$A^T y \le c. \tag{2.10}$$

Справедлива следующая теорема двойственности.

Теорема 2.3. *Если задачи (2.1)–(2.3) и (2.9), (2.10) допустимы, то они имеют оптимальные решения* x^* *и* y^**, причем*

$$\left\langle c, x^* \right\rangle = \left\langle b, y^* \right\rangle. \tag{2.11}$$

Если целевая функция одной из задач (2.1)–(2.3) и (2.9), (2.10) неограниченна на допустимом множестве, то вторая задача не имеет допустимых решений.

Теорема 2.4. *Соотношение двойственности (2.11) эквивалентно следующим условиям (условиям дополняющей нежесткости):*

$$x_j \left(\left(A^T y \right)_j - c_j \right) = 0, \ j = 1,..., n, \tag{2.12}$$

или, что то же самое,

$$x_j \left(\left\langle a^j, y \right\rangle - c_j \right) = 0, \ j = 1,..., n.$$

Если вектор x – опорный и невырожденный, то соотношения (2.12) означают, что y – решение системы $A_\sigma^T y = c_\sigma$, то есть

$$y = \left(A_\sigma^T \right)^{-1} c_\sigma. \tag{2.13}$$

Опорный вектор называется *псевдопланом*, если вектор y, заданный соотношением (2.13), допустим в двойственной задаче: $A^T y - c \le 0$.

Поскольку

$$\Delta = \Lambda^T c_\sigma - c = A^T \left(A_\sigma^T \right)^{-1} A_\sigma^T y - c = A^T y - c,$$

опорный вектор является псевдопланом, если все его оценки замещения неположительны.

Теорема 2.5. *Для того чтобы опорный вектор* x^0 *был оптимальным планом задачи (2.1)–(2.3), необходимо и достаточно, чтобы он являлся псевдопланом и удовлетворял условию неотрицательности.*

Таким образом, оптимальный опорный план задачи (2.1)–(2.3) можно определить либо как псевдоплан, являющийся планом задачи, либо как план задачи, являющийся псевдопланом. ***Симплекс-метод*** решения задачи линейного программирования (2.1)–(2.3) заключается в построении последовательности опорных планов задачи и, соответственно, использовании в качестве критерия оптимальности условия допустимости двойственного вектора. ***Двойственный симплекс-метод*** основан на построении последовательности псевдоплан*ов.* Текущий псевдоплан, удовлетворяющий условию неотрицательности, автоматически является оптимальным опорным планом.

Оба метода включают следующие этапы:

– поиск начального опорного вектора;

– проверка текущего вектора на оптимальность;

– переход к новому опорному вектору.

Всю информацию о текущем опорном векторе x^0 отражает так называемая симплекс-таблица (табл. 2.1).

Таблица 2.1

Симплекс-таблица

	x_1	x_2	...	x_n	
x_{j_1}	$\lambda_{j_1 1}$	$\lambda_{j_1 2}$	...	$\lambda_{j_1 n}$	$x^0_{j_1}$
x_{j_2}	$\lambda_{j_2 1}$	$\lambda_{j_2 2}$	...	$\lambda_{j_2 n}$	$x^0_{j_2}$
...	...	...	...	...	...
x_{j_m}	$\lambda_{j_m 1}$	$\lambda_{j_m 2}$	...	$\lambda_{j_m n}$	$x^0_{j_m}$
	Δ_1	Δ_2	...	Δ_n	$\langle c, x^0 \rangle$

Первоначальное заполнение таблицы может осуществляться по формулам (2.6), (2.8). Так как, согласно (2.4), компоненты опорного вектора определяются выбором базиса, переход к новому опорному вектору осуществляется путем перехода к новому базису. Пусть, например, $\sigma' = \sigma \cup \{k\} \setminus \{s\}$. Тогда $A_{\sigma'} = A_\sigma T$, где T – матрица перехода, получающаяся заменой в единичной матрице столбца, стоящего на том же месте, что столбец a^s в матрице A_σ, на столбец $A_\sigma^{-1} a^k$. Следовательно, $\Lambda = A_{\sigma'}^{-1} A = (A_\sigma T)^{-1} A = T^{-1}\Lambda$, $x^0_{\sigma'} = A_{\sigma'}^{-1} b = T^{-1} x^0_\sigma$, то есть

$$\lambda'_{ij} = \lambda_{ij} - \lambda_{sj}\frac{\lambda_{ik}}{\lambda_{sk}},\ i \in \sigma \setminus \{s\},\ \lambda'_{kj} = \frac{\lambda_{sj}}{\lambda_{sk}},\ j = 1,\ldots,n, \tag{2.14}$$

$$x'^0_i = x^0_i - x^0_s\frac{\lambda_{ik}}{\lambda_{sk}},\ i \in \sigma \setminus \{s\},\ x'^0_k = x^0_s\frac{1}{\lambda_{sk}}, \tag{2.15}$$

$$\Delta'_j = \Delta_j - \lambda_{sj}\frac{\Delta_k}{\lambda_{sk}},\ \langle c, x'^0 \rangle = \langle c, x^0 \rangle - x^0_s\frac{\Delta_k}{\lambda_{sk}}. \tag{2.16}$$

Столбец k таблицы называется ведущим столбцом, строка s – ведущей строкой, элемент λ_{sk} – ведущим элементом. Из формул (2.14)–(2.16) вытекает, что для преобразования симплекс-таблицы осуществляются элементарные преобразования строк матрицы

$$\begin{pmatrix} \Lambda & x^0_\sigma \\ \Delta' & \langle c, x^0 \rangle \end{pmatrix},$$

в результате которых ведущий элемент становится равным 1, а все остальные элементы ведущего столбца – равными нулю.

Рассмотрим подробнее итерации двух указанных методов.

Симплекс-метод

Текущий опорный вектор x^0 допустим, то есть $x^0 \geq 0$.

Если $\Delta_j \leq 0$, $j \notin \sigma$, то опорный план оптимален. В противном случае, выберем такое $k \notin \sigma$, что $\Delta_k > 0$. Например,

$$\Delta_k = \max\{\Delta_j : \Delta_j > 0\}.$$

Лемма 2.1. *Если $\lambda_{ik} \leq 0$ при всех $i \in \sigma$, то целевая функция задачи (2.1)–(2.3) не ограничена на допустимом множестве.*

Если ведущий столбец содержит положительные элементы, можно перейти к новому опорному плану. Переменная x_k вводится в базис. При этом не должна нарушаться допустимость опорного вектора, то есть, если из базиса выводится переменная x_s, то, согласно (2.15),

$$\lambda_{sk} > 0,\ x_i^0 - x_s^0 \frac{\lambda_{ik}}{\lambda_{sk}} \geq 0,\ i \in \sigma \setminus \{s\}.$$

Если $\lambda_{ik} \leq 0$, условие заведомо выполнено. Поэтому s определяется из соотношения

$$\frac{x_s^0}{\lambda_{sk}} = \min\left\{\frac{x_i^0}{\lambda_{ik}}, \lambda_{ik} > 0\right\}. \qquad (2.17)$$

Двойственный симплекс-метод

Текущие опорные вектора удовлетворяют условию оптимальности, то есть $\Delta_j \leq 0$, $j \notin \sigma$. Если $x_\sigma^0 \geq 0$, то текущий вектор – оптимальный опорный план. В противном случае выберем $s \in \sigma$ такое, что $x_s^0 < 0$, например,

$$s = \min\{i \in \sigma : x_i^0 < 0\}.$$

Лемма 2.2. *Если все* $\lambda_{sj} \geq 0$, $j = 1,..., n$, *то задача не имеет решения – множество X её планов пусто.*

Предположим, условия леммы не выполнены. Переменная x_s выводится из базиса. Сформулируем правило выбора номера k вводимой в базис переменной. Новый опорный вектор должен оставаться псевдопланом, то есть должны выполняться условия $\Delta'_j \leq 0$, $j = 1, 2, ..., n$. Из (2.16) получаем, что

$$\Delta_j - \lambda_{sj} \frac{\Delta_k}{\lambda_{sk}} \leq 0 \text{ для } j=1,\ldots,n. \qquad (2.18)$$

Если $j \in \sigma$ то $\lambda_{sj} = \delta_{sj} \geq 0$. Поэтому, чтобы в базис вводились лишь векторы, в настоящий момент в нем не занятые, введем дополнительное к (2.18) условие $\lambda_{sk} < 0$. Если $\lambda_{sj} \geq 0$, то, поскольку $\Delta_j \leq 0$ и $\Delta_k \leq 0$, неравенство (2.18), очевидно, выполнено. Если $\lambda_{sj} < 0$, то (2.18) эквивалентно неравенству

$$\frac{\Delta_k}{\lambda_{sk}} \leq \frac{\Delta_j}{\lambda_{sj}}.$$

Таким образом, в базис вводится переменная x_k такая, что

$$\frac{\Delta_k}{\lambda_{sk}} = \min\left\{\frac{\Delta_j}{\lambda_{sj}}, \lambda_{sj} < 0\right\}. \qquad (2.19)$$

Пример 2.1. Рассмотрим задачу линейного программирования

$$12x_1 + 7x_2 \to \max$$
$$x_1 + 2x_2 \leq 10$$
$$x_1 + 2x_2 \geq 2$$
$$2x_1 + x_2 \leq 10$$

$$x_1 \ge 0,\ x_2 \ge 0.$$

Перейдем к задаче на минимум, умножив на –1 целевую функцию, и введем дополнительные неотрицательные переменные x_3, x_4, x_5, превращающие ограничения задачи в равенства:

$$x_1 + 2x_2 + x_3 = 10$$
$$x_1 + 2x_2 - x_4 = 2$$
$$2x_1 + x_2 + x_5 = 10$$

Получим задачу вида (2.1)–(2.3), где матрица системы ограничений имеет вид

$$A = \begin{pmatrix} 1 & 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & -1 & 0 \\ 2 & 1 & 0 & 0 & 1 \end{pmatrix},$$

$c = (-7,-12,0,0,0),\ b = (10,2,10).$

Задача имеет 9 опорных векторов: $x^1 = (6,-2,8,0,0),\ x^2 = (10/3,10/3,0,8,0),$ $x^3 = (5,0,5,3,0),\quad x^4 = (2,0,8,0,6),\quad x^5 = (10,0,0,8,-10),\quad x^6 = (0,10,-10,18,0),$ $x^7 = (0,1,8,0,9),\ x^8 = (0,5,0,8,5),\ x^9 = (0,0,10,-2,10)$. Опорными планами задачи являются x^2, x^3, x^4, x^7, x^8, причем x^2 – оптимальный, вектор x^5, которому соответствует наименьшее значение целевой функции преобразованной задачи, равное –120, – псевдоплан.

Для иллюстрации применения симплекс-метода возьмём в качестве начального вектор x^3, $\sigma = \{1,3,4\}$,

$$A_\sigma = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 2 & 0 & 0 \end{pmatrix},\ \Lambda = A_\sigma^{-1} A = \begin{pmatrix} 1 & 0,5 & 0 & 0 & 0,5 \\ 0 & 1,5 & 1 & 0 & -0,5 \\ 0 & -1,5 & 0 & 1 & -1,5 \end{pmatrix}.$$

Составим симплекс-таблицу (табл. 2.2).

Таблица 2.2

Начальная симплекс-таблица

	x_1	x_2	x_3	x_4	x_5	
x_1	1	0,5	0	0	0,5	5
x_3	0	1,5	1	0	–0,5	5
x_4	0	–1,5	0	1	0,5	3
Δ	0	1	0	0	–6	–60

В базис вводится переменная x_2, выводится x_3. После преобразования получаем таблицу, соответствующую оптимальному плану (табл. 2.3). Следовательно, получено оптимальное решение исходной задачи $x_1 = x_2 = 10/3$, максимальное значение целевой функции равно 190/3.

Таблица 2.3

Итоговая симплекс-таблица

	x_1	x_2	x_3	x_4	x_5	
x_1	1	0	–1/3	0	2/3	10/3
x_2	0	1	2/3	0	–1/3	10/3
x_4	0	0	1	1	0	8
Δ	0	0	–2/3	0	–17/3	–190/3

Для применения двойственного симплекс-метода в качестве начального вектора берем псевдоплан x^5, $\sigma = \{1,4,5\}$,

$$A_\sigma = \begin{pmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 2 & 0 & 1 \end{pmatrix}, \ \Lambda = A_\sigma^{-1} A = \begin{pmatrix} 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & -3 & -2 & 0 & 1 \end{pmatrix}.$$

Составляем симплекс-таблицу (табл. 2.4).

Таблица 2.4

Начальная симплекс-таблица

	x_1	x_2	x_3	x_4	x_5	
x_1	1	2	1	0	0	10
x_4	0	0	1	1	0	8
x_5	0	–3	–2	0	1	–10
Δ	0	–17	–12	0	0	–120

Из базиса выводится x_5, вводится x_2 (17/3<12/2). Преобразуя таблицу, получаем оптимальный опорный план (табл. 2.3).

Замечание 2.1. Если задача линейного программирования невырожденная, то есть опорные вектора содержат m положительных компонент, на каждом этапе алгоритма симплекс-метода значение целевой функции уменьшается, и, следовательно, через конечное число шагов либо будет найден оптимальный опорный план, либо установлено, что оптимального плана не существует. Невырожденность двойственной задачи означает, что ровно m ограничений (2.10) могут быть выполнены как равенства, следовательно, на каждом шаге двойственного алгоритма все оценки замещения, соответствующие небазисным переменным, отрицательны и при переходе к новому базису значение целевой функции увеличивается. В вырожденном случае, для того, чтобы сделать невозможным зацикливание и обеспечить, тем самым, сходимость алгоритмов, применяются специальные методы, усиливающие требования к рассматриваемым опорным векторам.

Рассмотрим подход, основанный на понятии лексикографической упорядоченности в пространстве R^{n+1}. Говорят, что вектор y лексикографически больше нулевого вектора ($y \succ 0$), если $y \neq 0$ и первая его ненулевая координата положительна. Вектор y лексикографически больше вектора y' ($y \succ y'$), если $y - y' \succ 0$. Отношение $\succ$ является строгим линейным порядком.

Опорный план x^0 называется *строго допустимым*, если все вектора $\alpha^i = \left(x_i^0, \lambda_{i1}, ..., \lambda_{in}\right)$, $i \in \sigma$, лексикографически больше нуля: $\alpha^i \succ 0$, $i \in \sigma$. Очевидно, что если план x^0 невырожден, то он строго допустим.

Пусть на некоторой итерации симплекс-метода текущий опорный план x^0 строго допустим, вектор a^k решено ввести в базис. Тогда вектор a^s, выводимый из базиса, определяется так, чтобы

$$\frac{1}{\lambda_{sk}}\alpha^s \prec \frac{1}{\lambda_{ik}}\alpha^i,\ s,i \in \sigma,\ \lambda_{sk} > 0,\ \lambda_{ik} > 0. \qquad (2.20)$$

По определению отношения порядка, $\frac{x_s^0}{\lambda_{sk}} \le \frac{x_i^0}{\lambda_{ik}}$, $s,i \in \sigma$, $\lambda_{sk} > 0$, $\lambda_{ik} > 0$, то есть выполнено (2.17), но теперь вектор a^s, выводимый из базиса, определяется однозначно. Действительно, предположим $1/\lambda_{sk}\alpha^s = 1/\lambda_{ik}\alpha^i$, тогда вектора α^s и α^i линейно зависимы, а значит, линейно зависимы строки матрицы Λ, ранг которой равен рангу матрицы A.

Обозначим $\alpha^0 = \left(\left\langle c, x^0 \right\rangle, \Delta\right)$.

Лемма 2.3. *Новый опорный план x'^0, полученный с использованием правила (2.20), строго допустим и $\alpha^0 \succ \alpha'^0$.*

Из леммы следует, что если начать вычисления со строго допустимого опорного плана и руководствоваться правилом (2.20), то зацикливание невозможно – вектора α^0, однозначно восстанавливаемые по плану x^0, строго лексикографически убывают и поэтому повторяться.

Аналогичные построения возможно осуществить и для двойственного симплекс-метода. Пусть x^0 – текущий псевдоплан, обозначим

$$\beta^j = \left(-\Delta_j, \lambda_{1j}, \lambda_{2j}, ..., \lambda_{nj}\right),\ j \in \omega,\ \beta^0 = \left(-\left\langle c, x^0 \right\rangle, x_1^0, ..., x_n^0\right),$$

где для $i \in \omega$ $\lambda_{ij} = -\delta_{ij}$, то есть $\lambda_{ij} = -1$, если $i=j$ и 0 в противном случае. Следовательно, матрица размера (n-1)x(n-m), составленная из векторов β^j, содержит умноженную на (–1) единичную подматрицу порядка n-m и векторы β^j линейно независимы.

Псевдоплан x^0 называется *лексикографически нормальным*, если $\beta^j \succ 0$ для всех $j \in \omega$.

Предположим, вектор x^0 лексикографически нормальный, вектор a^s выводится из базиса. Тогда вектор a^k, который будет введен в базис, определим так, чтобы

$$\frac{1}{\lambda_{sk}}\beta^k \succ \frac{1}{\lambda_{sj}}\beta^j,\ k,j \in \omega,\ \lambda_{sk} < 0,\ \lambda_{sj} < 0. \qquad (2.21)$$

Очевидно, что в этом случае k находится однозначно и остается справедливым соотношение (2.19).

Роль леммы 2.3, из которой вытекает невозможность зацикливания, выполняет в данном случае следующее утверждение.

Лемма 2.4. *Новый псевдоплан* x'^0, *полученный с использованием правила (2.21), лексикографически нормальный и* $\beta^0 \succ \beta'^0$.

Замечание 2.2. Несмотря на очевидное сходство описываемых вычислительных процедур, каждая из них имеет свою область применения, в которой использование данного метода наиболее целесообразно. Так, двойственный симплекс-метод удобен тем, что его можно применять в том случае, когда решается не одна, но несколько задач линейного программирования с возрастающим количеством дополнительных ограничений.

Пусть *x** — оптимальный опорный план задачи (2.1)–(2.3). Предположим, что к ограничениям этой задачи добавлено еще одно ограничение типа неравенства:

$$\langle a_{m+1}, x\rangle \geq b_{m+1}, \tag{2.22}$$

где $a_{m+1} = (a_{m+1,1},...,a_{m+1,n})$, $b_{m+1} \in R^1$.

Поскольку добавление нового ограничения может привести только к сужению множества планов, если *x** удовлетворяет неравенству (2.21), то *x** является оптимальным планом новой задачи. В противном случае введем дополнительную неотрицательную переменную x_{n+1}:

$$\langle a_{m+1}, x\rangle - x_{n+1} = b_{m+1},\ x_{n+1} \geq 0.$$

Матрица системы ограничений задачи имеет вид

$$\begin{pmatrix} A & 0 \\ a_{m+1} & -1 \end{pmatrix},$$

поэтому вектор $x^0 = (x^*, \langle a_{m+1}, x\rangle - b_{m+1})$ является опорным, $\sigma = \{j_1,...,j_m, n+1\}$, где $\{a^{i_1},...,a^{i_m}\}$ - базис вектора *x**, $\omega = \{1,...,n+1\} \setminus \sigma = \{1,...,n\} \setminus \{j_1,...,j_m\}$. Элементы строк j_1, …, j_m матрицы Λ равны, ввиду (2.7), соответствующим элементам матрицы, отвечающей вектору *x**. Выразим, пользуясь (2.7), x_{n+1} через небазисные переменные:

$$x_{n+1} = \langle a_{m+1}, x\rangle - b_{m+1} = \sum_{i\in\sigma} a_{m+1,i}\left(x_i^* - \sum_{j\in\omega}\lambda_{ij}x_j\right) + \sum_{j\in\omega} a_{m+1,j}x_j - b_{m+1} =$$

$$= \sum_{i\in\sigma} a_{m+1,i}x_i^0 - b_{m+1} - \sum_{j\in\omega}\left(\sum_{i\in\sigma} a_{m+1,i}\lambda_{ij} - a_{m+1,j}\right)x_j.$$

Следовательно,

$$\lambda_{n+1,j} = \sum_{i\in\sigma} a_{m+1,i}\lambda_{ij} - a_{m+1,j},\ j \in \omega.$$

Поскольку, согласно (2.8), оценки замещения свободных переменных не изменяются при переходе к задаче с дополнительным условием, и так как

вектор x^* был оптимальным в исходной задаче, вектор x^0 является псевдопланом новой задачи.

Таким образом, чтобы найти решение задачи с дополнительным условием, можно использовать двойственный симплекс-метод, взяв x^0 в качестве начального псевдоплана, матрица Λ (и, следовательно, симплекс-таблица) получается присоединением строки и столбца, отвечающих переменной x_{n+1}, к матрице, соответствующей вектору x^*.

Замечание 2.3. Предположим, рассматривается задача максимизации линейной функции при условиях (2.2), (2.3), то есть

$$\langle c, x \rangle \to \max,\ x \in X .$$

Для того, чтобы перейти от задачи на максимум к эквивалентной задаче на минимум, достаточно умножить на –1 коэффициенты целевой функции. С другой стороны, так как эти коэффициенты используются только для определения оценок замещения, все утверждения параграфа можно переформулировать непосредственно для задачи на максимум, заменив в них Δ_j на $-\Delta_j$. Например, текущий опорный вектор будет оптимальным тогда и только тогда, когда все оценки замещения неотрицательны.

2.2 Общая идея методов отсечения

Пусть $X \subset R^n$ – выпуклый многогранник, например, задаваемый условиями (2.2), (2.3). Рассмотрим задачу целочисленного линейного программирования

$$\langle c, x \rangle \to \min \tag{2.23}$$

$$x = (x_1, ..., x_n) \in X , \tag{2.24}$$

$$x_i \in Z ,\ i=1,\ldots,n. \tag{2.25}$$

Пусть $X^c = \{x \in X : x_i \in Z, i = 1,...,n\}$ – множество целых точек из X. Таким образом, задача может быть записана в виде

$$\langle c, x \rangle \to \min,\ x \in X^c . \tag{2.26}$$

Под регуляризацией задачи (2.26) понимается поиск такого выпуклого многогранника $X' \subset R^n$, чтобы решение (2.26) сводилось к решению задачи линейного программирования

$$\langle c, x \rangle \to \min,\ x \in X' . \tag{2.27}$$

Если все крайние точки множества X целочисленные, то оптимальный опорный план задачи (2.23), (2.24) автоматически будет решением задачи (2.26), то есть $X' = X$. Так обстоит дело, например, с задачей о назначениях – задачей комбинаторной оптимизации, решение которой сводится к решению транспортной задачи линейного программирования [3].

В общем случае простого отбрасывания условия дискретности (2.25) недостаточно, поскольку оптимальный опорный план $x' \in X$ задачи линейного

программирования (2.23), (2.24) может быть нецелочисленным. Иногда в этой ситуации удается получить допустимое решение задачи (2.26), например, округляя компоненты вектора x', но при этом нет гарантий, что это решение будет оптимальным.

Обозначим через Y выпуклую оболочку множества X^c. По определению, $X^c \subseteq Y^c$. Так как множество X выпукло, $Y = co\left(X^c\right) \subset co(X) = X$ и $Y^c \subseteq X^c$. Следовательно, $Y^c = X^c$, задача (2.26) совпадает с задачей

$$\langle c, x\rangle \to \min,\ x = (x_1,...,x_n) \in Y,\ x_i \in Z,\ i=1,\ldots,n. \qquad (2.28)$$

Поскольку множество X^c конечно, Y – выпуклый многогранник и все его крайние точки, то есть опорные планы задачи (2.27) целочисленные. Таким образом, оптимальный опорный план задачи (2.27), где $X' = Y$, является оптимальным решением задачи (2.28) и, соответственно, задачи (2.26).

Можно показать, что $X' = co\left(X^c\right)$ – единственный целочисленный многогранник, такой, что

$$X'^c = X^c. \qquad (2.29)$$

Кроме того, $X' = co\left(X^c\right)$ – единственный многогранник, удовлетворяющий условию (2.29) и такой, что при любом векторе c все оптимальные опорные планы задачи (2.27) целочисленные, и оптимальные значения целевых функций для задач (2.26) и (2.27) совпадают.

Разумеется, для построения задачи (2.26), аппроксимирующей задачу (2.25), достаточно требовать, чтобы последние условия, наложенные на многогранник X' (целочисленность оптимальных опорных планов и совпадение оптимальных значений целевых функций) выполнялись для заданного вектора c. Это позволяет использовать информацию о целевой функции, но искомый многогранник будет определяться, вообще говоря, не единственным образом.

Пример 2.2. Рассмотрим задачу целочисленного линейного программирования, полученную добавлением условий целочисленности к задаче из примера 2.1:

$$12x_1 + 7x_2 \to \max$$
$$x_1 + 2x_2 \le 10$$
$$x_1 + 2x_2 \ge 2$$
$$2x_1 + x_2 \le 10$$
$$x_1 \ge 0,\ x_2 \ge 0.$$
$$x_1,\ x_2 \text{ - целые.}$$

Оптимальное решение задачи линейного программирования – точка $x^0 = (10/3, 10/3)$. Значение целевой функции в точке (3,3), полученной округлением компонент точки x^0, равно 57, что меньше, чем 60 - значение целевой функции в допустимой целочисленной точке (5,0).

Выпуклая оболочка множества допустимых целых точек (многогранник Y) задается ограничениями

$$x_1 + 2x_2 \geq 2,$$
$$2x_1 + x_2 \leq 10,$$
$$x_1 + 2x_2 \leq 10,$$
$$x_1 + x_2 \leq 6,$$
$$x_1 \geq 0,\ x_2 \geq 0.$$

Решение соответствующей задачи (2.27) и, следовательно, целочисленной задачи – точка (4,2). Максимальное значение целевой функции – 62.

Обозначим через Z множество, задаваемое ограничениями

$$12x_1 + 7x_2 \leq 62,$$
$$x_1 \geq 0,\ x_2 \geq 0.$$

Тогда линейной аппроксимацией в указанном смысле для задачи целочисленного программирования будет задача (2.27), где X' – любой такой многогранник, что $Y \subseteq X' \subset Z$, точка (4,2) – угловая и $X'^{c} = Y^{c}$ (рис.2.1).

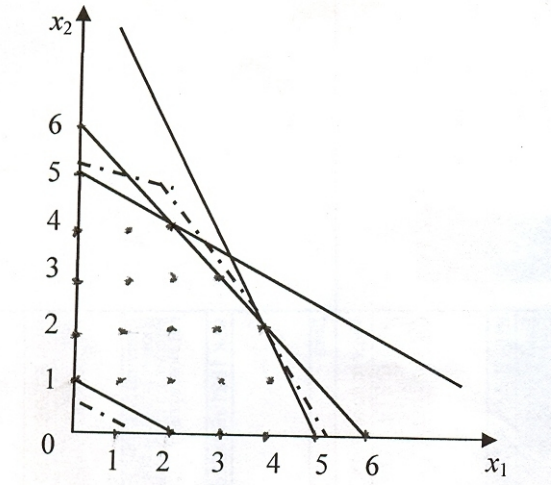

Рис. 2.1. Допустимое множество и его аппроксимации

В общем случае задача построения аппроксимирующей задачи, удовлетворяющей указанным условиям, (например, поиск выпуклой оболочки множества целочисленных точек заданного многогранника), сопоставима по сложностью с исходной задачей целочисленного программирования.

Требования к аппроксимирующей задаче можно ослабить следующим образом. Во-первых, обычно достаточно найти одно оптимальное решение задачи целочисленного программирования, поэтому можно отказаться от условия, что все оптимальные опорные планы аппроксимирующей задачи удовлетворяют условию целочисленности. Во-вторых, только в некоторых угловых точках многогранника X' целевая функция принимает наилучшее значение, поэтому большинство ограничений, задающих этот многогранник, не участвуют в формировании оптимального решения. Вместо того, чтобы сначала полностью определить аппроксимирующую задачу, а затем решать её, можно строить многогранник X' поэтапно, соблюдая условие (2.29), и на каждом этапе исследовать возможность получения оптимального решения исходной задачи.

Указанный подход реализован в ***методах отсечения***, которые заключаются в следующем.

Решается задача линейного программирования (2.23), (2.24). Если эта задача неразрешима, то не имеет решений и исходная задача.

Если оптимальный план x^0 задачи линейного программирования удовлетворяет условиям целочисленности, то он является оптимальным и для исходной задачи. В противном случае к задаче добавляется такое новое линейное ограничение

$$\sum_j a_j x_j \le b\,, \text{ (или в векторной записи } \langle a, x\rangle \le b\,) \tag{2.30}$$

что, во-первых, найденный нецелочисленный план ему не удовлетворяет, то есть

$$\langle a, x^0\rangle > b\,, \tag{2.31}$$

во-вторых, не исключается ни один целочисленный план, то есть

$$X^c \subset \{x : \langle a, x\rangle \le b\}. \tag{2.32}$$

Первое условие называется условием отсечения, второе – условием правильности, а удовлетворяющее этим условиям ограничение – *правильным отсечением* (или просто отсечением).

Геометрически добавление правильного отсечения отвечает проведению гиперплоскости, которая отсекает от текущего многогранника некоторую его часть вместе с оптимальной точкой с нецелыми координатами, но не затрагивает ни одной из целых точек этого многогранника.

Далее задача решается с учетом нового ограничения. В соответствии с замечанием 2.2, для этого можно использовать алгоритм двойственного симплекс-метода, начиная решение с псевдоплана, построенного на основании текущего вектора x^0. Если оптимальное решение этой новой задачи является целочисленным, то оно является оптимальным решением исходной задачи. В противном случае добавляется новое отсечение и так далее. Возможно добавление нескольких отсечений на каждом шаге.

Если на каждом этапе отсечения выбираются корректно, то начальный многогранник X будет постепенно уменьшаться, пока в окрестности оптимального решения не станет совпадать с выпуклой оболочкой целочисленных точек и решение текущей задачи линейного программирования не окажется целочисленным. Поскольку условие (2.29) выполняется для всех вспомогательных задач, целочисленный оптимальный план вспомогательной задачи автоматически оказывается оптимальным планом исходной задачи.

Так как существует бесконечно много ограничений, удовлетворяющих условиям (2.31), (2.32), для построения алгоритма метода отсечений необходимо указать способ выбора правильных отсечений, обеспечивающий конечность процесса решения.

Пример 2.3. В задаче из примера 2.2 правильным отсечением, устраняющим точку (10/3,10/3) является любое ограничение вида

$$x_1 + \alpha x_2 \le (\alpha + 1)\beta\,,$$

где $3 \le \beta < 10/3$, $\dfrac{4-\beta}{\beta-2} \le \alpha \le \dfrac{\beta-2}{4-\beta}$.

Если на шаге k (k=1,2,…) выбирать отсечение указанного типа, в котором

$$\beta_k = \beta_{k-1} - 10^{-k},\ \beta_0 = 10/3,\ \alpha_k = \frac{\beta_k - 2}{4-\beta_k},$$

то минимум в k-й задаче будет достигаться в точке $x^k = \left(x_1^k, x_2^k\right)$,

$$x_1^k = \frac{(10-\beta_k)\alpha_k - \beta_k}{2\alpha_k - 1},\ x_2^k = 2\frac{(\alpha_k+1)\beta_k - 5}{2\alpha_k - 1}.$$

Легко проверить, что $x^k \to (52/15, 46/15)$ при $k \to \infty$, то есть алгоритм не будет сходиться к целочисленному решению.

Если же выбрать в качестве сечения неравенство $x_1 + x_2 \le 6$ (β=3, α=1), которое вместе с ограничениями задачи определяет выпуклую оболочку целочисленных точек многогранника X, то оптимальное решение (4,2) расширенной задачи является оптимальным решением исходной задачи.

2.3 Алгоритм Гомори

Исторически первая реализация метода отсечений была предложена американским математиком Р. Гомори в 1958 году, позднее была доказана конечность алгоритма и рассмотрены некоторые его обощения.

Рассматривается полностью целочисленная задача линейного программирования:

$$\sum_{j=1}^{n} c_j x_j \to \min \tag{2.33}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i,\ i = 1,...,m, \tag{2.34}$$

$$x_j \ge 0,\ j = 1,...,n, \tag{2.35}$$

$$x_j \text{ - целое},\ j = 1,...,n. \tag{2.36}$$

Предполагаем, что элементы a_{ij} матрицы A и компоненты b_i вектора b – целые числа.

Пусть x^0 – оптимальный опорный план задачи линейного программирования (2.33)-(2.35), не являющийся целочисленным, A_σ – оптимальная базисная матрица, целевая функция и основные переменные выражены через неосновные x_j ($j \in \omega$) переменные оптимального решения:

$$x_i = x_i^0 - \sum_{j\in\omega} \lambda_{ij} x_j,\ i \in \sigma, \tag{2.37}$$

$$\langle c, x \rangle = \langle c, x^0 \rangle - \sum_{j\in\omega} \Delta_j x_j. \tag{2.38}$$

По предположению, определитель матрицы A_σ и элементы её присоединенной матрицы $\widetilde{A}_\sigma$ – целые числа. Так как $A_\sigma^{-1} = \frac{1}{\det(A_\sigma)}\widetilde{A}_\sigma$, равенства (2.37) можно записать в виде

$$x_i = \frac{\beta_i}{D} - \sum_{j\in\omega}\frac{\alpha_{ij}}{D}x_j,\ i\in\sigma, \tag{2.39}$$

где $D = |\det(A_\sigma)|$, коэффициенты $\beta_i = x_i^0 D$, $\alpha_{ij} = D\lambda_{ij}$ – целые.

Пусть x_i^0 – дробная компонента вектора x^0. Используя (2.39), запишем условие, что переменная x_i должна быть целой, в следующем виде:

$$\sum_{j\in\omega}\alpha_{ij}x_j = \beta_i(\mathrm{mod}\, D) \tag{2.40}$$

Эквивалентное соотношение можно получить, умножая обе части равенства (2.40) на одно и то же число μ, взаимно простое с D:

$$\sum_{j\in\omega}\mu\alpha_{ij}x_j = \mu\beta_i(\mathrm{mod}\, D).$$

Для произвольного целого y обозначим через $|y|_D$ остаток от деления y на D. Полагая $f_j = |\mu\alpha_{ij}|_D$ для $j\in\omega$ и $f_0 = |\mu\beta_i|_D$, получаем, что существует такое целое s, что

$$\sum_{j\in\omega}f_j x_j = f_0 + sD.$$

Если s отрицательно, то $f_0 + sD < 0$, что противоречит тому факту, что $f_j \ge 0$ и $x_j \ge 0$. Следовательно, s – неотрицательное целое, и для любого решения, в котором переменная x_i является целой, справедливо неравенство

$$\sum_{j\in\omega}\frac{f_j}{D}x_j \ge \frac{f_0}{D}. \tag{2.41}$$

С другой стороны, при $x = x^0$ это неравенство не выполняется, так как в этой точке $x_j = 0$ для всех $j\in\omega$.

Таким образом, неравенство (2.41) определяет сечение.

Предположим, что оптимальное значение целевой функции задачи тоже должно быть целым числом (например, все коэффициенты c_j – целые). Тогда, в случае нецелочисленности $\langle c, x^0\rangle$, можно построить сечение вида (2.41), основываясь на равенстве (2.38), так как $\langle c, x^0\rangle = \langle c_\sigma, A_\sigma^{-1}b\rangle$ и числа Δ_j являются дробными со знаменателем D. При этом, в соответствии с алгоритмом двойственного симплекс-метода, при выводе из базиса дополнительной переменной $x_{n+1} = \sum_{j\in\omega}\frac{f_j}{D}x_j - \frac{f_0}{D}$ в основные переходит переменная, для

которой заведомо $\Delta_j \neq 0$ и, следовательно, значение целевой функции в задаче на минимум строго возрастает.

В классическом алгоритме Гомори рассматривались сечения вида

$$\sum_{j\in\omega}\{\lambda_{ij}\}x_j \geq \{x_i^0\}, \qquad (2.42)$$

где через $[v]$ и $\{v\}$ обозначаются соответственно целая и дробная части числа *v*. Неравенство (2.42) – это сечение (2.41) при $\mu = 1$, так как в этом случае

$$f_j = |D\lambda_{ij}|_D = |D[\lambda_{ij}] + D\{\lambda_{ij}\}|_D = D\{\lambda_{ij}\},\ f_0 = |Dx_i^0|_D = D\{x_i^0\}.$$

Вообще говоря, можно пытаться выбрать значение μ таким образом, чтобы неравенство (2.41) отсекало как можно большую часть текущего многогранника. Так как расстояние от допустимой целочисленной точки *x* до гиперплоскости $\sum_{j\in\omega}\frac{f_j}{D}x_j = \frac{f_0}{D}$ равно $\left(\sum_{j\in\omega} f_j x_j - f_0\right)\left(\sum_{j\in\omega} f_j^2\right)^{-1/2}$, одним из способов этого добиться является выбор μ, при котором f_0 принимает наибольшее возможное значение, равное $D - \delta$, где $\delta = н.о.д.(D, \beta_i)$. Такое μ определяется из условий

$$н.о.д.(\mu, D) = 1,\ \mu\beta_i + \delta \text{ делится на } D.$$

Пример 2.4. Вернемся к задаче из примеров 2.2, 2.3. При решении задачи линейного программирования на минимум была получена табл. 2.3. Построим сечение, гарантирующее целочисленность оптимального значения целевой функции. Видим, что *D*=3, $f_3 = |-2\mu|_3 = |\mu|_3$, $f_5 = |-17\mu|_3 = |\mu|_3$, $f_0 = |-190\mu|_3 = |2\mu|_3$. Наибольше значение f_0 получается при μ=1, то есть

$$\frac{1}{3}x_3 + \frac{1}{3}x_5 \geq \frac{2}{3}.$$

Введем новую неотрицательную переменную соотношением

$$x_6 = \frac{1}{3}x_3 + \frac{1}{3}x_5 - \frac{2}{3}.$$

Осуществим итерацию двойственного симплекс-метода – из базиса выводится переменная x_6, вводится в базис переменная x_3. Получено оптимальное целочисленное решение (табл. 2.5).

Первый алгоритм Гомори заключается в следующем.

Итерация 0. Симплекс-методом решается задача линейного программирования (2.33)–(2.35), то есть задача без учета условия целочисленности. Если она неразрешима, задача целочисленного программирования также не имеет решений. Если оптимальный план задачи (2.33)–(2.35) целочисленный, то он является оптимальным и для исходной задачи. В противном случае, перейти к итерации 1.

Таблица 2.5

Добавление отсечения

	x_3	x_4	x_5			x_5	x_6	
x_1	–1/3	0	2/3	10/3	x_1	1	–1	4
x_4	1	1	0	8	x_4	–1	3	6
x_2	2/3	0	–1/3	10/3	x_2	–1	2	2
x_6	–1/3	0	–1/3	–2/3	x_3	1	1	2
Δ	–2/3	0	–17/3	–190/3	Δ	–5	0	–62

Итерация k, k=1,2,…. Пусть x^0 – нецелочисленный оптимальный опорный текущей задачи линейного программирования, основные переменные и целевая функция выражены через небазисные переменные (равенства (2.37), (2.38)).

Выбирается наименьшая по номеру нецелочисленная компонента вектора x^0 или, если гарантирована целочисленность целевой функции, вектора $\left(\left\langle c, x^0 \right\rangle, x^0\right)$, и строится правильное отсечение вида (2.41).

Задача линейного программирования с добавленным ограничением решается двойственным симплекс-методом, как указано в замечании 2.2. Если эта задача не имеет решений, то и исходная задача неразрешима. Если оптимальный план расширенной задачи целочисленный, то он является оптимальным планом исходной задачи. В противном случае перейти к итерации k+1.

Если в процессе решения появится строка симплекс-таблицы, в которой правый элемент дробный, а остальные элементы целые, то соответствующее уравнение (2.37) или (2.38) не имеет решения в целых числах, следовательно, не имеет решения и исходная задача.

Для однозначности определения оптимальных планов вспомогательных задач линейного программирования рекомендуется использовать подходы, основанные на лексигрофическом упорядочении векторов (см. Замечание 2.1).

Так как правильные отсечения нужны только для удаления текущего нецелочисленного оптимального плана и перехода к следующей итерации алгоритма, если дополнительная переменная $x_{n+k} = \sum_{j \in \omega} \frac{f_j}{D} x_j - \frac{f_0}{D}$ снова на каком-то этапе должна быть введена в базис, то после преобразования симплексной таблицы эту переменную можно больше не учитывать. Такой подход позволяет предотвратить чрезмерное увеличение размера вспомогательных задач в случае, если процесс решения потребует большого количества итераций.

Первый алгоритм Гомори неприменим в следующих случаях:
целевая функция не ограничена снизу на множестве X;
множество оптимальных планов задачи линейного программирования (множество оптимальных опорных планов множества X) неограничено.

Теорема 6. *Пусть выполнены следующие условия: 1) гарантирована целочисленность целевой функции (например, все* c_j *целые,* $j = 1,..., n$*) и равенство (2.38) учитывается при выборе строки для построения правильного отсечения; 2) по крайней мере одно из следующих двух утверждений верно: целевая функция ограничена снизу на X; множество* X^c *не пусто. Тогда первый алгоритм Гомори требует лишь конечного числа итераций.*

Пример 2.5. Рассмотрим следующую задачу [7]:

$$-x_1 - x_2 \to \min$$
$$2x_1 + 11x_2 \le 38,$$
$$x_1 + x_2 \le 7,$$
$$4x_1 - 5x_2 \le 5,$$
$$x_1 \ge 0,\ x_2 \ge 0 \text{ и целые.}$$

Вводя дополнительные неотрицательные переменные, приводящие задачу к канонической форме и решая задачу симплекс-методом, получаем следующую итоговую симплексную таблицу (табл.2.6):

Таблица 2.6

Решение линейной задачи

	x_1	x_2	x_3	x_4	x_5	
x_1	1	0	0	5/9	1/9	40/9
x_2	0	1	0	4/9	–1/9	23/9
x_3	0	0	1	–6	1	1
Δ	0	0	0	–1	0	–7

Оптимальный опорный план (40/9,23/9,1,0,0), минимальное значение целевой функции равно –7.

Итерация 1. Строим сечение, выражающее целочисленность переменной x_1. D=9, $f_4 = |5\mu|_9$, $f_5 = |\mu|_9$, $f_0 = |40\mu|_9 = |4\mu|_9$. Наибольшее значение f_0 достигается при μ=2, то есть получаем ограничение

$$\frac{1}{9}x_4 + \frac{2}{9}x_5 \ge \frac{8}{9}.$$

Введем новую неотрицательную переменную соотношением

$$x_6 = \frac{1}{9}x_4 + \frac{2}{9}x_5 - \frac{8}{9},$$

добавим это равенство к ограничениям задачи. Переменная x_6 выводится из базиса, в базис вводится переменная x_5. Из базиса выводится x_3, вводится в базис x_4 (табл. 2.7). Получено оптимальное решение вспомогательной задачи.

Итерация 2. Строим сечение, выражающее целочисленность значения целевой функции:

$$D=13,\ f_3 = |-2\mu|_{13} = |11\mu|_{13},\ f_6 = |-9\mu|_{13} = |4\mu|_{13},\ f_0 = |-85\mu|_{13} = |6\mu|_{13}.$$

Наибольшее значение f_0 получается при μ=2, то есть сечение имеет вид

$$\frac{9}{13}x_3 + \frac{8}{13}x_6 \geq \frac{12}{13}.$$

Таблица 2.7

Первая итерация

	x_4	x_5			x_4	x_6			x_3	x_6	
x_1	5/9	1/9	40/9	x_1	0,5	0,5	4	x_1	1/13	11/13	49/13
x_2	4/9	–1/9	23/9	x_2	0,5	–0,5	3	x_2	1/13	–2/13	36/13
x_3	–6	1	1	x_3	–6,5	4,5	–3	x_4	–2/13	–9/13	6/13
x_6	–1/9	–2/9	–8/9	x_5	0,5	–4,5	4	x_5	1/13	–54/13	49/13
Δ	–1	0	–7	Δ	–1	0	–7	Δ	–2/13	–9/13	–85/13

Введем новую неотрицательную переменную и добавим новое ограничение к ограничениям задачи. Переменная x_7 выводится из базиса, в базис возвращается x_3 (табл. 2.8). Получено оптимальное решение вспомогательной задачи.

Таблица 2.8

Вторая итерация

	x_3	x_6			x_6	x_7	
x_1	1/13	11/13	49/13	x_1	7/9	1/9	33/9
x_2	1/13	–2/13	36/13	x_2	–2/9	1/9	24/9
x_4	–2/13	–9/13	6/13	x_4	–5/9	–2/9	6/9
x_5	1/13	–54/13	49/13	x_5	–38/9	1/9	33/9
x_7	–9/13	–8/13	–12/13	x_3	8/9	–13/9	12/9
Δ	–2/13	–9/13	–85/13	Δ	-5/9	–2/9	–57/9

Итерация 3. Строим сечение, выражающее целочисленность значения целевой функции:

$$D=9,\ f_6 = |-5\mu|_9 = |4\mu|_9,\ f_7 = |-2\mu|_9 = |7\mu|_9,\ f_0 = |-57\mu|_9 = |6\mu|_9.$$

Наибольшее значение f_0 получается при μ=1, то есть сечение имеет вид

$$\frac{4}{9}x_6 + \frac{7}{9}x_7 \geq \frac{2}{3}.$$

Вводим новую неотрицательную переменную и добавляем новое ограничение к задаче. Выводим x_8 из базиса. В базис должна вернуться вспомогательная переменная x_7. Соответствующую ей строку не восстанавливаем, больше эту переменную не учитываем. В результате получено оптимальное решение вспомогательной задачи (табл.2.9).

Итерация 4. Строим сечение, выражающее целочисленность значения целевой функции:

$$D=7, f_6 = |-3\mu|_7 = |4\mu|_7,\ f_8 = |-2\mu|_7 = |5\mu|_7,\ f_0 = |-43\mu|_7 = |6\mu|_7.$$

Наибольшее значение f_0 получается при μ=1, то есть сечение имеет вид

$$\frac{4}{7}x_6 + \frac{5}{7}x_8 \geq \frac{6}{7}.$$

Таблица 2.9

Третья итерация

	x_6	x_7			x_6	x_8	
x_1	7/9	1/9	33/9	x_1	5/7	1/7	25/7
x_2	–2/9	1/9	24/9	x_2	–2/7	1/7	18/7
x_4	–5/9	–2/9	6/9	x_4	–3/7	–2/7	6/7
x_5	–38/9	1/9	33/9	x_5	–30/7	1/7	25/7
x_3	8/9	–13/9	12/9	x_3	12/7	–13/7	18/7
x_8	–4/9	–7/9	–2/3	Δ	–3/7	–2/7	–43/7
Δ	–5/9	–2/9	–57/9				

Вводим новую неотрицательную переменную и добавляем новое ограничение к задаче. Выводим x_9 из базиса. В базис должна вернуться вспомогательная переменная x_8. Соответствующую ей строку не восстанавливаем, больше эту переменную не учитываем. Получено оптимальное решение вспомогательной задачи (табл. 2.10).

Таблица 2.10

Четвертая итерация

	x_6	x_8			x_6	x_9	
x_1	5/7	1/7	25/7	x_1	0,6	0,2	3,4
x_2	–2/7	1/7	18/7	x_2	–0,4	0,2	2,4
x_4	–3/7	–2/7	6/7	x_4	–0,2	–0,4	1,2
x_5	–30/7	1/7	25/7	x_5	–4,4	0,2	3,4
x_3	12/7	–13/7	18/7	x_3	3,2	–2,6	4,8
x_9	–4/7	–5/7	–6/7	Δ	–0,2	–0,4	–5,8
	–3/7	–2/7	–43/7				

Итерация 5. Строим сечение, выражающее целочисленность значения целевой функции: D=5, $f_6 = |-\mu|_5 = |4\mu|_5$, $f_9 = |-2\mu|_5 = |3\mu|_5$, $f_0 = |-29\mu|_5 = |\mu|_5$. Наибольшее значение f_0 получается при μ=4, то есть сечение имеет вид

$$\frac{1}{5}x_6 + \frac{2}{5}x_9 \geq \frac{4}{5}.$$

Таблица 2.11

Пятая итерация

	x_6	x_9			x_6	x_{10}	
x_1	0,6	0,2	3,4	x_1	0,5	0,5	3
x_2	0,4	0,2	2,4	x_2	–0,5	0,5	2
x_4	–0,2	–0,4	1,2	x_4	0	–1	2
x_5	–4,4	0,2	3,4	x_5	–4,5	0,5	3
x_3	3,2	–2,6	4,8	x_3	4,5	–6,5	10
x_{10}	–0,2	–0,4	–0,8	Δ	0	–1	–5
Δ	–0,2	–0,4	–5,8				

Вводим новую неотрицательную переменную и добавляем новое ограничение к задаче. Выводим x_{10} из базиса. В соответствии с лексикографическим упорядочиванием, в базис должна вернуться вспомогательная переменная x_9. Соответствующую ей строку не восстанавливаем, больше эту переменную не учитываем (табл. 2.11). Получили оптимальное решение вспомогательной задачи. Так как оно целочисленное, найдено оптимальное решение исходной задачи $x_1 = 3$, $x_2 = 2$, минимальное значение целевой функции равно –5.

2.4. Метод убывающих конгруэнтностей

Перечисление вершин многогранника X' в соответствии с лексикографическим упорядочением, используемое в алгоритме Гомори и других классических методах отсечений, часто приводит к необходимости осуществлять большое количество итераций двойственного симплекс-метода. Рассмотрим алгоритм отсечений, конечность которого гарантирована и в котором не требуется на каждом шаге полностью решать вспомогательную задачу линейного программирования. Вместо этого после добавления отсечения осуществляется только одна итерация двойственного симплекс-метода, выводящая из базиса дополнительную переменную, связанную с отсечением. Если отсечение построено на основе равенства (2.38), то каждый такой шаг приводит к строгому возрастанию значения целевой функции и, следовательно, за конечное число шагов оно станет целым. Этот алгоритм называется методом убывающих конгруэнтностей [9].

Пусть $\left\langle c, x^0 \right\rangle$ – целое и добавлено сечение (2.41), полученное из равенства (2.37). Базисная матрица примет вид

$$\begin{pmatrix} A_\sigma & 0 \\ 0 & 1 \end{pmatrix},$$

где A_σ - предыдущая базисная матрица. Если в базис вводится переменная x_k, $k \in \omega$, то есть последний единичный столбец заменяется на вектор $\left(a^k, -f_k / D\right)$, модуль определителя полученной новой базисной матрицы станет равен

$$D' = D\frac{f_k}{D} = f_k \le D - 1.$$

Следовательно, за конечное число шагов можно получить целочисленный псевдоплан.

Экспериментально установлено, что если выбирать μ в (2.41) так, чтобы f_0 принимало максимальное значение, число шагов, необходимых для получения целочисленного псевдоплана, в среднем имеет порядок $\log_2 D$.

Поскольку многогранник X ограничен, множество X^c конечно и алгоритм может быть улучшен прямым перечислением целочисленных точек.

Обозначим $\omega' = \{j \in \omega : \Delta_j = 0\}$. Если $\omega' = \varnothing$, итерация двойственного смплекс-метода приводит к строгому возрастанию значения целевой функции. В противном случае, ввиду (2.19) в базис должна будет вернуться переменная x_k такая, что $k \in \omega'$.

Рассмотрим многогранник

$$X' = \{x' \in R^{n'} : A'x' = b, x' \geq 0\},$$

где A' – подматрица матрицы A, составленная из столбцов a^j, для которых $\Delta_j = 0$, то есть $n' = |\omega'| + m$. Если многогранник X' имеет целочисленную точку, то она соответствует оптимальному решению исходной задачи. В противном случае получаем, что в оптимальном целочисленном решении должна быть отлична от нуля хотя бы одна переменная x_j, $j \in \omega \setminus \omega'$ и неравенство

$$\sum_{j \in \omega \setminus \omega'} x_j \geq 1$$

является правильным отсечением. Шаг двойственного симплекс-метода после ввода этого отсечения приводит к строгому возрастанию значения целевой функции.

Таким образом, **алгоритм сечений методом убывающих конгруентностей** может быть осуществлён следующим образом.

Шаг 0. Решить задачу линейного программирования (2.33)–(2.35). Если полученное решение x^0 целочисленное, то оно является оптимальным решением исходной задачи. В противном случае – шаг 1.

Шаг 1. Пусть найдено допустимое решение x' исходной задачи. Так как

$$\langle c, x \rangle = \langle c, x^0 \rangle - \sum_{j \in \omega} \Delta_j x_j ,$$

где (в задаче на минимум) все $\Delta_j \leq 0$, в оптимальном решении

$$\langle c, x' \rangle - \langle c, x^0 \rangle \geq \sum_{j \in \omega} (-\Delta_j) x_j .$$

Следовательно, если $-\Delta_j > \langle c, x' \rangle - \langle c, x^0 \rangle$, переменная x_j должна быть равна нулю и её можно исключить из задачи.

Шаг 2. Пусть $\omega' = \{j \in \omega : \Delta_j = 0\}$. Если значение целевой функции в точке x^0 является целым и $\omega' \neq \varnothing$, перейти к шагу 4. Иначе, добавляя сечения, выражающие целочисленность базисных переменных, и осуществляя одну итерацию двойственного симплекс-метода, получить целочисленный псевдоплан расширенной задачи. Если это решение является допустимым, то есть $x_j \geq 0$ для всех j, то это оптимальное решение исходной задачи. В противном случае идти на шаг 3.

Шаг 3. Решить расширенную задачу двойственным симплекс-методом. Если полученный оптимальный опорный план целочисленный, то он является оптимальным решением исходной задачи. Иначе перейти на шаг 2.

Шаг 4. Найти с помощью неявного перечисления целочисленное решение задачи

$$Ax = b,\ x_j = 0 \text{ для } j \in \omega \setminus \omega'.$$

Если такое решение существует, то это — оптимальное решение исходной задачи. Иначе — добавить сечение

$$\sum_{j \in \omega \setminus \omega'} x_j \geq 1,$$

осуществить одну итерацию двойственного симплекс-метода и вернуться к шагу 2.

На каждом шаге алгоритма за конечное число действий либо получаем оптимальное решение, либо добиваемся строгого увеличения значения целевой функции. Следовательно, алгоритм является сходящимся. Как и в алгоритме Гомори, получение целочисленного решения рассматривается как приоритетная задача по отношению к получению оптимального решения. Для некоторых классов задач, особенно для таких, в которых матрица A имеет не слишком большие миноры (для примера, меньше 100), рассмотренный алгоритм порождает малое количество сечений даже для задач большого размера. Например, для задач покрытия, имеющих 2000 переменных и 120 ограничений, количество отсечений часто меньше десяти.

Пример 2.6. Применим указанный подход к решению задачи из примера 2.5. Для удобства систему ограничений канонической формы задачи преобразуем к виду

$$9x_2 + x_3 = 24 + 2x_4,$$
$$x_1 + x_2 + x_4 = 7,$$
$$x_3 + x_5 = 1 + 6x_4.$$

Оптимальное решение (40/9,23/9,1,0,0) задачи линейного программирования не является целочисленным, допустимых решений задачи не найдено, поэтому осуществляется шаг 2 алгоритма.

Шаг 2. $\omega = \{4,5\}$, $\omega' = \{5\}$, значение целевой функции равно –7, идем на шаг 4.

Шаг 4. Система $Ax=b$, $x_4 = 0$ не имеет неотрицательных целочисленных решений, так как x_3 может принимать значения либо 0, либо 1 (третье уравнение), следовательно, $9x_2 = 24$ либо $9x_2 = 23$. Добавляем ограничение

$$x_4 \geq 1,$$

вводим новую неотрицательную переменную (табл. 2.12). Переменная x_6 выводится из базиса, переменная x_4 вводится в базис.

Шаг 2. $\omega = \{5,6\}$, $\omega' = \{5\}$, идем на шаг 4.

Шаг 4. Если $x_6 = 0$, то $x_4 = 1$, $x_3 \leq 7$, $9x_2 + x_3 = 26$. Целочисленных неотрицательных решений эта система не имеет, добавляем ограничение

$$x_6 \geq 1,$$

вводим новую неотрицательную переменную (табл. 2.13). Переменная x_7 выводится из базиса, переменная x_6 вводится в базис.

Таблица 2.12

Итерация двойственного симплекс-метода

	x_4	x_5			x_5	x_6	
x_1	5/9	1/9	40/9	x_1	1/9	5/9	35/9
x_2	4/9	–1/9	23/9	x_2	–1/9	4/9	19/9
x_3	–6	1	1	x_3	1	–6	7
x_6	–1	0	–1	x_4	0	–1	1
Δ	–1	0	–7	Δ	0	–1	–6

Шаг 2. $\omega = \{5,7\}$, $\omega' = \{5\}$, значение целевой функции целое.

Таблица 2.13

Итерация двойственного симплекс-метода

	x_5	x_6			x_5	x_7	
x_1	1/9	5/9	35/9	x_1	1/9	5/9	10/3
x_2	–1/9	4/9	19/9	x_2	–1/9	4/9	5/3
x_3	1	–6	7	x_3	1	–6	13
x_4	0	–1	1	x_4	0	–1	2
x_7	0	–1	–1	x_6	0	–1	1
Δ	0	–1	–6	Δ	0	–1	–5

Шаг 4. $x_7 = 0$, $x_6 = 1$, $x_4 = 2$, следовательно

$$9x_2 + x_3 = 28,$$
$$x_1 + x_2 = 5,$$
$$x_3 + x_5 = 13.$$

Решения этой системы $x_1 = 3$, $x_2 = 2$, $x_3 = 10$, $x_5 = 3$ и $x_1 = 2$, $x_2 = 3$, $x_3 = 1$, $x_5 = 12$ отвечают оптимальным решениям (3,2) и (2,3) исходной задачи.

3. МЕТОДЫ ВЕТВЕЙ И ГРАНИЦ

3.1 Общая схема методов ветвей и границ

В данной главе рассматривается семейство методов, носящее название методов ветвей и границ, основанных, во-первых, на разбиении множества допустимых решений на подмножества (ветвлении), и, во-вторых, на оценивании целевой функции на этих подмножествах (вычислении границ).

Дерево поиска

Рассмотрим задачу комбинаторной оптимизации в следующей общей форме:

$$f(x) \to \min, \quad x \in D, \tag{3.1}$$

где D – некоторое конечное множество.

Ветвлением называется разбиение исходной задачи на некоторое число подзадач, в целом составляющих всю задачу (3.1). Как правило, любую подзадачу задачи (3.1) можно записать в виде

$$f(x) \to \min, \ x \in D', \ D' \subset D, \tag{3.2}$$

поэтому ветвление эквивалентно разбиению допустимого множества D на подмножества, то есть

$$D = \bigcup_{i=1}^{k} D_i, \ D_i \subset D. \tag{3.3}$$

Далее процедуру ветвления можно применить к подзадачам с допустимыми множествами D_i, затем к задачам, полученным в результате этого ветвления и так далее. Процесс конечен, поскольку полное дерево поиска содержит в качестве листьев одноэлементные подмножества множества D.

На каждом этапе ветвления имеющаяся совокупность подмножеств множества D частично упорядочена по включению, и это отношение порядка можно изобразить в виде ациклического графа, вершинами которого являются полученные подмножества. Этот граф называется *деревом поиска* или *деревом решений*. Множества D', D'' связаны дугой (D', D''), если D'' получено непосредственно в результате ветвления D', причем D'' называется сыном или потомком вершины D'. Корню дерева (вершине с нулевой степенью захода) отвечает множество D. Множества, не подвергнутые ветвлению, образуют концевые вершины дерева или листья.

Поскольку каждое множество, соответствующее вершине дерева поиска, является объединением своих потомков, множество D равно объединению множеств, соответствующих концевым вершинам дерева. Чтобы решить задачу (3.1), необходимо, следовательно, для каждой концевой подзадачи либо найти её оптимальное решение, либо показать, что допустимое множество этой подзадачи заведомо не содержит оптимального решения задачи (3.1). Методы решения задачи (3.1), использующие дерево поиска, основаны на том, что

получающиеся в результате ветвления подзадачи решаются проще, чем исходная задача.

Построение дерева решений может происходить по следующей схеме. На начальном, нулевом шаге $D^0 = D$. Осуществляется ветвление в корневой вершине: $D^0 = \bigcup_{i=1}^{r_1} D_i^1$.

Пусть на шаге с номером $k \geq 1$ концевые вершины построенного дерева обозначены D_1^k, D_2^k, …, $D_{r_k}^k$. Среди них выбирается множество D_s^k и разбивается на конечное число подмножеств: $D_s^k = \bigcup_{t=1}^{r} D_{s,t}^k$.

Множество D_s^k заменяется совокупностью множеств $D_{s,t}^k$ и все еще не подвергшиеся разбиению множества обозначаются через

$$D_1^{k+1},\ D_2^{k+1},\ \ldots,\ D_{r_{k+1}}^{k+1}.$$

Существуют и более сложные системы индексации подмножеств, при которых не требуется их переобозначение на каждом шаге. Примеры деревьев поиска – на рис. 3.2 – 3.4, 3.6.

Для каждой конкретной задачи существует много способов представления ее в виде дерева. *Правило ветвления* может быть задано с помощью некоторого оператора, определенного на множестве подмножеств множества D и ставящего в соответствие каждому $D' \subset D$, $|D'| > 1$, конечное число подмножеств $D_i' \subset D'$ таких, что $D' = \bigcup D_i'$.

Наибольшее распространение получили два следующих правила ветвления. Первое из них, дихотомическое ветвление, связано с разбиением по некоторому признаку множества решений D' на два непересекающихся множества – D'' и его дополнение $\overline{D''}$. Второе реализует так называемое покомпонентное ветвление, осуществляемое путем фиксирования значений переменных. В этом случае всякий путь из корня к некоторой вершине отвечает частичному решению задачи.

Например, пусть некоторая переменная x_j может принимать в текущей задаче значения a_i, $i = 1,...,k_j$. Тогда покомпонентное ветвление по переменной x_j означает разбиение на k_j подзадач, *i*-я подзадача определяется введением соответствующего дополнительного условия $x_j = a_i$. При дихотомическое ветвлении две подзадачи задаются, чаще всего, с помощью условий вида $x_j = a$ и $x_j \neq a$ соответственно.

Оба указанных выше способа ветвления удовлетворяют условию

$$D_i' \bigcap D_j' = \varnothing,\ i \neq j,$$

определяющему собственное разбиение задачи. Это условие означает, в частности, что любое допустимое решение задачи (3.1) принадлежит ровно одной концевой вершине дерева поиска.

Применение границ

Чтобы ускорить поиск решения, избежав перебора «неперспективных» вариантов, могут быть использованы два типа оценок: нижние границы для значений целевой функции на подмножестве допустимых решений и верхние границы для оптимального значения целевой функции.

Пусть $D' \subseteq D$ – некоторое множество, получаемое в процессе ветвления. Число $\xi(D')$ называется *нижней оценкой* целевой функции f на множестве D', если

$$f(x) \geq \xi(D') \text{ для любого } x \in D'.$$

Таким образом, оценка ξ – функция, определенная на множестве подмножеств множества допустимых решений. При этом если $D' = \{x\}$, считается, что $\xi(D') = f(x)$ и $\xi(\varnothing) = +\infty$.

Для вычисления $\xi(D')$ в случае, когда $|D'| > 1$, как правило, решается некоторая оптимизационная задача простой структуры, являющаяся оценочной для задачи (3.2). Обычно от оценочной задачи требуют выполнения следующих условий:

а) если оценочная задача не имеет допустимых решений, то и задача (3.2) их не имеет;

б) оптимальное значение целевой функции в задаче (3.2) не меньше оптимального значения целевой функции в оценочной задаче.

Поэтому оценочную задачу можно получить либо исключением некоторых условий, задающих D', например условий целочисленности переменных, или заменой целевой функции $f(x)$ минорантой $g(x) \leq f(x)$.

Если $D_1 \subset D_2$, то, очевидно,

$$\min_{x \in D_1} f(x) \geq \min_{x \in D_2} f(x).$$

Поэтому, производя ветвление в вершине $D' \subseteq D$, можно считать, что

$$\xi(D'_i) \geq \xi(D') \quad \forall i.$$

Пусть, далее, $x' \in D$ – некоторое допустимое решение задачи (3.1). Очевидно, что $f(x')$ – верхняя граница для оптимального значения целевой функции на D. Наилучшее из имеющихся допустимых решений принято называть рекордным, а его значение – *рекордом*. Если допустимые решения задачи не известны, полагают, что рекорд равен $+\infty$. Для нахождения элементов множества D могут быть использованы, например, приближенные алгоритмы решения задачи.

Рассмотрим теперь применение границ. Пусть на некотором этапе ветвления концевые вершины обозначены D_i, $i = 1,..., k$, то есть выполнено (3.3), вычислены оценки $\xi(D_i)$, $x' \in D$ – рекордное решение. Положим $\xi_0 = \min_{1 \leq i \leq s} \xi(D_i)$. Тогда из определения границ вытекает неравенство

$$\xi_0 \leq \min_{x \in D} f(x) \leq f(x'). \qquad (3.4)$$

Разность $\Delta = f(x') - \xi_0$ является оценкой гарантированного отклонения рекорда от оптимума. Если эта разность невелика, то x' можно принять за приближенное решение, а Δ – за оценку точности приближения.

Далее, если $\bar{x} \in D_i$, и $f(\bar{x}) = \xi(D_i)$, то, очевидно, $\bar{x}$ – оптимальное решение на множестве D_i. Из (3.4) ясно, что $\bar{x}$ – оптимальное решение задачи (3.1), если

$$f(\bar{x}) = \xi(D_i) = \xi_0. \tag{3.5}$$

Пусть для некоторого множества D_r

$$\xi(D_r) > f(x'). \tag{3.6}$$

В соответствии с (3.4) и критерием оптимальности (3.5), D_r не может содержать оптимальных решений, поэтому его можно исключить из дальнейшего рассмотрения, то есть отсеять. На практике часто отсеивают множества, удовлетворяющие условию

$$\xi(D_r) \geq f(x'), \tag{3.7}$$

которое означает, что множество D_r не содержит решений, лучших, чем уже найденное решение x'.

Таким образом, правило (3.6) гарантирует, что будут найдены все оптимальные решения задачи (3.1), соотношение (3.7) – что будет найдено хотя бы одно оптимальное решение. Проверка условий (3.6) или (3.7) называется *элиминирующим (исключающим) тестом*.

Если

$$\xi(D_r) < f(x') \leq \min_{D_r} f(x),$$

то тест (3.7) не может исключить подмножество D_r, хотя оно не содержит допустимого решения, лучше рекордного. Поэтому желательно, чтобы оценка $\xi(D_r)$ была как можно ближе к оптимальному значению $f(x)$ на D_r. К сожалению, в большинстве случаев, чем точнее способ вычисления нижних границ, тем он сложнее или требует большего времени. На практике можно использовать несколько способов вычисления оценки, начиная с самого простого.

Функция ветвления

В зависимости от того, в каком порядке порождаются и рассматриваются вершины дерева решений, различают два основных типа обхода вершин – поиск по глубине и поиск по ширине.

При *поиске по глубине* ветвление осуществляется в последней полученной подзадаче до тех пор, пока не будет порождена подзадача, которую можно разрешить. В этом месте делается шаг возвращения, т. е. берется предпоследняя порожденная подзадача и ветвление продолжается в соответствующей вершине. Таким образом, если на шаге k ветвление производилось в вершине D_s^k, то на шаге k+1 для ветвления будет выбран один из её потомков $D_{s,t}^k$.

При *поиске по ширине* ветвление происходит от уровня к уровню, так что каждая из задач D_i^k, $i \neq s$, исследуется раньше, чем задачи $D_{s,t}^k$, полученные при ветвлении в вершине D_s^k.

Таким образом, при каждом типе поиска на каждом шаге построения дерева решений определено множество допустимых вершин, из которых может производиться ветвление. Для того, чтобы выбрать из этого множества одну вершину, используется *функция ветвления*, определенная на множестве подмножеств *D*. Для вершины $D' \subset D$ значение этой функции $\lambda(D')$ является некоторой мерой возможности того, что оптимальное решение задачи (3.1) содержится в D'. Очевидно, что вершина, соответствующая подзадаче с большими шансами на оптимальное решение, должна пользоваться правом преимущественного выбора при очередном ветвлении. Значение $\lambda(D_j^k)$ приписывается множеству D_j^k в момент его образования и может совпадать, например, с нижней границей D_j^k, то есть $\lambda(D_j^k) = \xi\left(D_k^j\right)$. В последнем случае, в соответствии с (3.5), считается, что большую вероятность содержать оптимальное решение имеет вершина с меньшей границей, поэтому наиболее перспективное подмножество выбирается из соотношения

$$\lambda(D_j^k) = \min_i \lambda(D_i^k) \tag{3.8}$$

Такая схема ветвления называется *многосторонней* или одновременной. При *односторонней* схеме ветвления для исследования выбирается последнее полученное множество, то есть можно считать, что $\lambda(D_j^k) = j$ и рассматривается множество с наибольшим номером.

Функцию ветвления можно использовать, выбирая множество для ветвления из всей совокупности концевых вершин, а не только определенной типом поиска (по глубине или ширине). При этом задается новый тип поиска, гибрид двух основных, который, тем не менее, часто называют поиском по ширине.

Общая схема метода ветвей и границ

Пусть стоит задача (3.1), где *D* – конечное множество. На каждой итерации алгоритма происходит работа с некоторым подмножеством концевых вершин дерева поиска. Это множество называется списком задач-кандидатов или задач для ветвления. Задача решена, если список задач-кандидатов пуст, т.е. если все они отсеяны по правилам отсева.

На начальном шаге процесса список состоит из множества *D*. Некоторым способом вычисляется значение нижней оценки $\xi(D)$ для целевой функции.

Пусть можно указать план $x' \in D$. Если $f(x') = \xi(D)$, то x' – оптимальное решение задачи. В противном случае полагаем $x_0 = x'$ – рекордное решение, рекорд равен $f_0 = f(x_0)$. Если допустимых планов не найдено, $f_0 = +\infty$.

Стандартная (*k*-я) итерация алгоритма состоит из следующих этапов:

1. Если список кандидатов пуст, прекратить работу, при этом, если рекорд конечен, рекордное решение является оптимальным, в противном случае задача (3.1) не имеет допустимых решений.

2. Выбрать для ветвления одно из множеств списка.

3. Осуществить ветвление. Модифицировать список.

4. Для каждого подмножества, получившегося в результате ветвления, найти нижние границы $\xi(D_i^k)$, $i = 1,2,...,r_k$.

5. Если становятся известны допустимые решения (например, если $D_i^k = \{x\}$), откорректировать сведения о рекорде. Пусть X_k – множество допустимых решений, полученных на *k*-й итерации. Тогда рекорд равен

$$f_k = \min\left\{f_{k-1}, \min_{x \in X_k} f(x)\right\}, \tag{3.9}$$

и рекордное решение x^k есть допустимое решение, на котором достигается минимум в (3.9).

6. Проверить выполнение правила отсева. Если для некоторого множества D_i^k имеет место (3.7), то исключить его из списка.

В схеме с одновременным ветвлением на шаге 2 выбирается множество с использованием (3.8), при одностороннем ветвлении рассматривается последнее из множеств, добавленных к списку.

При использовании одностороннего ветвления итерация алгоритма может быть изменена следующим образом. После шага 2 – выбора множества для ветвления – осуществляются шаги 4 – 6 алгоритма для этого множества, то есть вычисляется его оценка, корректируются сведения о рекорде. Далее рассматриваемое множество либо исключается из списка по правилу отсева, либо производится ветвление и модифицируется список.

Схема с многосторонним ветвлением требует много оперативной памяти, но в среднем просматривает меньше вершин, чем односторонняя.

Конечность алгоритма следует из конечности множества *D*, правила ветвления, гарантирующего конечность дерева (например, при покомпонентном и дихотомическом ветвлении путь от корня к висячей вершине содержит не более |*D*| ребер), определения оценки и рекорда, гарантирующих отсев концевых вершин.

Алгоритм ветвей и границ можно остановить до его завершения, используя рекордное решение x^k в качестве приближенного решения задачи. Согласно (3.4), относительная погрешность такого решения не превосходит

$$\frac{f(x^k) - \xi_0}{\xi_0}.$$

Для одной и той же задачи можно строить различные алгоритмы метода ветвей и границ, конкретизируя правила ветвления и выбора кандидата, методы вычисления оценок. Эффективность полученных алгоритмов определяется числом решенных задач.

Впервые метод ветвей и границ был предложен в 1960 г. в работе Лэнд и Дойг применительно к задаче целочисленного линейного программирования. Однако возникновение интереса к методу ветвей и границ обычно связывают с посвященной задаче коммивояжера работой Литтла, Мурти, Суини и Кэрела, в которой были сформулированы основные принципы метода и предложено его название.

Замечание 3.1. Иногда для того, чтобы облегчить применение методов ветвей и границ, вводят конечное множество $\Omega \supseteq D$, например, избавившись от наиболее жестких ограничений, задающих D, и операции ветвления и вычисления нижних границ выполняют с подмножествами множества Ω.

Замечание 3.2. При применении методов ветвей и границ для решения задачи максимизации функции на конечном множестве, вместо нижних границ для значений целевой функции в вершине D_i дерева поиска определяются верхние границы:

$$f(x) \le \xi(D_i) \text{ для всех } x \in D_i .$$

Значение целевой функции в допустимой точке является нижней границей для оптимального значения целевой функции. Другие элементы метода (функция ветвления, правила отсева, критерий оптимальности) модифицируются соответственно:

если $D = \bigcup D_i$, где D_i — множества, еще не подвергавшиеся разбиению, x^* — оптимум, x' — рекордное решение, то

$$\xi_0 = \max_i \xi(D_i) \ge f(x^*) \ge f(x');$$

оценка гарантированного отклонения рекорда от оптимума $\Delta = \xi_0 - f(x')$;

x' – оптимальное решение, если $\xi_0 = f(x')$;

для ветвления выбирается множество с наибольшей оценкой;

множество D_i отсеивается, если $\xi(D_i) \le f(x')$.

3.2 Метод Ленд и Дойг для задачи частично целочисленного линейного программирования

Рассматривается задача частично целочисленного линейного программирования в стандартной форме:

$$f(x) = \sum_{j=1}^{n} c_j x_j \to \min \tag{3.10}$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i,\ i = 1,...,m, \tag{3.11}$$

$$l_j \le x_j \le d_j,\ j = 1,...,n, \tag{3.12}$$

$$x_j \text{ целые},\ j = 1,...,n_1,\ n_1 \le n. \tag{3.13}$$

Предполагается, $l_j \geq 0$, $j = 1,...,n$. Если ограничения (3.12) не заданы изначально, их можно получить, решив $2n$ задач линейного программирования

$$x_j \to \min \text{ и } x_j \to \max$$

при условиях (3.11) и $x_j \geq 0$, $j = 1,...,n$ и положив $d_j = \lfloor \max x_j \rfloor$ и $l_j = \min x_j$, если это число целое и $l_j = \lfloor \min x_j \rfloor + 1$ в противном случае.

Очевидно, что многогранное множество X, описываемое условиями (3.11), (3.12), ограниченное.

Алгоритм осуществляется по схеме, описанной в предыдущем параграфе. Вершины дерева поиска – это задачи частично целочисленного линейного программирования

$$f(x) \to \min,\ x \in D', \tag{3.14}$$
$$D' = \{x = (x_1,...,x_n) \in X' \subseteq X : x_j \in Z, j = 1,...,n_1\},$$

а оценочные задачи для них – соответствующие задачи линейного программирования, получаемые отбрасыванием условий целочисленности.

Если оценочная задача неразрешима, не имеет решений и задача (3.14) и, следовательно, соответствующая вершина дерева поиска удаляется из списка. Если решение оценочной задачи удовлетворяет условиям целочисленности (3.13), то оно является допустимым решением исходной задачи и используется для корректировки сведений о рекорде.

Предположим, на k-й итерации алгоритма (k=0,1,…) выбрано для ветвления множество D^k (на начальном этапе множество D^0 определяется условиями (3.11)-(3.13)), $\bar{x} = (\bar{x}_1,...,\bar{x}_n)$ – решение соответствующей оценочной задачи, и его компонента $\bar{x}_{j_0}$, где $1 \leq j_0 \leq n_1$, не является целой. Тогда множество D^k разбивается на множества D_1^k и D_2^k, первое из них образовано добавлением к ограничениям, задающим D^k, условия $x_{j0} \leq \lfloor \bar{x}_{j_0} \rfloor$, второе – добавлением условия $x_{j0} \geq \lfloor \bar{x}_{j_0} \rfloor + 1$. Если $\lfloor \bar{x}_{j_0} \rfloor < l_{j_0}$ или $\lfloor \bar{x}_{j0} \rfloor \geq d_{j_0}$ то множество D_1^k или D_2^k соответственно пусто.

Если x_{j_0} — булева переменная, то ветвление из вершины D^k соответствует заданию условия $x_{j_0} = 0$ для задачи D_1^k и условия $x_{j_0} = 1$ для задачи D_2^k, то есть, по сути, осуществляется покомпонентное ветвление.

Если нецелочисленных компонент вектора $\bar{x}$ несколько, ветвление осуществляется, например, по дробной компоненте, имеющей наименьший номер.

Замечание 3.3. Если в задаче (3.10)–(3.13) $n_1 = n$ и все c_j целые, то, поскольку значения целевой функции в допустимых точках целые и $f(x) \geq f(\bar{x})$ для $x \in D^k$, можно положить

$$\xi(D^k) = f(\bar{x}), \text{ если } \bar{x} \in D,\ \xi(D^k) = [f(\bar{x})] + 1, \text{ если } \bar{x} \notin D.$$

Другой способ учёта целочисленности целевой функции – считать, что $\xi(D^k) = f(\bar{x})$, но модифицировать правило отсева следующим образом.

Пусть x' — рекордное решение, D^i, i = 1,2,..., r, — множества, еще не подвергавшиеся разбиению. Предположим, что множество D^k не было отсеяно по обычным правилам отсева, т.е. $\xi(D^k) < f(x')$. Если

$$f(x') - \xi(D^k) < 1,$$

то множество D^k не содержит точек, лучших, чем x', и может быть отсеяно. Действительно, пусть $x \in D^k$. По определению оценки $f(x) > f(x') - 1$. Так как числа $f(x)$ и $f(x')$ целые, отсюда вытекает, что $f(x) \geq f(x')$. Отсюда следует, что задача решена, если $\Delta = f(x') - \min \xi(D_i) < 1$, при этом x' – оптимальное решение.

Пример 3.1. Применим метод Ленд и Дойг для решения следующей задачи:

$$2x_1 + 5x_2 \to \min$$
$$3x_1 + 3x_2 \geq 11$$
$$6x_1 - 3x_2 \leq 1$$
$$0 \leq x_1 \leq 4,\ 0 \leq x_2 \leq 4.$$

Решая задачу линейного программирования с этими ограничениями (см. рис. 3.1), получаем, что на начальном шаге $\bar{x} = (4/3, 7/3)$, $f(\bar{x}) = 43/3$. Поскольку коэффициенты целевой функции целые, $\xi(D) = [43/3] + 1 = 15$, рекорд $f_0 = +\infty$.

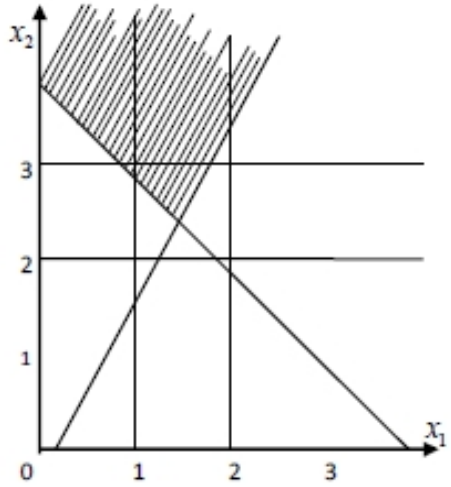

Рис. 3.1. Решение задачи из примера 3.1

Итерация 1. Выполняем ветвление $D = D_1^1 \cup D_2^1$, множество D_1^1 получено добавлением ограничения $x_1 \leq 1$, множество D_2^1 – добавлением ограничения $x_1 \geq 2$. Решая оценочные подзадачи, получаем: $\bar{x}^1 = (1, 8/3)$, $f(\bar{x}^1) = 46/3$, $\xi(D_1^1) = 16$, $\bar{x}^2 = (2, 11/3)$, $f(\bar{x}^2) = 67/3$, $\xi(D_2^1) = 23$.

Итерация 2. Производим ветвление $D_1^1 = D_1^2 \cup D_2^2$, где D_1^2 получено добавлением ограничения $x_2 \le 2$, множество D_2^2 - добавлением ограничения $x_2 \ge 3$. Решаем оценочные подзадачи. Множество D_1^2 пусто и соответствующая задача удаляется из списка. Для множества D_2^2: $\bar{x}^3 = (2/3, 3)$, $f(\bar{x}^3) = 49/3$, $\xi(D_2^2) = 17$.

Итерация 3. Производим ветвление $D_2^2 = D_1^3 \cup D_2^3$, где D_1^3 получено из D_2^2 добавлением ограничения $x_1 = 0$, множество D_2^3 - добавлением ограничения $x_1 = 1$. Оценочная задача для D_1^3 имеет решение $(0, 11/3)$, $\xi(D_1^3) = 19$, задача для D_2^3 имеет целочисленное решение $(1,3)$, которое даёт рекорд $f_0 = \xi(D_2^3) = 17$. Множества D_1^3, D_2^3, D_2^1 отсеиваются.

Итерация 4. Список допустимых вершин пуст, $x^* = (1,3)$ – оптимальное решение задачи. Дерево поиска – на рис. 3.2.

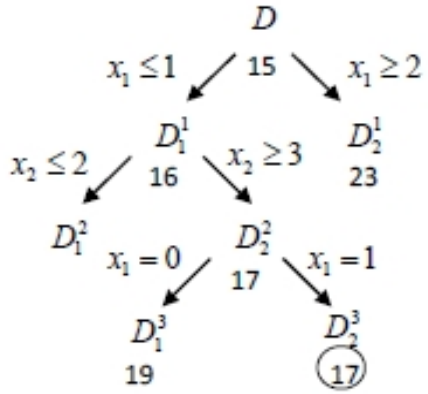

Рис. 3.2. Дерево поиска для примера 3.1

3.3 Метод Ленд и Дойг для задачи о рюкзаке

Ограничения в задачах о рюкзаке имеют вид (3.11) – (3.13), где $n_1 = n$ и $l_j = 0$, $d_j = 1$ для j=1,…,n, если рассматривается бинарный рюкзак и $d_j = \lfloor b / a_j \rfloor$ в противном случае. Поэтому может применяться рассмотренный в предыдущем параграфе алгоритм, соответствующим образом модифицированный для решения задачи на максимум (см. замечание 3.2).

Конкретизируем элементы метода Ленд и Дойг для задачи о бинарном ранце

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

$$\sum_{j=1}^{n} a_j x_j \le b, \qquad (3.15)$$

$$x_j \in \{0,1\},\ j=1,...,n,$$

$c_j > 0,\ 0 \le a_j \le b,\ j=1,...,n.$

При ветвлении значение одной из переменных полагается равным 0 или 1, поэтому подзадачи также имеют вид (3.15).

Оценочной для задачи (3.15) является задача линейного программирования

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

$$\sum_{j=1}^{n} a_j x_j \le b,\ 0 \le x_j \le 1,\ j=1,...,n. \qquad (3.16)$$

Алгоритм поиска оптимального решения этой задачи сформулирован в следующей теореме.

Теорема 3.1 (правило Данцига). *Пусть переменные* x_j, $1 \le j \le n$, *перенумерованы так, что* $\lambda_1 \ge \lambda_2 \ge \ldots \ge \lambda_n$, *где* $\lambda_j = c_j / a_j$. *Тогда оптимальное решение* $\bar{x} = (\bar{x}_1,...,\bar{x}_n)$ *задачи (3.16) имеет вид*

$$\bar{x}_1 = \bar{x}_2 = ... = \bar{x}_{s-1} = 1,\ \bar{x}_s = \left(b - \sum_{j=1}^{s-1} a_j \right) / a_s,\ \bar{x}_{s+1} = \bar{x}_{s+2} = ... = \bar{x}_n = 0,$$

где s определяется из условия $\sum_{j=1}^{s-1} a_j \le b < \sum_{j=1}^{s} a_j$.

Если вектор, полученный по правилу Данцига, целочисленный, то он является оптимальным решением соответствующей задачи (3.15). В противном случае, при $0 < \bar{x}_s < 1$, можно получить допустимое решение x' задачи, обнулив s-ю компоненту вектора $\bar{x}$, тогда

$$f(\bar{x}) - f(x') = c_s \bar{x}_s = \lambda_s \left(b - \sum_{j=1}^{s-1} a_j \right).$$

В соответствии с замечанием 3.3, если в задаче о рюкзаке все c_j целые, множество D^k может быть отсеяно, если

$$0 \le \xi(D^k) - f(x') < 1,$$

где x' — рекордное решение. Задача решена, если $\Delta = \max \xi(D_i) - f(x') < 1$, при этом x' – оптимальное решение.

Пример 3.2. Решим задачу

$$5x_1 + 3x_2 + 4x_3 + 4x_4 \to \max$$
$$2x_1 + 2x_2 + x_3 + 2x_4 \le 4,$$
$$x_i \in \{0,1\},\ i=1,\ldots,4.$$

Находим решение оценочной задачи по правилу Данцига. Так как

$$\lambda_1 = 5/2,\ \lambda_2 = 3/2,\ \lambda_3 = 4,\ \lambda_4 = 2,$$

$\bar{x} = (1,0,1,1/2)$, $\xi(D) = f(\bar{x}) = 11$, допустимое решение – $x' = (1,0,1,0)$, $f(x') = 9$.

Производим ветвление $D = D_1^1 \cup D_2^1$, в первой задаче $x_4 = 0$, во второй - $x_4 = 1$. Решаем оценочные задачи: $\bar{x}^1 = (1,1/2,1,0)$, $\xi(D_1^1) = 10,5$, $\bar{x}^1 = (1/2,0,1,1)$, $\xi(D_2^1) = 10,5$. Допустимое решение, построенное по вектору $\bar{x}^1$, совпадает с x', решение $(0,0,1,1)$, полученное из $\bar{x}^2$, хуже рекордного.

Производим ветвление $D_1^1 = D_1^2 \cup D_2^2$ ($x_2 = 0$ или $x_2 = 1$ соответственно), обозначаем D_2^1 через D_3^2. Решаем оценочные задачи: $\bar{x}^1 = (1,0,1,0)$, $\xi(D_1^2) = 9$, $\bar{x}^2 = (1/2,1,1,0)$, $\xi(D_2^2) = 9,5$. Так как коэффициенты целевой функции целые, множества D_1^2 и D_2^2 могут быть отсеяны.

Производим ветвление $D_3^2 = D_1^3 \cup D_2^3$, фиксируя на 0 и 1 значение переменной x_1. Решаем оценочные задачи: $\bar{x}^1 = (0,1/2,1,1)$, $\xi(D_1^3) = 9,5$, $\bar{x}^2 = (1,0,0,1)$, $\xi(D_2^2) = 9$. Множества D_1^3 и D_2^3 могут быть отсеяны, $x' = (1,0,1,0)$ – оптимальное решение задачи (рис. 3.3).

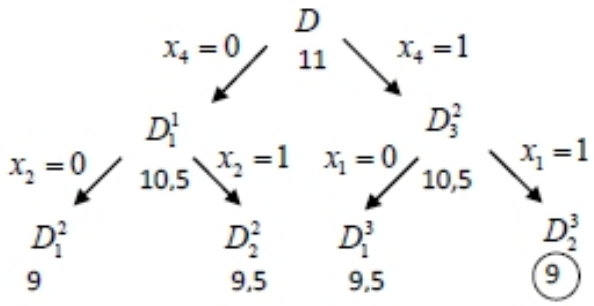

Рис. 3.3. Дерево поиска для примера 3.2

Рассмотрим теперь особенности применения метода ветвей и границ к задаче о многомерном рюкзаке

$$\sum_{j=1}^{n} c_j x_j \to \max$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i,\ i = 1,...,m,$$

$$x_j \in \{0,1\},\ j = 1,...,n.$$

Исходный алгоритм Лэнд и Дойг предусматривает для получения верхних оценок решение задач линейного программирования, например, симплекс – методом. Иногда удобнее использовать оценку хоть и более грубую, но найти которую можно проще и быстрее.

По правилу Данцига решаются *m* одномерных линейных задач, в качестве ограничения *i*-й задачи выступает *i*-е ограничение исходной задачи. Пусть f_i— оптимальное значение целевой функции для *i*-й задачи. Тогда, поскольку оптимальное решение *x'* задачи линейного программирования с полным набором ограничений удовлетворяет условию $f(x') \le f_i$, $i = 1,...,m$, можно положить

$$\xi(D) = \min f_i .$$

Пусть $\bar{x}^i$ – оптимальное решение *i*–й задачи, x^i – целочисленное решение *i*–й задачи, полученное заменой нулем дробных значений в $\bar{x}^i$. Если некоторые из векторов x^i удовлетворяют всем ограничениям задачи, то лучший из них выступает кандидатом на рекордное решение. В противном случае, допустимое решение *x'* исходной задачи можно получить по формуле

$$x'_j = \prod_{i=1}^{m} x_j^i ,\ j = 1,..., n .$$

Для ветвления выбирается задача с максимальным значением верхней оценки. Ветвление происходит стандартным образом по переменной, принимающей дробное значение при решении хотя бы одной из подзадач. Если переменной с дробным значением нет, то для ветвления выбирается переменная с незафиксированным ранее значением, которой соответствует наибольшее значение коэффициента целевой функции.

Пример 3.3. Решим задачу

$$3x_1 + 12x_2 + 5x_3 + 12x_4 + 11x_5 + 10x_6 + 7x_7 \to \max$$
$$x_1 + 2x_2 + x_3 + 2x_4 + 3x_5 + 3x_6 + 2x_7 \le 6,$$
$$x_1 + 2x_2 + x_3 + 3x_4 + 2x_5 + 2x_6 + 2x_7 \le 8,$$
$$x_j \in \{0,1\}, j=1,\ldots,7.$$

Для первой из одномерных задач

$$\lambda_1^1 = 3,\ \lambda_2^1 = 6,\ \lambda_3^1 = 5,\ \lambda_4^1 = 6,\ \lambda_5^1 = 11/3,\ \lambda_6^1 = 10/3,\ \lambda_7^1 = 7/2,$$

$\bar{x}^1 = (0,1,1,1,1/3,0,0),\ f^1 = 98/3,\ x^1 = (0,1,1,1,0,0,0),\ f(x^1) = 29,$

для второй задачи

$$\lambda_1^2 = 3,\ \lambda_2^2 = 6,\ \lambda_3^2 = 5,\ \lambda_4^2 = 4,\ \lambda_5^2 = 3/2,\ \lambda_6^2 = 5,\ \lambda_7^2 = 7/2,$$

Таким образом, $\xi(D) = 98/3$, вектор x^1 удовлетворяет всем условиям задачи, следовательно, $x' = x^1 = (0,1,1,1,0,0,0)$ – рекордное решение, $f(x') = 29$.

Выполняем ветвление $D = D_1^1 \cup D_2^1$ по дробной переменной x_5. В первой задаче для D_1^1 получаем $\bar{x}^1 = (0,1,1,1,0,0,1/2),\ f^1 = 32{,}5,\ x^1 = (0,1,1,1,0,0,0),$ $f(x^1) = 29$, во второй задаче – $\bar{x}^2 = (0,1,1,1,0,0,1),\ f^2 = 36,\ x^2 = \bar{x}^2$, то есть $\xi(D_1^1) = 32{,}5$. В первой задаче для D_2^1 находим $\bar{x}^1 = (0,1,0,1/2,1,0,0),\ f^1 = 29,$ $x^1 = (0,1,0,0,1,0,0),\ f(x^1) = 23$, во второй задаче – $\bar{x}^2 = (0,1,1,1,1,0,0),\ f^2 = 40,$ $x^2 = \bar{x}^2$ и $\xi(D_2^1) = 29$.

Вершина D_2^1 отсеивается, выполняем ветвление $D_1^1 = D_1^2 \cup D_2^2$ по переменной x_7. В первой задаче для D_1^2 получаем $\bar{x}^1 = (0,1,1,1,0,1/3,0),\ f^1 = 973,$ $x^1 = (0,1,1,1,0,0,0),\ f(x^1) = 29$, во второй задаче – $\bar{x}^2 = (0,1,1,1,0,1,0),\ f^2 = 39,$ $x^2 = \bar{x}^2$, то есть $\xi(D_1^2) = 97/3$. В первой задаче для D_2^2 находим $\bar{x}^1 = (0,1,0,1,0,0,1),\ f^1 = 31,\ x^1 = \bar{x}^1$, во второй задаче – $\bar{x}^2 = (0,1,1,1,0,0,1),$

$f^2 = 36$, $x^2 = \bar{x}^2$. Таким образом, $\xi\left(D_2^2\right) = 31$, $x' = (0,1,0,1,0,0,1)$ – новое рекордное решение, рекорд равен 31, вершина D_2^2 отсеивается.

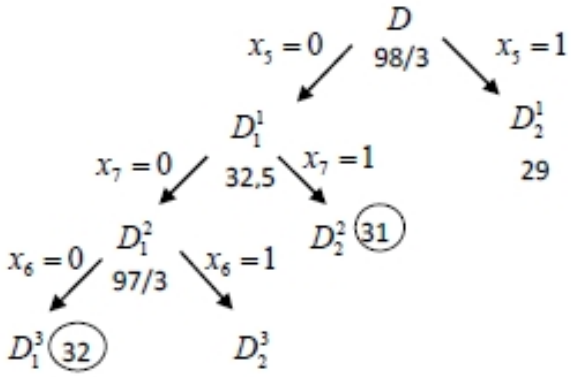

Рис. 3.4. Дерево поиска для примера 3.3

Выполняем ветвление $D_1^2 = D_1^3 \cup D_2^3$ по переменной x_6. В первой задаче для D_1^3 получаем $\bar{x}^1 = (1,1,1,1,0,0,0)$, $f^1 = 32$, $x^1 = \bar{x}^1$, во второй задаче – $\bar{x}^2 = \bar{x}^1$, то есть $(1,1,1,1,0,0,0)$ - новое рекордное решение, $\xi\left(D_1^3\right) = 32$. В первой задаче для D_2^3 находим $\bar{x}^1 = (0,1,0,1/2,0,1,0)$, $f^1 = 28$, и, так как $\xi\left(D_2^2\right) \le f^1$, множество D_2^3 можно отсеять.

Таким образом, $x^* = (1,1,1,1,0,0,0)$ - оптимальное решение задачи, $f\left(x^*\right) = 32$ (рис. 3.4).

3.4 Алгоритм Литтла решения задачи коммивояжера

Рассмотрим классическую задачу коммивояжёра с матрицей *C*. Множество допустимых решений *D* состоит из всех гамильтоновых контуров в соответствующем полном графе. Длина контура $z \in D$ обозначается через $l(z)$. Каждому подмножеству циклов соответствует матрица стоимостей, полученная из исходной. Алгоритм Литтла, Мурти, Суини и Кэрел реализуется по схеме с многосторонним ветвлением. Конкретизируем отдельные элементы схемы.

Нижние границы

Операция вычитания из всех элементов строки (столбца) матрицы минимального элемента этой строки (столбца) называется *приведением* или редукцией матрицы по строке (столбцу). После приведения матрицы стоимостей по всем строкам и столбцам получается неотрицательная матрица, в каждой строке и столбце которой есть по крайней мере один нулевой элемент. Обладающая этими свойствами матрица называется *приведенной* или *редуцированной.*

Пусть α_i, β_j, $i, j = 1,...,n$ – константы приведения матрицы по строкам и столбцам соответственно. (Например, если матрица приводится сначала по

строкам, $\alpha_i = \min\limits_j c_{ij}$, $\beta_j = \min\limits_i (c_{ij} - \alpha_i)$). Тогда элементы приведенной матрицы имеют вид $c'_{ij} = c_{ij} - \alpha_i - \beta_j$, Длина произвольного гамильтонова контура $z = (i_1,...,i_n,i_1)$ по приведенной матрице C' –

$$l'(z) = l(z) - \sum_{i=1}^{n} \alpha_i - \sum_{j=1}^{n} \beta_j ,$$

то есть приведение матрицы издержек не изменяет относительной длины циклов.

Поскольку $l'(z) \geq 0$ (элементы приведенной матрицы неотрицательны),

$$l(z) \geq \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{n} \beta_j .$$

Таким образом, сумма констант приведения дает нижнюю границу на оптимальное значение целевой функции.

Каждой вершине D^k дерева, которое строится в процессе решения, будет соответствовать оценка $\xi(D^k)$, равная сумме констант приведения текущей матрицы издержек. В начале алгоритма

$$\xi(D^0) = \sum_{i}^{n} \alpha_i + \sum_{j}^{n} \beta_j .$$

Поскольку $l(z) = l'(z) + \xi(D^0)$, если можно выбрать по одному нулевому элементу в каждой строке и столбце приведенной матрицы так, чтобы соответствующие дуги образовали единственный цикл, то задача коммивояжера решена и длина кратчайшего маршрута равна $\xi(D^0)$.

Если матрица C не является симметричной, результат приведения и, следовательно, качество оценки может зависеть от того, в каком порядке осуществлялось приведение строк и столбцов. Поэтому целесообразно выбрать тот способ приведения (начинать процесс со строк либо столбцов), который дает большую сумму констант приведения.

Ветвление

Пусть (r,l) – некоторая дуга. Она может либо принадлежать оптимальному циклу коммивояжера, либо не принадлежать ему. Текущее множество маршрутов $D^k \subseteq D$ разбивается на два непересекающихся подмножества, $D^k(r,l)$ и $D^k\overline{(r,l)}$, первое из которых включает все циклы из D^k, содержащие дугу (r,l), второе – циклы, в которых эта дуга отсутствует.

Дуга (*r*,*l*), на основании которой производится ветвление в заданной вершине, выбирается так, чтобы множество $D^k(r,l)$ «с наибольшей вероятностью» содержало оптимальный цикл.

Рассмотрим приведенную матрицу стоимостей C^k, соответствующую вершине D^k. Поскольку в каждой строке и каждом столбце матрицы имеются нулевые элементы, можно предположить, что оптимальный гамильтонов

контур содержит хотя бы одну дугу с нулевой длиной. Длина маршрута, составленного полностью из нулевых дуг приведенной матрицы, равнялась бы, очевидно, $\xi(D^k)$. Пусть $c_{ij}^k = 0$. Рассмотрим цикл *z*, не содержащий дугу (i, j). Тогда в цикл *z* входят дуги (i,t), $t \neq j$ и (s, j), $s \neq i$, поэтому

$$l(z) \geq \xi(D^k) + c_{sj}^k + c_{it}^k.$$

Положим $\theta_{ij} = \min\limits_{s \neq i} c_{sj}^k + \min\limits_{t \neq j} c_{it}^k$. Таким образом, отказ от использования дуги (i, j) приводит к увеличению оценки по крайней мере на θ_{ij}.

Выберем теперь (r,l) так, чтобы циклам, входящим в $D^k\overline{(r,l)}$, соответствовало максимальное гарантированное удлинение пути, то есть

$$\theta_{rl} = \max \theta_{ij},$$

где максимум берется по всем дугам (i, j), для которых $c_{ij}^k = 0$.

Преобразование матрицы расстояний при ветвлении

Пусть имеется разветвление

$$D^k = D^k(r,l) \cup D^k\overline{(r,l)}.$$

Рассмотрим сначала множество $D^k\overline{(r,l)}$ и укажем правило перехода от матрицы C^k к матрице C_2^k. Матрица C_2^k содержит те же строки и столбцы, что и C^k. Поскольку переход из *r* в *l* запрещен, положим $c_{rl}^{(2)} = \infty$ и применим процедуру приведения. Сумма приводящих констант равна θ_{rl}. Таким образом, оценкой для $D^k\overline{(r,l)}$ будет

$$\xi(D^k\overline{(r,l)}) = \xi(D^k) + \theta_{rl}.$$

Обратимся теперь к множеству $D^k(r,l)$ и выясним правило перехода от матрицы C^k к матрице C_1^k. По определению, маршруты из $D^k(r,l)$ заведомо содержит непосредственный переход из *r* в *l*. Поэтому можно, не внося никаких поправок в оценку, вычеркнуть строку *r* и столбец *l*. В результате получим матрицу $\overline{C}_1^k$.

Далее следует запретить возможность возникновения подциклов (т. е. замкнутых маршрутов, содержащих меньше чем *n* вершин), возникающую из-за добавления непосредственного перехода (r,l). Предположим, что из дуг, фиксированных для циклов, принадлежащих $D^k(r,l)$, можно составить путь, который начинается в вершине *p* и кончается в вершине *r*. Чтобы запретить возможный подцикл (*p*,…,*r*,*l*,*p*), следует положить $c_{lp} = \infty$ в матрице $\overline{C}_1^k$. Аналогично, если из фиксированных дуг можно составить путь, начинающийся в вершине *l* и заканчивающийся в вершине *q*, для предотвращения появления цикла (*l*,…,*q*,*r*,*l*) полагаем $c_{qr} = \infty$. Кроме того, $c_{lr} = \infty$, для запрещения цикла (*r*,*l*,*r*), $c_{qp} = \infty$ для запрещения цикла (*p*,…,*r*,*l*,…,*q*,*p*).

Полученную матрицу, если необходимо, приводим по строкам l или q, столбцам r или p (нули во всех других строках и столбцах сохраняются). Пусть сумма констант приведения равна h, тогда

$$\xi(D^k(r,l)) = \xi\left(D^k\right) + h.$$

Если $D^k(r,l)$ состоит ровно из одного цикла (результирующая матрица имеет размер 2x2), то $\xi(D^k(r,l))$ в точности равно длине этого цикла и позволяет обновить сведения о рекорде.

Таким образом, ***общий (k-й) шаг алгоритма метода ветвей и границ*** выглядит следующим образом.

На любом этапе процесса объединения множеств, изображаемых концевыми вершинами, – это множество всех циклов.

Среди неотсеянных по правилам отсева множеств, не подвергавшихся разбиению, выбрать множество D^k с наименьшей нижней оценкой.

В матрице, соответствующей выбранному множеству, выбрать дугу (r,l) для ветвления.

Разбить множество D^k на два подмножества $D^k(r,l)$ и $D^k\overline{(r,l)}$

Преобразовать матрицы, соответствующие этим множествам, пересчитать оценки. Если при пересчете оценки добавление к нижней границе очередной константы приведения ведет к отсеву подзадачи, процедуру приведения можно не доводить до конца.

Модифицировать список множеств, не подвергавшихся разбиению, записав вместо D^k множества $D^k(r,l)$ и $D^k\overline{(r,l)}$ и переобозначая концевые множества: D_1^{k+1}, D_2^{k+1}, …, $D_{r_{k+1}}^{k+1}$.

Если достигнута вершина, содержащая единственный цикл, то откорректировать сведения о рекорде, отсеять множества по правилу (3.7).

Если отсеяны все концевые вершины, рекордное значение является оптимальным. В противном случае перейти к шагу k+1.

Рассмотренный алгоритм можно модифицировать также и для нахождения приближенного решения, взяв в качестве приближенного решения цикл из любого подмножества, состоящего ровно из одного цикла. Сэкономить время при поиске приближенного решения можно с использованием одностороннего ветвления, рассматривая вершины вида $D^k(r,l)$.

Пример 3.4. Решим задачу коммивояжера с матрицей стоимостей

–	34	10	28	41
20	–	22	13	15
9	21	–	32	27
19	8	26	–	14
25	17	11	14	–

Приводим матрицу сначала по строкам (константы 10, 13, 9, 8, 11), затем по последнему столбцу. Сумма констант приведения 53.

–	24	0	18	29
7	–	9	13	0
0	12	–	23	16
11	0	18	–	4
14	6	0	3	–

Оцениваем нулевые элементы приведенной матрицы: $\theta_{13}=18$, $\theta_{24}=3$, $\theta_{25}=4$, $\theta_{31}=19$, $\theta_{42}=10$, $\theta_{53}=3$. Выполняем ветвление по дуге (3,1). Оценка для множества $D^1\overline{(3,1)}$ равна 53+19=72. Чтобы оценить множество $D^1(3,1)$, вычеркиваем третью строку и первый столбец матрицы, блокируем клетку (1,3) и приводим матрицу по строке 1. Получаем $\xi\left(D^1(3,1)\right)=71$ (рис.3.5).

	2	3	4	5
1	6	–	0	11
2	–	9	13	0
4	0	18	–	4
5	6	0	3	–

	2	4	5
1	6	0	–
2	–	13	0
4	0	–	4

	4	5
1	0	–
2	–	0

Рис. 3.5. Матрицы стоимостей для вершин дерева поиска

Выполняем ветвление из вершины $D^1(3,1)$. Оценки нулевых элементов: $\theta_{14}=1$, $\theta_{24}=0$, $\theta_{42}=10$, $\theta_{53}=12$. Дуга для ветвления – (5,3), $\xi\left(D^2\overline{(5,3)}\right)=83$. Вычеркиваем строку 5 и столбец 3, и, так как маршруты множества $D^2(5,3)$ содержат участок 5 – 3 – 1, блокируем клетку (1,5). Так как матрица остается приведенной, $\xi\left(D^2(5,3)\right)=71$.

Выполняем ветвление из вершины $D^2(5,3)$. Оценки нулевых элементов: $\theta_{14}=6$, $\theta_{24}=0$, $\theta_{42}=10$. Дуга для ветвления – (4,2), $\xi\left(D^3\overline{(4,2)}\right)=81$. Матрица для множества $D^3(4,2)$ имеет размер 2x2, поэтому множество $D^3(4,2)$ содержит один маршрут (5,3,1,4,2,5) длиной 71. Остальные вершины дерева отсекаются, найденный маршрут оптимальный (рис. 3.6).

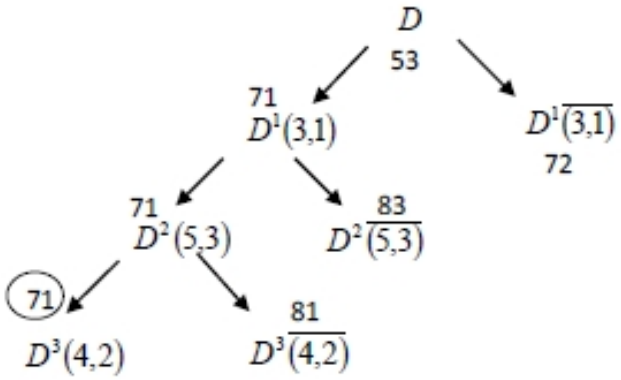

Рис. 3.6. Дерево поиска для примера 3.4

УПРАЖНЕНИЯ

1. Сформулируйте открытую задачу коммивояжера как задачу целочисленного линейного программирования.

2. Определите многогранник X' для примера 2.2.

3. Решите задачу из примера 1.1 с помощью метода отсечений и с помощью метода ветвей и границ.

4. Решите задачу из примера 2.2 методом ветвей и границ.

5. Решите задачу из примера 2.5 методом ветвей и границ.

6. Докажите теорему 3.1

7. Решите задачу целочисленного линейного программирования

$$x_1 - 3x_2 + 3x_3 \to \max$$
$$2x_1 + x_2 - x_3 \le 4$$
$$4x_1 - 3x_2 \le 2$$
$$-3x_1 + 2x_2 + x_3 \le 3$$
$$x_i \ge 0 \text{ и целые, } i = 1,2,3.$$

8. Решите задачу целочисленного линейного программирования

$$-5x_1 - 2x_2 + 3x_3 - 2x_4 - 3x_5 + 3x_6 \to \min$$
$$5x_1 + 6x_2 + 4x_3 + 2x_4 - 3x_5 + 5x_6 = 11$$
$$5x_1 + 5x_2 + 7x_3 + 3x_5 + 5x_6 = 10$$
$$2x_1 + 2x_2 + 2x_3 + 3x_5 = 4$$
$$x_i \ge 0 \text{ и целые, } i = 1,\ldots,6.$$

9. Нефтедобывающая компания изучает вопрос о бурении четырёх нефтяных скважин на двух нефтеносных участках. Издержки подготовки к бурению на каждом участке и затраты на бурение скважины *j* на участке *i* приведены в таблице 4.1. Сформулируйте задачу в виде задачи целочисленного линейного программирования и найдите её оптимальное решение.

Таблица 4.1

Данные к упражнению 9

Участок	Затраты на бурение скважины (млн долл.)				Издержки подготовки к бурению (млн долл.)
	1	2	3	4	
1	2	1	8	5	5
2	4	6	3	1	6

10. Рассматриваются три промышленных участка для размещения обрабатывающих заводов, которые снабжают своей продукцией трех потребителей. Данные об объёмах производства продукции, объёмах потребления и себестоимости (в долл.) перевозки продукции от заводов к потребителям содержатся в таблице 4.2.

В дополнение к транспортным, существуют ещё фиксированные затраты в объёме 10000, 15000 и 12000 долл., связанные с 1, 2 и 3 заводом соответственно. Сформулируйте задачу в виде задачи целочисленного линейного программирования и найдите её оптимальное решение.

Таблица 4.2

Данные к упражнению 10

	1	2	3	Объём производства
1	10	15	12	1800
2	17	14	20	1400
3	15	10	11	1300
Спрос	1200	1700	1600	

11. Пароход может быть использован для перевозки 11 наименований груза, масса, объём и цена единицы каждого из которых приведены в табл. 4.3

Таблица 4.3

Данные к упражнению 11

Параметры единицы груза	Номер груза										
	1	2	3	4	5	6	7	8	9	10	11
Масса (т)	80	62	92	82	90	60	81	83	86	65	83
Объём (м3)	100	90	96	110	120	80	114	60	106	114	86
Цена (усл. ед.)	4,4	2,7	3,2	2,8	2,7	2,8	3,3	3,5	4,7	3,9	4,0

На пароход может быть погружено не более 800 т груза общим объёмом, не превышающим 600 м^3. Определить, сколько единиц каждого груза следует поместить на пароход так, чтобы общая стоимость размещённого груза была максимальной.

12. Из листового проката нужно выкроить заготовки четырёх видов. Один лист длиной 184 см можно разрезать на заготовки длиной 45, 50, 65 и 85 см. Всего заготовок каждого вида необходимо соответственно 90, 96, 88 и 56 шт. Способы разреза одного листа на заготовки и величина отходов при каждом способе приведены в таблице 4.4.

Таблица 4.4

Данные к упражнению 12

Длина заготовки (см)	Количество заготовок, выкраиваемых из одного листа при разрезе способом											
	1	2	3	4	5	6	7	8	9	10	11	12
45	4	2	2	2	1	1	1	1	–	–	–	–
50	–	1	–	–	2	–	1	1	3	2	1	–
65	–	–	1	–	–	2	1	–	–	1	2	1
85	–	–	–	1	–	–	–	1	–	–	–	1
Величина отходов (см)	4	44	29	9	39	9	24	4	34	19	4	34

13. Руководство завода предполагает провести комплекс организационно-технических мероприятий в целях модернизации производства. Затраты производственных площадей, трудовых и финансовых ресурсов, требуемые мероприятиями, отражены в табл. 4.5. На реализацию всех мероприятий завод может выделить трудовых ресурсов 1300 человеко-дней, финансовых – 10 млн у.е., производственных площадей – 700 м^2. Определите мероприятия, которые следует провести, располагая этими ресурсами, с тем чтобы общий экономический эффект был максимальным.

Таблица 4.5

Данные к упражнению 13

Мероприятие	Трудовые ресурсы, человеко-дни	Финансовые ресурсы, тыс. у.е.	Производственные площади, $м^2$	Экономический эффект, тыс. у.е.
Закупка станков с ЧПУ	350	400	130	13000
Текущий ремонт	250	90	–	3000
Монтаж транспортного конвейера	100	60	300	8000
Установка рельсового крана	200	300	150	12000
Ввод системы контроля качества	130	–	150	2500
Разработка АСУП	800	500	100	15000

14. Для реконструкции машиностроительного предприятия было представлено 10 проектов, каждый из которых характеризуется четырьмя агрегированными показателями: затратами труда, энергии, материалов, денежных средств, а также ежегодной прибылью в случае реализации проекта. Соответствующие данные и объём имеющихся ресурсов приведены в таблице 4.6.

Таблица 4.6

Данные к упражнению 14

Проект	1	2	3	4	5	6	7	8	9	10	Ресурсы
Труд, нормо-часы	50	60	30	40	80	70	50	20	40	50	300
Энергия, тыс. кВт·ч	4	4	2	5	5	2	3	6	6	3	24
Материалы, млн руб.	3	2	4	5	3	2	4	2	2	3	20
Денежные средства, млн руб.	7	5	9	6	4	3	7	2	4	5	30
Прибыль, млн руб.	9	8	8,5	8,8	9	8	9	8,7	8,9	8	

При выборе проектов необходимо учесть ряд ограничений технологического характера: одновременно может быть реализовано не более семи проектов; проекты 5 и 8 исключают друг друга; проект 1 может быть реализован лишь при условии реализации проекта 2; проект 4 может быть реализован лишь при условии реализации хотя бы одного из двух проектов: либо проекта 3, либо проекта 10. Какова максимальная прибыль?

15. Менеджер проектов имеет 10 сотрудников, которые работают над шестью проектами, причём каждый работает одновременно над несколькими проектами, как показано в таблице 4.7.

Раз в неделю менеджер должен встретиться с каждой группой сотрудников, работающих над одним и тем же проектом. Планируется составить график обсуждения проектов таким образом, чтобы уменьшить движение в офисе, то есть сократить число сотрудников, входящих и выходящих из комнаты для совещаний. В какой последовательности необходимо рассматривать проекты?

Таблица 4.7

Данные к упражнению 15

		Проект					
Сотрудники		1	2	3	4	5	6
	1		x		x	x	
	2	x		x		x	
	3		x	x	x		x
	4			x	x	x	
	5	x	x	x			
	6	x	x	x	x		x
	7	x	x			x	x
	8	x		x	x		
	9					x	x
	10	x	x		x	x	x

16. Решить задачу коммивояжера с матрицей стоимостей

–	43	21	20	14	24	17
10	–	9	22	15	8	23
20	10	–	5	15	25	15
42	50	27	–	33	45	30
30	17	15	7	–	21	27
25	31	36	19	24	–	38
36	26	18	11	21	13	–

ЛИТЕРАТУРА

1. Акулич И. Л. Математическое программирование в примерах и задачах. – М.: Высш. шк., 1986. – 319 с.
2. Афанасьев М.Ю., Суворов Б.П. Исследование операций в экономике: модели, задачи, решения. – М.: ИНФРА-М, 2003. – 444с.
3. Ашманов С.А. Линейное программирование. – М.: Наука, 1981. – 340 с.
4. Зайченко Ю.П., Шумилова С.А. Исследование операций: Сборник задач. – К.: Выща шк., 1990. – 239 с.
5. Ковалев М.М. Дискретная оптимизация (целочисленное программирование). – М.: Едиториал УРСС, 2003. – 192 с.
6. Конюховский П.В. Математические методы исследования операций в экономике. – СПб.: Питер, 2002. – 208 с.
7. Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование. – М.: Наука, 1969. – 368 с.
8. Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.
9. Мину М. Математическое программирование. Теория и алгоритмы. – М.: Наука, 1990. – 488 с.
10. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. – М.: Мир, 1985. – 512 с.
11. Сигал И.Х., Иванова А.П. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. – М.: ФИЗМАТЛИТ, 2007. – 304 с.
12. Таха Х. Введение в исследование операций. – М.: Издательский дом «Вильямс», 2005. – 912 с.
13. Ху Т. Целочисленное программирование и потоки в сетях: Пер. с англ. – М.: Мир, 1974. – 516 с.
14. Ху Т.Ч., Шинг М.Т. Комбинаторные алгоритмы. – Нижний Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2004. – 330 с.
15. Bramel J., Simchi-Levi D. The logic of logistics: theory, algorithms, and applications for logistics management. – Springer-Verlag New York, 1997.

Алла Александровна **Тюхтина**

Методы дискретной оптимизации

Часть 1

Учебно-методическое пособие

Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Нижегородский государственный университет им. Н.И. Лобачевского».
603950, Нижний Новгород, пр. Гагарина, 23.