Исследование операций

Курс лекций подготовлен профессором кафедры экономической информатики и математической экономики

Ковалевым Михаилом Яковлевичем.

Рассчитан на студентов 2 курса экономических специальностей.
Продолжительность лекций – 34 часа (17 занятий). Такое же количество практических занятий.

Минск

2004

Содержание

1 Литература **3**

2 Основные понятия исследования операций (ИСО). Классификация задач **4**
2.1 Классификация задач ИСО 5
2.2 Многокритериальные задачи 6

3 Основы линейного программирования **8**
3.1 Графический метод решения задач ЛП 10
3.2 Возможные варианты решения задач ЛП 12
3.3 Эквивалентные постановки задач ЛП 12
3.4 Симплекс-метод 15
3.5 Транспортная задача ЛП 17
3.6 Задача о назначениях 21

4 Основы теории графов **23**
4.1 Основные понятия 23
4.2 Алгоритм построения эйлерова цикла для неориентированного графа 25
4.3 Задание графов 25
4.4 Топологическая сортировка 26
4.5 Остовное дерево (остов) минимального веса 26
4.6 Кратчайшие пути 27

5 Сетевое планирование и управление проектами **29**

6 Задача коммивояжера и метод ветвей и границ **31**

7 Имитационное моделирование на примере метода Монте-Карло **37**

8 Основы теории сложности вычислений **39**

9 Динамическое программирование **42**

1 Литература

Основная:

1. Вентцель Е.С. Исследование операций. - М., Советское радио, 1972. - 552 с.

2. Волков И.К., Загоруйко Е.А. Исследование операций: учебное пособие для вузов. 2-е изд. / Под ред. В.С. Зарубина, А.П. Крищенко. - М. Изд-во МГТУ им. Н.Э. Баумана, 2002. - 436 с.

3. Taha H. Operations research: an introduction. - Prentice Hall, Upper Saddle River, New Jersey, 1977.

4. М.М. Кавалеу, М.М. Пісарук. Сучаснае лінейнае праграмаванне. - Мінск, Выдавецкі цэнтр Белдзяржуніверсітэта, 1998. - 260 с.

Дополнительная:

5. Вентцель Е.С. Исследование операций. - М., Знание, 1976.

6. Вентцель Е.С. Исследование операций. Задачи, принципы, методология. - М., Наука, 1988.

7. Соболь И.С. Метод Монте-Карло. - М., Наука, 1972.

8. Nahmias S. Production and operations analysis. 3rd edition. - The McGraw-Hill Companies, Inc., 1997. - 858 pp.

9. Dilworth J.B. Production and operations management: manufacturing and services. 5th edition. - Mcgraw-Hill, Inc., 1993. - 742 pp.

10. Ritzman L.P., Krajewski L.J. Foundations of operations management. - Prentice Hall, Upper Saddle River, NJ, 2003. - 473 pp.

11. Танаев В.С., Гордон В.С., Шафранский Я.М. Теория расписаний. Одностадийные системы. – М.: Наука, 1984.

12. Танаев В.С., Сотсков Ю.Н., Струсевич В.А. Теория расписаний. Многостадийные системы. – М.: Наука, 1987.

13. Танаев В.С., Ковалев М.Я., Шафранский Я.М. Теория расписаний. Групповые технологии. – Минск: ИТК НАН Беларуси, 1998.

14. Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. – Минск: Университетское, 1995.

2 Основные понятия исследования операций (ИСО). Классификация задач

Исследование операций – это комплексная математическая дисциплина, занимающаяся построением, анализом и применением математических моделей принятия *оптимальных решений* при проведении *операций*.

Операция – система *управляемых* действий, объединенная единым замыслом и направленная на достижение определенной цели.

Примеры операций.

Пример 1. Предприятие выпускает несколько видов изделий, при изготовлении которых используются ограниченные ресурсы различного типа. Требуется составить план выпуска изделий на месяц, т.е. указать количество выпускаемых изделий каждого вида, так, чтобы максимизировать прибыль при выполнении ограничений на потребляемые ресурсы.

Пример 2. Требуется создать сеть временных торговых точек так, чтобы обеспечить максимальную эффективность продаж. Для этого требуется определить
– число точек,
– их размещение,
– количество персонала и их зарплату,
– цены на товары.

Пример 3. Требуется организовать строительство железнодорожного вокзала. При этом необходимо указать порядок выполнения работ во времени и распределить требуемые ресурсы между работами так, чтобы завершить строительство во время и минимизировать его стоимость.

Набор управляющих параметров (переменных) при проведении операции называется **решением**. Решение называется **допустимым**, если оно удовлетворяет набору определенных условий. Решение называется **оптимальным**, если оно допустимо и, по определенным признакам, предпочтительнее других, или, по крайней мере, не хуже.

Признак предпочтения называется **критерием оптимальности**. Критерий оптимальности включает в себя *целевую функцию* и *направление оптимизации* или набор целевых функций и соответствующих направлений оптимизации.

Целевая функция – это количественный показатель предпочтительности или эффективности решений.

Направление оптимизации – это *максимум (минимум)*, если наиболее предпочтительным является наибольшее (наименьшее) значение целевой функции. Например, критерием может быть максимизация прибыли либо минимизация расходов.

Математическая модель задачи ИСО включает в себя:

1) описание переменных, которые необходимо найти,

2) описание критериев оптимальности,

3) описание множества допустимых решений (ограничений, накладываемых на переменные).

Цель ИСО – количественно и качественно обосновать принимаемое решение. Окончательное решение принимает ответственное лицо (либо группа лиц), называемое **лицо, принимающее решение (ЛПР)**.

Математическая модель задачи ИСО составляется в соответствии с представлениями ЛПР об этой задаче, т.е. в соответствии с его *информационным состоянием*. При этом важно, чтобы математическая модель задачи была наиболее *адекватной*, т.е. наиболее правильно

отражала информационное состояние ЛПР. Для этого разработчик математической модели должен работать в тесном контакте с ЛПР.
Основной принцип разработчика: *"Разрабатывай не то, что заказчик просит, а то, что ему нужно."* (М. Гэри и Д. Джонсон "Вычислительные машины и труднорешаемые задачи")
Проверка адекватности представлений ЛПР о задаче не является предметом ИСО. Изменение информационного состояния ЛПР может привести к изменению математической модели задачи.

2.1 Классификация задач ИСО

Классификация по зависимости параметров задачи от времени.
1. Статическая задача. Принятие решения происходит при условии, что все параметры задачи заранее известны и не изменяются во времени. Процедура принятия решения осуществляется один раз.
2. Динамическая задача. В процессе принятия решения параметры задачи изменяются во времени. Процедура принятия решения осуществляется поэтапно и может быть представлена в виде процесса, зависящего от времени, в том числе непрерывно. Пример – навигационная задача.
Классификация в зависимости от достоверности информации о задаче.
1. Детерминированная задача. Все параметры задачи заранее известны. Для решения детерминированных задач в основном применяются методы *математического программирования.*
2. Недетерминированная задача. Не все параметры задачи заранее известны. Например, необходимо принять решение об управлении устройством, некоторые узлы которого могут непредсказуемо выходить из строя. Оптимальное решение недетерминированной задачи ИСО отыскать практически невозможно. Однако некоторое "разумное" решение отыскать можно.
"Исследование операций представляет собой искусство давать плохие ответы на те практические вопросы, на которые даются еще худшие ответы другими методами". (Т.Л. Саати)
2,а). Стохастическая задача. Не все параметры задачи заранее известны, но имеются статистические данные о неизвестных параметрах (вероятности, функции распределения, математические ожидания и т.д.).
Для отыскания оптимального решения стохастической задачи применяется один из следующих приемов:
– искусственное сведение к детерминированной задаче (неизвестные параметры заменяются их средними значениями),
– "оптимизация в среднем" (вводится и оптимизируется некоторый статистический критерий).
2,б). Задача в условиях (полной) неопределенности. Статистические данные о неизвестных параметрах отсутствуют. Задачи ИСО в условиях неопределенности в основном изучаются в рамках *теории игр.*

Классификация по виду критерия оптимальности.
Критерий оптимальности может иметь любой вид, в том числе *неформализуемый.* Наиболее распространенные *формализуемые* критерии оптимальности заключаются в оптимизации (минимизации либо максимизации) одной либо нескольких *скалярных* целевых функций.
Функция называется скалярной, если ее значением является некоторое число. Задача оптимизации скалярной функции на заданном множестве допустимых числовых решений называется *задачей математического программирования.* Наиболее изученными представителями однокритериальных задач математического программирования, т.е. задач с одной целевой функцией, являются следующие задачи.
Задачи линейного программирования. Целевая функция – линейная, множество допустимых решений – выпуклый многогранник.
Задачи квадратичного программирования. Целевая функция – квадратичная ($\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_ix_j$), а множество допустимых решений – выпуклый многогранник.
Задачи стохастического программирования. Это задачи линейного программирования с неизвестными числовыми параметрами, о которых имеются статистические данные.
Задачи дискретного программирования. Множество допустимых решений – дискретное множество.
Задачи целочисленного программирования. Множество допустимых решений – точки целочисленной решетки.
Задачи булева программирования. Множество допустимых решений – 0-1 матрицы.

2.2 Многокритериальные задачи

В задачах ИСО, как правило, присутствует не один, а несколько признаков предпочтения (критериев). Такие задачи называются многокритериальными.
Критерии могут оказаться противоречивыми, т.е. решение, лучшее по определенному признаку, может оказаться худшим по другому признаку. Например, минимизация стоимости и максимизация качества товара почти всегда противоречивы. В этом случае задача отыскания решения, предпочтительного по всем признакам, будет *некорректной*, т.е. не будет иметь ни одного решения.
В случае противоречивых критериев, ИСО предлагает следующие подходы к отысканию подходящего решения.
1) Замена некоторых критериев ограничениями вида $\leq$ или $\geq$. Например, минимизация стоимости $f(x) \to \min$, может быть заменена ограничением вида $f(x) \leq A$, где A – некоторая верхняя оценка стоимости, т.е. максимально допустимая стоимость.
2) Свертка критериев. Создается один глобальный скалярный критерий, целевая функция которого является некоторой функцией от исходных целевых функций. Наиболее употребимыми являются линейные свертки вида $\alpha f(x) + \beta g(x)$ (в случае двух критериев). Нетривиальной является задача отыскания адекватных значений коэффициентов α и β, отражающих относительную важность целевых функций $f(x)$ и $g(x)$.
3) Ранжирование критериев. Критерии ранжируются по степени важности.
4) Отыскание решений, лучших хотя бы по одному критерию.

Подходы 1) и 2) приводят к однокритериальной задаче. Подход 3) приводит к *задаче с упорядоченными критериями.* Подход 4) приводит к *задаче с независимыми критериями.* В задаче с упорядоченными критериями критерии упорядочиваются по важности и требуется найти оптимальное решение для наименее важного критерия на множестве решений, оптимальных для более важного критерия (см. рисунок). Самое большое –

множество всех допустимых решений, в него вложено множество решений, оптимальных по самому важному критерию, далее вложено множество оптимальных решений по второму по важности критерию, и т.д.

В задаче с независимыми критериями требуется найти множество *недоминируемых (эффективных) решений.* Недоминируемое решение лучше любого другого допустимого решения хотя бы по одному критерию либо не хуже по всем критериям. Множество недоминируемых решений также называется *множеством Парето.*

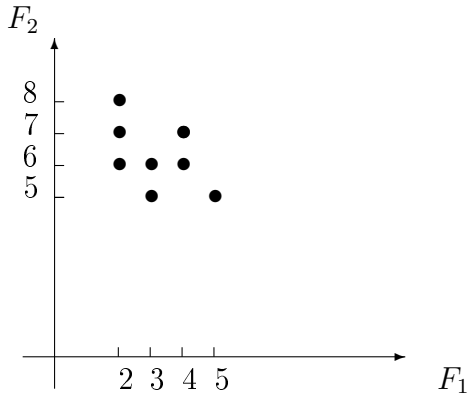
Пример многокритериальной задачи с независимыми критериями. Фирма по разработке программного обеспечения должна выполнить проекты 1 и 2 в последовательности 1,2. Для выполнения каждого проекта можно привлечь одного или двух исполнителей. Пусть x_1 и x_2 – число исполнителей, привлеченных для выполнения проектов 1 и 2 соответственно. Время выполнения проекта i равно $p_i(x_i)$ месяцев, а соответствующая стоимость – $c_i(x_i)$ млн. рублей. Требуется минимизировать общее время выполнения проектов при минимальной стоимости.

Значения функций заданы следующим образом:

x	1	2	3
$p_1(x)$	2	1	1
$p_2(x)$	3	1	1
$c_1(x)$	1	2	3
$c_2(x)$	4	4	5

Общее время выполнения проектов равно $F_1(x_1, x_2) = p_1(x_1) + p_2(x_2)$, а стоимость их выполнения равна $F_2(x_1, x_2) = c_1(x_1) + c_2(x_2)$.

Определим все возможные значения пар $(F_1, F_2) = (F_1(x_1, x_2), F_2(x_1, x_2)) \in \{(2,6), (2,7), (2,8), (3,5), (3,6), (4,6), (4,7), (5,5)\}$, см. рис. Задача отыскания множества Парето в случае двух критериев вида $F_1(x) \to \min$ и $F_2(x) \to \min$ может быть решена графически следующим образом. Находим все точки с наименьшим значением $F_1(x)$.



Если их несколько, выбираем из них точку с наименьшим значением $F_2(x)$. Включаем ее в множество Парето. Отсекаем точки с большими либо равными значениями $F_1(x)$ и $F_2(x)$ (северо-восточный угол с вершиной в выбранной точке). Повторяем процедуру для оставшейся части допустимой области.

Из рисунка видно, что для нас представляют интерес пары $(F_1, F_2) \in \{(2,6),(3,5)\}$ и соответствующие решения $(x_1, x_2) \in \{(2,2),(1,2)\}$, которые являются недоминируемыми и образуют множество Парето рассматриваемой задачи.

3 Основы линейного программирования

Задачи линейного программирования (ЛП) и методы их решения широко используются в промышленном производстве, экономике, логистике, военном деле и других областях целенаправленной деятельности человека. Напомним, что задача ИСО называется задачей ЛП, если множество ее допустимых решений – выпуклый многогранник, а целевая функция – скалярная линейная.

Примеры задач ЛП.

Макроэкономические линейные модели

Основой большинства макроэкономических моделей является *модель межотраслевого баланса*, разработанная и изученная американским ученым Василием Леонтьевым в 1920-х годах. Модель получила название автора, а сам автор получил за ее разработку Нобелевскую премию в области экономики.

Модель Леонтьева. Предположим, что весь производственный сектор страны (или любой другой крупной экономической системы) разделен на n отраслей. Предполагается, что каждая отрасль выпускает однородную продукцию (продукцию одного типа), но разные отрасли выпускают разную продукцию. Таким образом, в целом выпускается n типов продукции. В процессе производства каждая отрасль использует продукцию других отраслей и, в свою очередь, снабжает другие отрасли своей продукцией. Кроме того, каждая отрасль выпускает продукцию, которая потребляется в непроизводственной сфере (сфере потребления, создание запасов).

Допустим, что для отрасли i известно, что a_{ij} единиц продукции этой отрасли используется для производства единицы (!) продукции отрасли j и c_i единиц продукции этой отрасли используется в непроизводственной сфере (накопление, потребление).

Величина c_i также называется *внешним спросом*. Требуется определить x_i – валовый объем (количество единиц) продукции отрасли i, $i = 1, \ldots, n$, так, чтобы выполнялось **условие межотраслевого баланса:**

$$\sum_{j=1}^{n} a_{ij}x_j + c_i = x_i,\ x_i \geq 0,\ i = 1, \ldots, n,$$

или в матричной форме

$$x - Ax = c,\ x \geq 0.$$

Приведенные соотношения называются моделью Леонтьева. Соотношение $x - Ax = c$ можно ослабить: $x - Ax \geq c$ (в этом случае c определяет нижнюю границу внешнего спроса). С моделью Леонтьева связаны некоторые задачи оптимизации.

1) Задача максимизации суммарного валового выпуска при ограниченных трудовых ресурсах.

Пусть для выпуска единицы продукции отрасли j требуется t_j единиц трудовых ресурсов. Обозначим $t = (t_1, \ldots, t_n)$. Пусть T – общее количество трудовых ресурсов, имеющееся в наличии. Тогда модель задачи максимизации суммарного валового выпуска при ограниченных трудовых ресурсах состоит в следующем:

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

$$\begin{cases} x - Ax \geq c \\ \sum_{j=1}^{n} t_j x_j \leq T \\ x \geq 0 \end{cases}$$

2) Задача минимизации требуемых трудовых ресурсов при заданном уровне суммарного валового выпуска:

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

$$\begin{cases} x - Ax \geq c \\ \sum_{j=1}^{n} x_j \geq V \\ x \geq 0, \end{cases}$$

где V – заданный уровень суммарного валового выпуска.

Микроэкономические линейные модели

Модель оптимальной программы предприятия. Предприятие выпускает n видов продукции. Для выпуска единицы продукции вида j требуется a_{ij} единиц ресурса i, который имеется в объеме b_i, $i = 1, \ldots, m$ (всего имеется m видов ресурсов). Выпуск единицы продукции вида j приносит прибыль в объеме p_j. Требуется определить программу выпуска продукции $x = (x_1, \ldots, x_n)$, где x_j – количество выпускаемой продукции вида j, такую, что выполнены ограничения на ресурсы и суммарная прибыль максимальна. Соответствующая задача ЛП имеет вид:

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

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j \le b_i, \ i = 1, \dots, m, \\ x_j \ge 0, \ j = 1, \dots, n \end{cases}$$

Модель распределения ресурсов. Имеется n видов деятельности и m видов ресурсов. Ресурс i имеется в объеме b_i, $i = 1, \dots, m$. С помощью единицы i-го ресурса можно выполнить d_{ij}-ю часть j-го вида деятельности и при этом затраты будут составлять c_{ij}. Введем переменные x_{ij}, показывающие какое количество ресурса i направлено для вида деятельности j. Задача состоит в следующем.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min,$$

$$\begin{cases} \sum_{i=1}^{m} d_{ij}x_{ij} = 1, \ j = 1, \dots, n, \\ \sum_{j=1}^{n} x_{ij} \le b_i, \ i = 1, \dots, m, \\ x_{ij} \ge 0, \ i = 1, \dots, m, \ j = 1, \dots, n. \end{cases}$$

Модель оптимальной купли-продажи валюты. Рассмотрим ситуацию, когда брокер осуществляет куплю-продажу валюты с целью получения прибыли за счет разницы в курсах валют. Пусть n – количество доступных валютных рынков, m – количество операций купли-продажи. Обозначим через r_{ij} количество денежных единиц i-го валютного рынка, которые можно купить ($r_{ij} > 0$) либо продать ($r_{ij} < 0$) в результате операции j *стандартного* объема. Например, при проведении операции номер 2 стандартного объема, продавая 2 доллара, можно купить 1 евро и 2000 бел.рублей. Пусть x_j – объем операции j. В этом случае соответствующие величины r_{ij} умножаются на x_j. Например, продавая $2x_2$ долларов, можно получить x_2 евро и $2000x_2$ бел.рублей.

Рассмотрим идеализированную ситуацию, когда все операции выполняются одновременно. Ограничение состоит в том, что количество проданных денежных единиц не должно превосходить количества купленных для каждого вида валюты. Задача состоит в отыскании таких объемов операций, что количество денежных единиц по одному из типов валют, например, первому, максимально. Соответствующая математическая модель выглядит следующим образом.

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

$$\sum_{j=1}^{n} r_{ij}x_j \ge 0, \ i = 1, \dots, m.$$

3.1 Графический метод решения задач ЛП

Вначале рассмотрим наиболее простой метод решения, применимый для случая, когда количество переменных не превосходит двух. Рассмотрим следующий пример.

Компания производит погрузчики и тележки. От одного погрузчика компания получает доход в размере \$80 и от одной тележки в размере \$40. Имеется три обрабатывающих центра, на которых выполняются операции металлообработки, сварки и сборки, необходимые для производства любого из продуктов. Для интервала планирования, равного месяцу, задана предельная производственная мощность каждого обрабатывающего центра в часах, а также количество часов, необходимое на этом центре для производства одного погрузчика и одной тележки. Эта информация задана в таблице.

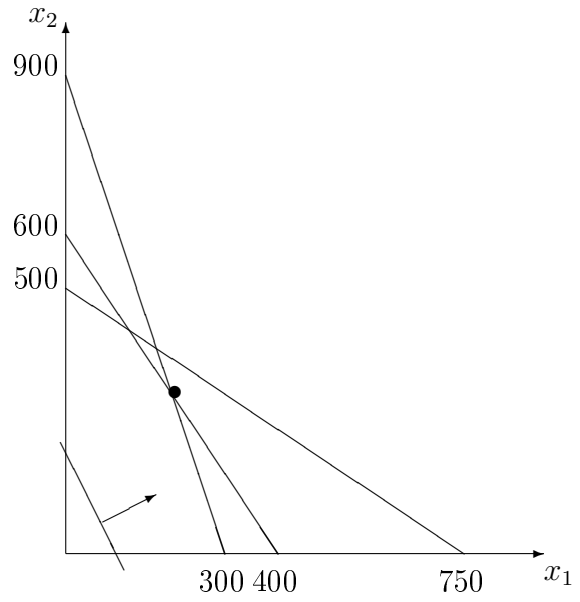
Центр/Изделие	Погрузчик (часы на ед.)	Тележка (часы на ед.)	Общая мощность (часы)
Мет.обработка	6	4	2400
Сварка	2	3	1500
Сборка	9	3	2700

Требуется составить допустимый план работ на месяц с максимальным доходом. Математическая модель задачи может быть записана следующим образом.

$$80x_1 + 40x_2 \to \max,$$

$$\begin{cases} 6x_1 + 4x_2 \le 2400 \\ 2x_1 + 3x_2 \le 1500 \\ 9x_1 + 3x_2 \le 2700 \\ x_1 \ge 0,\ x_2 \ge 0 \end{cases}$$

где x_1 и x_2 – количества производимых погрузчиков и тележек соответственно.
Представим ограничения в виде следующего графика, где по осям $(0x_1)$ и $(0x_2)$ откладывается количество произведенных погрузчиков и тележек соответственно, а линии представляют ограничения на производственные мощности. Допустимая область задачи представлена в виде многоугольника.



Представим целевую функцию $80x_1 + 40x_2 = z$ с переменным значением z прерывистой линией. Любая точка (x_1, x_2) на линии $80x_1 + 40x_2 = z$ соответствует доходу в размере z. Перемещая ее параллельно себе самой, получаем разные значения дохода.
Выразим x_2 через x_1 и $z : x_2 = z/40 - 2x_1$. При $x_1 = 0$ значение $x_2 = z/40$. Следовательно, значение z увеличивается при увеличении x_2, т.е. при перемещении целевой функции вверх.
Нетрудно заметить, что максимальный доход среди допустимых точек достигается в одной из вершин многоугольника ограничений. В данном случае в точке пересечения линий, соответствующих ограничениям на металлообработку и сборку, $x_1^* = 200$, $x_2^* = 300$.

Для определения точки, соответствующей оптимальному решению, нужно сравнить углы наклона прямых, соответствующих целевой функции и ограничению. Для нахождения угла наклона целевой функции можно положить $z = 0$, так как константа не влияет на угол наклона, и выразить x_2 через $x_1 : x_2 = -2x_1$. Затем рассмотреть точку $x_1 = 0$, $x_2 = 0$ и $x_1 = 1$, $x_2 = -2$. Если ограничение $2x_1 + 3x_2 \leq 1500$, то рассмотрим $2x_1 + 3x_2 = 0$, выразим $x_2 = -3/2x_1$ и найдем точки $x_1 = 0$, $x_2 = 0$ и $x_1 = 1$, $x_2 = -3/2$. Из рисунка видно, что в данном случае целевая функция более "крутая".

3.2 Возможные варианты решения задач ЛП

1) ЗЛП имеет единственное решение
2) Не существует допустимого решения ЗЛП
3) ЗЛП имеет ∞ много оптимальных решений, которые называются *альтернативными*
4) Целевая функция ЗЛП неограничена (в допустимой области).

3.3 Эквивалентные постановки задач ЛП

Общая постановка:

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

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i, i \in I_1 \\ \sum_{j=1}^{n} a_{ij} x_j = b_i, i \in I_2 \\ \sum_{j=1}^{n} a_{ij} x_j \geq b_i, i \in I_3 \\ x_j \geq 0, j \in N_1, \\ x_j \in R, j \in N_2, \\ x_j \leq 0, j \in N_3, \end{cases}$$

где c_j, a_{ij}, b_i – заданные числовые параметры, множества I_1, I_2 и I_3 попарно не пересекаются, $I_1 \cup I_2 \cup I_3 = \{1, \ldots, m\}$ и $N_1 \cup N_2 \cup N_3 = \{1, \ldots, n\}$. Существует несколько эквивалентных постановок задач ЛП:

– Задачи максимизации и минимизации. Для перехода от одной задачи к другой коэффициенты целевой функции можно умножить на -1.

– Задачи с ограничениями типа "=", "$\leq$" и "$\geq$". Для перехода от ограничения "$\leq$" к ограничению "$\geq$" и наоборот достаточно домножать соответствующее ограничение на -1. Ограничение типа "=" может быть представлено в виде двух ограничений "$\leq$" и "$\geq$" с той же левой и правой частью. Для перехода от ограничений типа "=" к ограничениям типа $\leq$ можно применить метод Гаусса. Например, рассмотрим задачу

$$-x_1 - x_2 + x_5 \to \min$$

$$\begin{cases} x_1 + x_2 = 1 \\ x_2 - 2x_3 = -3 \\ x_3 - x_4 + x_5 = 1 \\ x_j \geq 0 \end{cases}$$

При помощи метода Гаусса выделяем единичную подматрицу в матрице ограничений.

$$\left|\begin{array}{ccccc|c} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & -2 & 0 & 0 & -3 \\ 0 & 0 & 1 & -1 & 1 & 1 \end{array}\right| \Rightarrow \left|\begin{array}{ccccc|c} 1 & 0 & 2 & 0 & 0 & 4 \\ 0 & 1 & -2 & 0 & 0 & -3 \\ 0 & 0 & 1 & -1 & 1 & 1 \end{array}\right| \Rightarrow \left|\begin{array}{ccccc|c} 1 & 0 & 0 & 2 & -2 & 2 \\ 0 & 1 & 0 & -2 & 2 & -1 \\ 0 & 0 & 1 & -1 & 1 & 1 \end{array}\right|$$

$$\begin{cases} x_1 = 2 - (2x_4 - 2x_5) \geq 0 \\ x_2 = -1 - (-2x_4 + 2x_5) \geq 0 \\ x_3 = 1 - (-x_4 + x_5) \geq 0 \end{cases}$$

Эквивалентная задача

$$-2 + 2x_4 - 2x_5 - 1 + 2x_4 - 2x_5 + x_5 = -3 + 4x_4 - 3x_5 \to \min$$

$$\begin{cases} 2x_4 - 2x_5 \leq 2 \\ -2x_4 + 2x_5 \leq -1 \\ -x_4 + x_5 \leq 1 \\ x_4, x_5 \geq 0 \end{cases}$$

Для перехода от ограничений типа $\leq$ к ограничениям типа $=$ достаточно ввести дополнительные неотрицательные переменные, по одной для каждого ограничения $\leq$. Эти переменные войдут в целевую функцию с нулевыми коэффициентами.

Переменная $x_j \in R$ может быть представлена в виде двух новых неотрицательных переменных: $x_j = x_j^+ - x_j^-$, $x_j^+ \geq 0$, $x_j^- \geq 0$.

Для каждой задачи ЛП можно сформулировать *двойственную ей* задачу. Рассмотрим задачу максимизации с ограничениями "$\leq$". Представим эту задачу в матричной форме:

$$cx \to \max,$$

$$Ax \leq b,$$

где $c = (c_1, \ldots, c_n)$ – вектор-строка, $x = (x_1, \ldots, x_n)^T$ – вектор-столбец, $A = ||a_{ij}||$ – матрица размерности $m \times n$, $b = (b_1, \ldots, b_m)^T$ – вектор-столбец. Отметим, что ограничения $x \geq 0$ отсутствуют. Они могут быть войти в ограничения $Ax \leq b$ путем введения эквивалентных ограничений $-x \leq 0$. Введем в рассмотрение *двойственную ей задачу*:

$$yb \to \min,$$

$$yA = c, \ y \geq 0,$$

где $y = (y_1, \ldots, y_m)$ – вектор-строка *двойственных переменных* или *потенциалов*.

Двойственные задачи могут быть сформулированы не только для задачи $cx \to \max$, $Ax \leq b$, но и для других форм задач.

Пусть A_i обозначает строку i матрицы A, а A^j – столбец j этой же матрицы. Пусть R – множество всех действительных чисел, как положительных так и отрицательных.

Соответствие между компонентами прямой и двойственной задач представлено в следующей таблице.

Прямая задача	Двойственная задача
$\max\{cx\}$	$\min\{yb\}$
$A_i x \leq b_i, \ i \in I_1$	$y_i \geq 0, \ i \in I_1$
$A_i x = b_i, \ i \in I_2$	$y_i \in R, \ i \in I_2$
$x_j \geq 0, \ j \in N_1$	$yA^j \geq c_j, \ j \in N_1$
$x_j \in R, \ j \in N_2$	$yA^j = c_j, \ j \in N_2$
$x_j \leq 0, \ j \in N_3$	$yA^j \leq c_j, \ j \in N_3$

Теорема 1 *(Теорема двойственности) Справедливо одно из следующих утверждений:*
а) обе задачи, прямая и двойственная ей, имеют допустимые решения и

$$\max\{cx\} = \min\{yb\},$$

б) если одна из задач, прямая и двойственная ей, не имеет допустимого решения а другая имеет, то целевая функция другой задачи не ограничена.
в) обе задачи не имеют допустимых решений.

Теорема 2 *(О дополняющей нежесткости) Пусть x и y – допустимые решения прямой и двойственной задач соответственно. Тогда следующие условия а), б) и в) эквивалентны:*
а) x и y оптимальные решения прямой и двойственной задач,
б) $cx = yb$,
в) (условие дополняющей нежесткости) $y_i(A_i x - b_i) = 0$ $i = 1, \ldots, m$, и $(yA^j - c_j)x_j = 0$, $j = 1, \ldots, n$.

Из этой теоремы, например, следует, что если решения прямой и двойственной задач допустимы и выполняется условие в) дополняющей нежесткости, то эти решения оптимальны.
Отметим, что понятие двойственности симметрично, т.е. неважно какую из пары двойственных задач назвать прямой, а какую двойственной. Если одна из них названа прямой, то вторая будет двойственной.
Оптимальное решение прямой задачи может быть получено из оптимального решения двойственной задачи. Например, рассмотрим задачу
$4x_1 - 3x_2 - 6x_3 - x_4 \to \max,$
$$\begin{cases} -2x_1 - x_2 + 5x_3 + 2x_4 \geq 2 \\ -3x_1 + x_2 - x_3 - x_4 \geq 1 \\ x_i \geq 0,\ i = 1, 2, 3, 4 \end{cases}$$
Построим двойственную ей задачу
$2y_1 + y_2 \to \min,$
$$\begin{cases} -2y_1 - 3y_2 \geq 4 \\ -y_1 + y_2 \geq -3 \\ 5y_1 - y_2 \geq -6 \\ 2y_1 - y_2 \geq -1 \\ y_j \geq 0,\ j = 1, 2 \end{cases}$$
Решим двойственную задачу графически: $y^* = (-9/4, -21/4)$, $z^* = -39/4$. Оптимальное решение определяется пересечением линий, соответствующих ограничению 2 и 3 двойственной задачи. Эти ограничения выполняются как строгие равенства, а остальные - как нестрогие. Тогда по условию дополняющей нежесткости должно выполняться $x_1^* = 0$, $x_4^* = 0$, x_2^* и x_3^* могут быть найдены из равенств $-x_2^* + 5x_3^* = 2$ и $x_2^* - x_3^* = 1$ (рассматриваем подматрицу матрицы прямой задачи, состоящую из столбцов 2 и 3). Получаем $x^* = (0, 7/4, 3/4, 0)$.

3.4 Симплекс-метод

Симплекс-метод является методом решения задач ЛП. Он впервые использовался лауреатом Нобелевской премии советским ученым Канторовичем в 1939 г. и описан в том виде, в котором существует сейчас, Данцигом в 1947 г.
Описание метода проведем для задачи ЛП в *стандартной* форме:

$$z = cx \to \max,$$

$$Ax = b, x \geq 0.$$

Также должно выполняться $b \geq 0$.
Рассмотрим следующий пример. Фирма производит краску для наружных и внутренних работ из исходных материалов M1 и M2. Расход материалов и их запасы представлены в таблице.

Исх.мат./Краска	Для наруж. раб. (тонн на 1 тонну)	Для внутр. раб. (тонн на 1 тонну)	Суточные запасы (тонн)
M1	6	4	24
M2	1	2	6

Рынок накладывает следующие ограничения: краски для внутренних работ можно произвести в день не более, чем на 1 тонну больше, чем краски для наружных работ, и краски для внутренних работ требуется не более 2 тонны в день. Доход от производства 1 тонны краски для наружных (внутренних) работ равен 5 (4) млн. рублей. Требуется составить оптимальный план выпуска красок на день.
Математическая модель:

$$z = 5x_1 + 4x_2 \to \max,$$

$$6x_1 + 4x_2 \leq 24,$$

$$x_1 + 2x_2 \leq 6,$$

$$-x_1 + x_2 \leq 1,$$

$$x_2 \leq 2,$$

$$x_1, x_2 \geq 0.$$

Приведем построенную задачу к стандартной форме:

$$z = 5x_1 + 4x_2 + 0x_3 + 0x_4 + 0x_5 + 0x_6 \to \max,$$

$$6x_1 + 4x_2 + x_3 = 24,$$

$$x_1 + 2x_2 + x_4 = 6,$$

$$-x_1 + x_2 + x_5 = 1,$$

$$x_2 + x_6 = 2,$$

$$x_1, \ldots, x_6 \geq 0.$$

Выделим единичную подматрицу в матрице ограничений. Соответствующие переменные называются *базисными*. Они образуют *базис*. Остальные переменные называются небазисными или *свободными*.

Симплекс-метод осуществляет направленный перебор допустимых *базисных решений*, в которых базисные переменные неотрицательны, а небазисные равны нулю. Этот процесс соответствует переходу от одной угловой точки многогранника ограничений к другой в направлении неуменьшения значения целевой функции.

Представим задачу в виде *симплексной таблицы*:

Базис	z	x_1	x_2	x_3	x_4	x_5	x_6	Значение
z	1	-5	-4	0	0	0	0	0
x_3	0	6	4	1	0	0	0	24
x_4	0	1	2	0	1	0	0	6
x_5	0	-1	1	0	0	1	0	1
x_6	0	0	1	0	0	0	1	2

Таблица 1: Симплекс-таблица 1

Симплекс-метод.

Шаг 0. Находим начальное допустимое базисное решение.

Шаг 1. Определяем *вводимую переменную* x_{in} (*ведущий столбец*): это небазисная переменная с наименьшим отрицательным коэффициентом в строке z. Если все коэффициенты в строке z неотрицательны, то построенное базисное решение оптимально. Если существует небазисная переменная с отрицательным коэффициентом в строке z, все коэффициенты которой в матрице ограничений неположительны (столбец в симплекс-таблице), то значение целевой функции неограничено (может быть бесконечно большим).

Шаг 2. Определяем *выводимую переменную* x_{out} (*ведущую строку*): это базисная переменная с наименьшим неотрицательным значением отношения $b_{out}/a_{out,in}$. Элемент $a_{out,in}$ на пересечении ведущей строки и ведущего столбца называется *ведущим*.

Шаг 3. Пересчитываем симплекс-таблицу, применяя метод Гаусса (получаем ведущий элемент равным 1 и остальные элементы в ведущем столбце равными 0):

– модифицированная ведущая строка = ведущая строка, деленная на ведущий элемент,

– (любая другая строка, включая строку z): модифицированная строка = (строка) - (ее коэффициент в ведущем столбце)×(модифицированная ведущая строка).

Повторяем Шаг 1. ■

Отметим, что выбор ведущего столбца и ведущей строки может быть неоднозначным. Пока будем считать, что в таком случае выбор произволен.

В нашем примере вводимая переменная - x_1, выводимая переменная - x_3, и новая симплекс-таблица имеет вид (пересчитали таблицу и обозначение x_3 в первом столбце заменили на x_1):

Далее, вводимая переменная - x_2, выводимая переменная - x_4, и новая симплекс-таблица имеет вид: Поскольку все коэффициенты строки z неотрицательны, полученное решение $(x_1, \ldots, x_6) = (3, 3/2, 0, 0, 5/2, 1/2)$ со значением целевой функции $z = 21$ оптимально.

Базис	z	x_1	x_2	x_3	x_4	x_5	x_6	Значение
z	1	0	-2/3	5/6	0	0	0	20
x_1	0	1	2/3	1/6	0	0	0	4
x_4	0	0	4/3	-1/6	1	0	0	2
x_5	0	0	5/3	1/6	0	1	0	5
x_6	0	0	1	0	0	0	1	2

Таблица 2: Симплекс-таблица 2

Базис	z	x_1	x_2	x_3	x_4	x_5	x_6	Значение
z	1	0	0	3/4	1/2	0	0	21
x_1	0	1	0	1/4	-1/2	0	0	3
x_2	0	0	1	-1/8	3/4	0	0	3/2
x_5	0	0	0	3/8	-5/4	1	0	5/2
x_6	0	0	0	1/8	-3/4	0	1	1/2

Таблица 3: Симплекс-таблица 3

Таким образом, оптимальный план выпуска красок на день включает производство 3 тонн красок для наружных работ и 1.5 тонны краски для внутренних работ.
Иногда возникает проблема выбора начального допустимого базисного решения, например, когда единичная подматрица в матрице ограничений не просматривается. Тогда можно ввести искусственную переменную в каждое ограничение-равенство и положить ее коэффициент в целевой функции равным достаточно большому отрицательному числу $-M$. Такой метод получения начального допустимого базиса называется M-методом.
Возможность зацикливания. В некоторых случаях симплекс-метод через несколько итераций может вернуться к ранее определенному базисному решению и далее возвращаться к нему ∞ число раз. Эта ситуация, например, может возникнуть, когда при $n = 2$ в вершине многогранника ограничений пересекается более 2 прямых, т.е. когда некоторые ограничения являются несущественными. В этом случае некоторые базисные переменные равняются 0 и базис называется *вырожденным*. Для того, чтобы избежать зацикливания, при разработке математической модели желательно избегать излишних ограничений. Кроме того, зацикливания можно избежать, применяя следующее правило:
- в качестве ведущего выбирать столбец с отрицательным коэффициентом (необязательно наименьшим!) в строке z с **наименьшим номером**.
- в качестве ведущей выбирать строку с наименьшим неотрицательным значением отношения $b_{out}/a_{out,in}$ с **наименьшим номером**.
Указанное правило делает выбор ведущего элемента однозначным.

3.5 Транспортная задача ЛП

Транспортная задача (ТЗ) ЛП может быть сформулирована следующим образом. Имеется m пунктов отправления (производства) $A_1, \ldots, A_m$, в которых имеется товар в количестве $a_1, \ldots, a_m$ соответственно. Кроме того, имеется n пунктов назначения (потребления)

$B_1, \ldots, B_n$, в которых имеются заявки на этот товар в количестве $b_1, \ldots, b_n$ соответственно. Предполагается, что

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j,$$

т.е. все, что произведено, должно быть получено. Приведенное уравнение называется *условием баланса.*

Известна стоимость c_{ij} перевозки единицы товара из пункта отправления A_i в пункт потребления B_j. Требуется составить такой план перевозок товара, при котором весь товар из пунктов производства вывезен, все заявки в пунктах потребления удовлетворены и общая стоимость перевозок минимальна.

Математическая модель ТЗ состоит в следующем. Обозначим через x_{ij} количество товара, перевозимого из A_i в B_j. Составим задачу ЛП:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \to \min,$$

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

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

$$x_{ij} \geq 0, \ \forall i, j.$$

На практике условие баланса может не выполняться. Что делать? Если

$$\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j, \quad \text{(производится больше, чем потребляется)},$$

то вводим новый пункт потребления B_{n+1} с запросом

$$b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j.$$

Полагаем

$$c_{i,n+1} = 0, \ i = 1, \ldots, m.$$

Если же

$$\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j, \quad \text{(потребляется больше, чем производится)},$$

то вводим новый пункт производства A_{m+1} с предложением

$$a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i.$$

Полагаем

$$c_{m+1,j} = 0, \ j = 1, \ldots, n.$$

В дальнейшем будем считать, что условие баланса выполнено.

Отметим, что ранг системы ограничений ТЗ равен $m+n-1$ (на 1 меньше $m+n$ за счет связующего условия баланса). Поэтому количество базисных переменных также равно $m+n-1$.

Исходные данные ТЗ удобно представлять в виде транспортной таблицы (ТТ), см. ниже.

A_i/B_j	B_1	B_2	...	B_n	Запасы a_i
A_1	c_{11}	c_{12}	...	c_{1n}	a_1
A_2	c_{21}	c_{22}	...	c_{2n}	a_2
...	...	...	...	...	...
A_m	c_{m1}	c_{m2}	...	c_{mn}	a_m
Заявки b_j	b_1	b_2	...	b_n	$\sum a_i = \sum b_j$

Опорным планом ТЗ называется такое распределение объемов перевозок в ТЗ, что

– это распределение является допустимым,

– число базисных переменных равно $m+n-1$, все остальные (свободные) переменные равны нулю. Отметим, что некоторые базисные переменные также могут равняться нулю,

– не существует *цикла*, все вершины которого соответствуют базисным переменным.

Циклом в ТТ называется набор клеток, соединенных замкнутой ломаной линией, которая в точках излома совершает поворот на 90^0.

Метод потенциалов решения ТЗ.

Основные этапы:

Этап 1. Отыскание начального опорного плана.

Этап 2. Проверка текущего опорного плана на оптимальность.

Этап 3. Переход к лучшему опорному плану.

Этап 1. Известны 2 основных метода отыскания начального опорного плана: *метод северо-западного угла* и *метод минимальной стоимости*.

В методе северо-западного угла каждый раз определяется максимальный объем перевозок, соответствующий северо-западной допустимой клетке таблицы. При этом, если необходимо, полагаем $x_{ij}=0$. Поясним на примере.

A_i/B_j	B_1	B_2	B_3	B_4	B_5	Запасы a_i
A_1	18^{10}	27^{8}	3^{5}	^{6}	^{9}	48
A_2	^{6}	^{7}	30^{8}	^{6}	^{5}	30
A_3	^{8}	^{7}	9^{10}	12^{8}	6^{7}	27
A_4	^{7}	^{5}	^{4}	^{6}	20^{8}	20
Заявки b_j	18	27	42	12	26	$\sum = 125$

$$c = 1039$$

В методе минимальной стоимости каждый раз определяется максимальный объем перевозок, соответствующий допустимой клетке таблицы с наименьшей стоимостью c_{ij}. Поясним на примере.

A_i/B_j	B_1	B_2	B_3	B_4	B_5	Запасы a_i
A_1	14^{10}	^{8}	22^{5}	12^{6}	^{9}	48
A_2	4^{6}	^{7}	^{8}	^{6}	26^{5}	30
A_3	^{8}	27^{7}	^{10}	^{8}	^{7}	27
A_4	^{7}	0^{5}	20^{4}	^{6}	^{8}	20
Заявки b_j	18	27	42	12	26	$\sum = 125$

$$c = 745$$

В базис ввели $7 < m + n - 1 = 8$ положительных переменных. Нужно ввести еще одну переменную, равную нулю, так, чтобы не образовался цикл. Например, $x_{42} = 0$.

Этап 2. Проверка на оптимальность В методе потенциалов вводятся в рассмотрение *двойственные переменные – потенциалы* u_i, $i = 1, \ldots, m$, (дополнительный столбец в матрице ТЗ) и v_j, $j = 1, \ldots, n$, (дополнительная строка в ТТ).

Для клеток, соответствующих базисным переменным, должно выполняться

$$u_i + v_j = c_{ij}.$$

Вначале один из потенциалов можно выбрать произвольно, например, $u_1 = 0$, остальные подсчитать по формуле $u_i + v_j = c_{ij}$ для клеток, соответствующих базисным переменным.

A_i/B_j	B_1	B_2	B_3	B_4	B_5	Запасы a_i	Потенциалы u_i
A_1	$^{+\ 10}$	$-\ 14^{8}$	$+\ 22^{5}$	12^{6}	$^{+\ 9}$	48	0
A_2	4^{6}	$^{+\ 7}$	$^{+\ 8}$	$^{+\ 6}$	26^{5}	30	-3
A_3	14^{8}	13^{7}	$^{+\ 10}$	$^{+\ 8}$	$^{+\ 7}$	27	-1
A_4	$^{+\ 7}$	$^{-2}\ +^{5}$	$-\ 20^{4}$	$^{+\ 6}$	$^{+\ 8}$	20	-1
Заявки b_j	18	27	42	12	26	$\sum = 125$	
Потенциалы v_j	9	8	5	6	6		

Критерий оптимальности: если

$$\Delta_{ij} = c_{ij} - (u_i + v_j) \geq 0 \text{ для всех клеток ТТ,}$$

то построенный опорный план является оптимальным. В левый верхний угол каждой клетки ТТ, соответствующей небазисной переменной, запишем значение Δ_{ij}.

Этап 3. Переход к лучшему опорному плану Если существует $\Delta_{ij} < 0$, то переходим к лучшему опорному плану следующим образом.

Определяем переменную, которая будет введена в базис, и переменную, которая будет из него выведена. В базис введем переменную $x_{i^0 j^0}$ такую, что

$$\Delta_{i^0 j^0} = \min\{\Delta_{ij} | \forall i, j\}.$$

A_i/B_j	B_1	B_2	B_3	B_4	B_5	Запасы a_i	Потенциалы u_i
A_1	$^{+\ 10}$	^{8}	36^{5}	12^{6}	$^{+\ 9}$	48	0
A_2	4^{6}	$^{+\ 7}$	$^{+\ 8}$	$^{+\ 6}$	26^{5}	30	-1
A_3	14^{8}	13^{7}	$^{+\ 10}$	$^{+\ 8}$	$^{+\ 7}$	27	1
A_4	$^{+\ 7}$	14^{5}	6^{4}	$^{+\ 6}$	$^{+\ 8}$	20	-1
Заявки b_j	18	27	42	12	26	$\sum = 125$	
Потенциалы v_j	7	6	5	6	6		

$$c = 721 - (8 + 4 - 5 - 5) \cdot 14 = 693$$

Обозначим клетку (i^0, j^0) знаком "+". Рассмотрим цикл, образованный этой клеткой и клетками, соответствующими базисным переменным. Припишем клеткам цикла знаки "+" и "-" так, что они чередуются при каком-либо *обходе* этого цикла. Вычислим

$$x_{i^*j^*} = Q = \min\{x_{ij} | \text{ клетка } (i, j) \text{ отмечена "-"}\}.$$

Переменную $x_{i^*j^*}$ выводим из базиса.
Если минимум достигается на нескольких переменных, то выводим из базиса только одну переменную.
Пересчитываем новые объемы перевозок для элементов цикла:
$x_{ij} = x_{ij} + Q$, если клетка (i, j) помечена "+", и
$x_{ij} = x_{ij} - Q$, если клетка (i, j) помечена "-".
Для нового опорного плана определяем новые потенциалы и проверяем его на оптимальность. Процесс повторяем до тех пор, пока условие оптимальности не будет выполнено.
В нашем примере меняется лишь $v_2 = 6$. Все $\Delta_{ij} \geq 0$. Следовательно, построенное решение оптимально.

3.6 Задача о назначениях

Задача о назначениях может быть сформулирована следующим образом. Имеется n рабочих и m работ и задана стоимость назначения рабочего i на работу j для всех i и j. Каждый рабочий может быть назначен не более, чем на одну работу, и каждая работа может выполняться не более, чем одним рабочим. Требуется найти допустимое назначение, при котором суммарная стоимость F минимальна.
Если рабочий не может выполнять какую-то работу, соответствующая стоимость равна бесконечности. Не ограничивая общности, можно считать, что $n = m$. Иначе можно ввести фиктивных рабочих или фиктивные работы с нулевыми стоимостями.
Очевидно, что задача о назначениях является частным случаем транспортной задачи, в котором рабочие и работы могут рассматриваться как пункты производства и потребления, а запрос или заявка в каждом пункте равна единице. Для ее решения можно воспользоваться методом решения ТЗ. Однако, существует более эффективный метод ее решения, называемый *венгерским*.
Рассмотрим задачу о назначениях, заданную следующей матрицей стоимостей назначения.

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

Вначале вычтем из каждого элемента строки наименьший элемент этой строки по всем строкам. Получим матрицу

0	3	5	2
2	0	3	2
0	1	7	3
3	2	3	0

Найдем сумму вычтенных чисел $\Delta = 17$. Далее вычтем из каждого элемента столбца наименьший элемент этого столбца по всем столбцам. Получим матрицу

0	3	2	2
2	0	0	2
0	1	4	3
3	2	0	0

Найдем новую сумму $\Delta = 20$. Задача о назначениях с полученной матрицей стоимостей эквивалентна исходной задаче о назначениях. Для получения стоимости оптимального назначения исходной задачи необходимо к стоимости оптимального назначения новой задачи прибавить сумму вычтенных чисел: $F = F + \Delta$.
Последняя полученная матрица называется *приведенной*.
Общий шаг. Если в приведенной матрице можно выбрать n нулевых элементов так, что все они расположены в разных строках и столбцах, то такой выбор определяет оптимальное назначение. Иначе, находим минимальное число линий - строк и столбцов, покрывающих все нули. В нашем случае таких линий 3 - это столбец 1 и строки 2 и 4. Среди элементов, не покрытых этими линиями, находим наименьший элемент. Вычитаем его из непокрытых элементов и прибавляем к дважды покрытым (строкой и столбцом). Получаем матрицу

0	2	1	1
3	0	0	2
0	0	3	2
4	2	0	0

Повторяем Общий шаг. На сей раз существуют n нулевых элементов в разных строках и столбцах:

0	2	1	1
3	0	**0**	2
0	**0**	3	2
4	2	0	**0**

Они и определяют оптимальное назначение: рабочий 1 на работу 1, рабочий 2 на работу 3 и т.д. Стоимость назначения равна 21.
Неочевидным является вопрос о нахождении минимального числа линий, покрывающих все нули в матрице. При небольшой размерности матрицы это число можно найти при помощи полного перебора. Иначе, можно применить следующий алгоритм решения *упрощенной задачи о назначениях*. В этой задаче лишь нулевые элементы матрицы рассматриваются как допустимые назначения и необходимо выделить максимальное количество нулей таких, что никакие 2 нуля не расположены в одной строке или столбце.
Алгоритм решения упрощенной задачи о назначениях

Шаг 0. Строим любое допустимое назначение. Например, в каждой строке отмечаем ноль в столбце с наименьшим номером, не содержащем отмеченный ноль.

	1	2	3	4
1	**0**	3	2	2
2	2	**0**	0	2
3	0	1	4	3
4	3	2	**0**	0

Шаг 1. Помечаем знаком "-" все строки, не содержащие отмеченных нулей.

	3				
	1	2	3	4	
1	**0**	3	2	2	1
2	2	**0**	0	2	
3	0	1	4	3	-
4	3	2	**0**	0	

а) В каждой помеченной строке, например i, находим неотмеченные нули. Столбцы, соответствующие этим нулям, отмечаем номером строки (i). Однажды помеченный столбец далее не помечается. Просматриваем все помеченные строки.
б) В каждом помеченном столбце, например j, отыскиваем отмеченный нуль. Если он найден, помечаем строку, где он расположен, номером столбца (j). Просматриваем все помеченные столбцы. Возвращаемся к а), так как появились новые помеченные строки.
Шаг 1 завершается, когда помечен столбец, не содержащий отмеченного нуля. В этом случае количество отмеченных нулей может быть увеличено следующим образом. В помеченном столбце, не содержащем отмеченного нуля, отмечаем ноль в строке, указываемой пометкой этого столбца. Затем в этой строке делаем неотмеченным нуль в столбце, указываемым пометкой этой строки. Процесс повторяется до тех пор, пока не отметим нуль в строке, первоначально помеченной знаком "-" (не содержащей отмеченных нулей).
Повторяем Шаг 1. Алгоритм завершается, когда невозможно сделать ни одной пометки.
Помеченные столбцы и непомеченные строки образуют минимальное количество линий, покрывающий все нули. ■

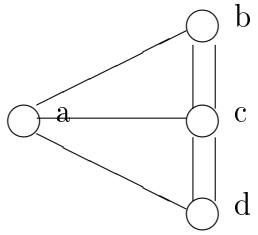
4 Основы теории графов

Граф – это математическое понятие, которое служит для моделирования связей между объектами. Различают ориентированные и неориентированные графы. Точные определения даны ниже.

4.1 Основные понятия

Неориентированным графом называется пара конечных множеств (V, E), где V – произвольное множество объектов, называемых *вершинами*, и $E \subseteq \{\{i, j\} | i, j \in V\}$ – множество неупорядоченных пар вершин, где $\{i, j\} \in E$ называется *ребром*.

Ориентированным графом (орграфом) называется пара конечных множеств (V, A), где V – множество вершин, и $A \subseteq \{(i,j)|i,j \in V\}$ – множество **упорядоченных** пар вершин, где $(i,j) \in A$ называется *дугой*.
Мультиграфом (ориентированным мультиграфом) называется граф с кратными ребрами (кратными дугами).
Примеры: рисунки графов с 3 вершинами.
Говорят следующее. Вершина i является *начальной*, а вершина j – *конечной* вершиной дуги (i,j). При этом i и j – *смежные вершины*, а дуга (i,j) *инцидентна* вершинам i и j. Такая же терминология используется для неориентированных графов – смежные вершины и инцидентные ребра. Дуга (i,j) *выходит из* i и *входит в* j. Если (i,j) – дуга, то i называется *непосредственным предшественником* j, а j – *непосредственным последователем* i. Количество дуг, входящих в вершину i называется *степенью захода* этой вершины, а количество дуг, выходящих из вершины i называется *степенью исхода* этой вершины. В неориентированном графе вершины i и j называются *концами* ребра $\{i,j\}$. Количество ребер, инцидентных вершине i, называется *степенью* вершины i.
Маршрутом в неориентированном графе $G = (V, E)$ называется такая последовательность вершин $W = (v_0, v_1, \ldots, v_k)$, $k \geq 0$, что $(v_{i-1}, v_i) \in E$, $i = 1, \ldots, k$. Граф называется *связным*, если существует маршрут, связывающий любые его две вершины. Говорят, что маршрут W связывает вершины v_0 и v_k, а вершины $v_1, \ldots, v_{k-1}$ являются *внутренними* вершинами этого маршрута. Количество вершин маршрута называется его *длиной*. Маршрут W называется *замкнутым*, если $k > 0$ и $v_k = v_0$. Маршрут называется *цепью*, если в нем нет повторяющихся вершин. Замкнутый маршрут, в котором никакие вершины, кроме первой и последней, не повторяются, называется *циклом*. Цикл, который включает все вершины графа, называется *гамильтоновым*. Граф, который содержит гамильтонов цикл, называется *гамильтоновым*.
Замкнутый маршрут, который включает каждое ребро графа ровно один раз, называется *эйлеровым маршрутом*, а граф, который содержит такой маршрут, называется *эйлеровым графом*. Задачу, является ли граф эйлеровым, впервые поставил и решил Эйлер. В 1736 г. он писал: "В городе Кенигсберге есть два острова, которые соединяются между собой и с берегами реки семью мостами, см. рис.



Можно ли спланировать прогулку так, чтобы, начиная с одного из 4 участков суши a, b, c, d, пройти по каждому из этих мостов один раз и вернуться в начальный пункт?

Теорема 3 *(Эйлер, 1736 г) Связный мультиграф является эйлеровым тогда и только тогда, когда все его вершины имеют четную степень.*

Аналогами приведенных выше определений для орграфов являются следующие.

Граф	Маршрут	Цепь	Цикл
Орграф	Путь	Простой путь	Цикл

4.2 Алгоритм построения эйлерова цикла для неориентированного графа

Предполагается, что неориентированный граф $G = (V, E)$ связен и степени его вершин четны.

Шаг 1. Выбираем произвольную вершину v_1 и полагаем частичный эйлеров цикл $C^* = \{v_1\}$. В графе G будем двигаться от некоторой вершины частичного цикла и помечать пройденные ребра. Помеченное ребро будем включать в эйлеров цикл. Полагаем $i = 1$.

Шаг 2. Двигаемся от вершины v_i по непомеченным ребрам и помечаем их до тех пор, пока не вернемся в v_i. Пусть при этом построен цикл C.

Цикл C включаем в C^* так, что вначале идут ребра C^* до вершины v_i, затем ребра цикла C и затем оставшиеся ребра C^*.

Если все ребра исходного графа помечены, то эйлеров цикл построен. Если есть непомеченные ребра, то в силу связности графа должна существовать вершина $v_j \in C^*$, которая является концом непомеченного ребра. Пусть v_j – последняя из таких вершин, входящая в C^*.

Шаг 3. Удаляем из графа ребра цикла C. В оставшемся графе вершины будут по-прежнему иметь четную степень. Полагаем $i = j$ и переходим к Шагу 2, т.е. строим новый цикл на непомеченных ребрах, начиная с вершины v_j. ■

Примеры. Два треугольника, имеющие общую вершину. Два четырехугольника, имеющие общую сторону, вершины которой дополнительно соединены цепью.

4.3 Задание графов

Матрицей смежности графа $G = (V, E)$ называется матрица $A = ||a_{ij}||_{n\times n}$ с элементами $a_{ij} = 1$, если $\{i, j\} \in E$, и $a_{ij} = 0$, если $\{i, j\} \notin E$.

Для мультиграфов a_{ij} равно кратности ребра $\{i, j\}$. Аналогично вводится матрица смежности для орграфов.

Матрицей инцидентности ориентированного графа $G = (V, A)$ называется матрица $A = ||a_{ie}||_{n\times m}$ с элементами $a_{ie} = 1$, если e является выходящей из вершины i дугой, $a_{ie} = -1$, если e является входящей в вершину i дугой и $a_{ie} = 0$, если e не инцидентна i.

Для неориентированного графа, $a_{ie} = 1$, если вершина i инцидентна ребру e, и $a_{ie} = 0$ в противной случае.

Графы могут быть также представлены одним или несколькими списками непосредственных последователей $A^0(i)$ и непосредственных предшественников $B^0(i)$ каждой вершины i.

4.4 Топологическая сортировка

Предположим, что некоторой фирме необходимо выполнить n проектов, на множестве $V = \{1, \ldots, n\}$ которых задано *отношение порядка* $\rightarrow$ такое, что из $i \rightarrow j$ следует, что проект j не может начаться раньше, чем закончится проект i (проект j использует информацию или продукцию, получаемую в результате выполнения проекта i). Нужно найти допустимый порядок выполнения проектов во времени.

Рассмотрим орграф $G = (V, A)$ *отношения порядка.* Дуга $(i, j) \in A$ тогда и только тогда, когда $i \rightarrow j$.

Очевидно, что решение задачи существует, если орграф отношения порядка не содержит циклов. Если есть цикл, то ни один из проектов, принадлежащих циклу, нельзя начинать.

Пусть перестановка $(i_1, \ldots, i_n)$ определяет порядок выполнения проектов. Этот порядок является допустимым, если из $i_j \rightarrow i_k$ следует, что i_j предшествует i_k в указанной перестановке.

Введем понятие номера вершины – $label(i) \in \{1, \ldots, n\}$. Нетрудно заметить, что задача сводится к отысканию нумерации вершин графа G такой, что из $(i, j) \in A$ следует $label(i) < label(j)$. Тогда последовательность проектов в порядке возрастания их номеров $label(i)$ будет являться решением задачи. Указанная нумерация называется *топологической сортировкой вершин графа.*

Алгоритм топологической сортировки.

Предположим, что граф задан списками непосредственных последователей $A^0(i)$, $i = 1, \ldots, n$, и, кроме того, задано количество непосредственных предшественников b_i каждой вершины $i : b_i = |B^0(i)|$.

Шаг 1. Просматриваем список $(b_1, \ldots, b_n)$ и создаем список $J = \{j | b_j = 0\}$ вершин, у которых нет предшественников. Если $J = \phi$, то в графе есть цикл.

Шаг 2. Присваиваем текущий номер j (вначале $j = 1$) любой вершине из J. Далее рассматриваем всех ее непосредственных последователей $i \in A^0(j)$ и преобразуем $b_i = b_i - 1$. Если $b_i = 0$, то вершину i включаем в J. После того, как все множество $A^0(j)$ просмотрено, удаляем j из J. Если текущий номер не равен n, увеличиваем его на единицу. Повторяем Шаг 2. ■

4.5 Остовное дерево (остов) минимального веса

Граф, не имеющий циклов, называется *лесом.* Связный граф, не имеющий циклов, называется *деревом.*

Пусть n и m – количества вершин и ребер графа соответственно.

Теорема 4 (Свойства деревьев) *Пусть G – неориентированный граф. Тогда следующие условия эквивалентны:*

1) G – дерево,

2) G – связный граф и $m = n - 1$,

3) G – граф без циклов и $m = n - 1$,

4) между любой парой вершин существует единственная цепь,

5) G не имеет циклов, но при добавлении произвольного ребра в G в нем возникает единственный цикл.

В ориентированном графе вводятся понятия *входящего и выходящего дерева.* Это орграф, неориентированный аналог которого является деревом, и,
в случае выходящего дерева, в каждую вершину входит не более одной дуги, а
в случае входящего дерева, из каждой вершины выходит не более одной дуги.
Единственная вершина выходящего дерева, в которую не входит ни одна дуга, называется *корнем.*
Подграф неориентированного графа $G = (V, E)$ называется *остовным деревом (остовом)* и обозначается T, если T – дерево и множество его вершин совпадает с V.
Граф называется *(реберно) взвешенным*, если каждому его ребру $\{i, j\}$ приписан некоторый вес w_{ij}.
Весом остовного дерева называется сумма весов всех его ребер.
Задача поиска в графе остовного дерева минимального веса имеет много приложений. Она возникает при проектировании дорог, электрических сетей, трубопроводов и т.п., т.е. в ситуациях, когда необходимо связать заданное множество объектов коммуникационными линиями и суммарная стоимость этих линий должна быть минимальна.
Ниже приводятся два алгоритма, которые строят остовное дерево минимального веса T_{n-1}. Напомним, что T_{n-1} не содержит циклов и имеет $n - 1$ ребер.
Алгоритм Краскала
Шаг 1. Полагаем $T_0 = (V, \phi)$, $|V| = n$.
Шаг 2. Для $i = 0, \ldots, n-1$ полагаем $T_{i+1} = T_i \cup e$, где e – ребро G с минимальным весом такое, что оно не является ребром T_i и не образует цикла с ребрами T_i. ■
Алгоритм Прима
Шаг 1. Полагаем $T_1 = (V_1, E_1)$, где $V_1 = \{a, b\}$, $E_1 = \{a, b\}$, и $w_{ab} = \min\{w_e | e \in E\}$.
Шаг 2. Для $i = 1, \ldots, n-1$ полагаем $T_{i+1} = T_i \cup e$, где e – ребро G с минимальным весом такое, что оно не является ребром T_i и связывает T_i с вершиной G, не принадлежащей T_i.
■

Пример. Любой неориентированный реберно взвешенный граф.

4.6 Кратчайшие пути

Рассмотрим орграф со взвешенными дугами и путь из вершины i в вершину j в этом графе. Сумма весов дуг этого пути называется его *длиной.* Путь минимальной длины из i в j называется *кратчайшим* путем из i в j.
Дейкстра предложил следующий алгоритм построения кратчайших путей из данной вершины во все остальные вершины орграфа в случае неотрицательных весов дуг.
Рассмотрим орграф $G = (V, A)$ с выделенной вершиной s и весами w_{ij}, $(i, j) \in A$. Алгоритм присваивает вершинам временные и постоянные метки. Постоянная метка вершины равна длине кратчайшего пути из s в эту вершину. Кроме того, в алгоритме формируется массив $Pred$, в котором $Pred(v)$ содержит номер непосредственного предшественника вершины

v в кратчайшем пути из s в v. Таким образом $s, \ldots, Pred(Pred(v)), Pred(v), v$ является кратчайшим путем из s в v.

Алгоритм Дейкстры (построения кратчайших путей из вершины s во все остальные вершины орграфа в случае неотрицательных весов дуг)

Шаг 1. (Начало). Помечаем вершину s постоянной меткой $Length(s) = 0$. Все остальные вершины $v \neq s$ помечаются временными метками $Length(v) = \infty$. Полагаем $i = 1$ и $u = s$.

Шаг 2. (Рекурсия). Полагаем $i = i + 1$. Для всех непосредственных последователей v с временной меткой вершины u вычисляем $M = \min\{Length(v), Length(u) + w_{uv}\}$. Если M меньше временной метки $Length(v)$, то вычисляем новую метку $Length(v) = M$ и полагаем $Pred(v) = u$.

Далее среди вершин с временными метками выбираем вершину k с наименьшей меткой и делаем ее постоянной. Если $i < n - 1$, то полагаем $u = k$ и повторяем Шаг 2. Иначе **стоп**: все кратчайшие пути из s в остальные вершины могут быть восстановлены по массиву Pred. ■

Алгоритм построения кратчайших путей между всеми парами вершин и проверки наличия цикла отрицательного веса (дополнительный материал). Приводимый ниже алгоритм предложен Флойдом.

Рассмотрим орграф $G = (V, A)$. Пусть $W^0 = ||w_{ij}|| - n \times n$ матрица весов дуг этого графа. Полагаем $w_{ij} = \infty$, если дуга (i, j) отсутствует, и $w_{ii} = 0$ для всех $i \in V$. Алгоритм Флойда строит последовательность матриц $W^1, W^2, \ldots, W^n$ такую, что элемент w_{ij}^n матрицы W^n равен длине кратчайшего пути из i в j в графе G. Матрица W^k определяется по матрице W^{k-1}:

$$w_{ij}^k = \min\{w_{ij}^{k-1}, w_{ik}^{k-1} + w_{kj}^{k-1}\}. \tag{1}$$

Пусть P_{ij}^k – кратчайший путь из i в j с внутренними вершинами из множества $\{1, 2, \ldots, k\}$. Справедлива

Теорема 5 *Для $0 \leq k \leq n$, величина w_{ij}^k является длиной пути P_{ij}^k.*

В алгоритме Флойда одновременно с длинами кратчайших путей отыскиваются сами пути с помощью матриц $Z^0, Z^1, \ldots, Z^n$, где элемент z_{ij}^k матрицы Z^k указывает на вершину, непосредственно следующую за i в P_{ij}^k. Тогда ясно, что

$$z_{ij}^0 = \begin{cases} j, & \text{если } w_{ij}^0 \neq \infty, \\ 0, & \text{если } w_{ij}^0 = \infty. \end{cases}$$

Матрица Z^k получается из Z^{k-1} следующим образом. Пусть $M = \min\{w_{ij}^{k-1}, w_{ik}^{k-1} + w_{kj}^{k-1}\}$. Тогда

$$z_{ij}^k = \begin{cases} z_{ij}^{k-1}, & \text{если } M = w_{ij}^{k-1}, \\ z_{ik}^{k-1}, & \text{если } M < w_{ij}^{k-1}. \end{cases}$$

Если $M = w_{ij}^{k-1}$, то длина пути P_{ij}^k равна длине пути P_{ij}^{k-1}. Иначе P_{ij}^k получается в результате склеивания путей P_{ik}^{k-1} и P_{kj}^{k-1} и $z_{ij}^k = z_{ik}^{k-1}$.

Очевидно, что кратчайший путь из i в j определяется последовательностью вершин $i, i_1, i_2, \ldots, i_p, j$, где

$$i_1 = z_{ij}^n,\ i_2 = z_{i_1 j}^n,\ i_3 = z_{i_2 j}^n,\ \ldots,\ j = z_{i_p j}^n.$$

Отметим, что в (1) если w_{ik}^{k-1} или w_{kj}^{k-1} равно ∞, то $w_{ij}^k = w_{ij}^{k-1}$.

Алгоритм Флойда (построения кратчайших путей между всеми парами вершин и проверки наличия цикла отрицательного веса)

(В процессе вычислений матрицу W^k записываем на место W^{k-1}, а матрицу Z^k – на место Z^{k-1}, $k = 1, \ldots, n$. Таким образом, используем одно и то же обозначение W для всех W^k и Z для всех Z^k.)

Шаг 1. (Начало). Полагаем $k = 0$. Строим матрицы $W = W^0$ и $Z = Z^0$.

Шаг 2. (Рекурсия). Полагаем $k = k + 1$. Для всех таких вершин $i \neq k$, что $w_{ik} \neq \infty$ и всех таких вершин $j \neq k$, что $w_{kj} \neq \infty$, вычисляем $M = \min\{w_{ij}, w_{ik} + w_{kj}\}$. Если $M < w_{ij}$, то полагаем $z_{ij} = z_{ik}$ и $w_{ij} = M$.

Если появляется $w_{ii} < 0$, то **стоп**: вершина i принадлежит некоторому циклу отрицательного веса.
Если все $w_{ii} \geq 0$ и $k = n$, то **стоп**: w_{ij} – длины всех кратчайших путей, а z_{ij} – первая после i вершина в кратчайшем пути из i в j.

Если все $w_{ii} \geq 0$ и $k < n$, то повторяем Шаг 2. ■

5 Сетевое планирование и управление проектами

На практике довольно часто встречаются задачи планирования и управления сложными работами или проектами.
Примерами таких проектов могут быть
– строительство большого объекта,
– разработка комплекса компьютерных программ,
– производство самолета, ракеты или другого большого изделия,
– выпуск нового продукта,
– передислокация фирмы или производства.

Каждый сложный проект включает ряд более или менее элементарных работ, которые взаимосвязаны по времени и/или по использованию общих ресурсов. При оптимальном планировании и управлении проектом необходимо составить *расписание* выполнения всех элементарных работ. Под расписанием понимается указание о том, как распределены имеющиеся ресурсы между работами во времени. При этом расписание допустимо, если выполнены ограничения на ресурсы и временные соотношения между работами.
Наиболее распространенными методами планирования и управления проектами являются *сетевые методы*. Для того, чтобы использовать эти методы, необходимо предварительно выполнить следующие действия:
1. Составить математическую модель проекта. Для этого
а) Определить элементарные работы (иногда называемые требованиями или операциями), которые должны быть проведены для выполнения проекта.
б) Представить связи между элементарными работами в виде *сети*, в которой вершины и дуги представляют собой связи между элементарными работами. При этом будет присутствовать начальная, возможно фиктивная, работа, соответствующая началу проекта, и конечная работа, соответствующая его завершению.
2. Построить расписание выполнения проекта. Для этого
а) Определить или оценить длительность выполнения p_j каждой элементарной работы j.
б) Определить *критический*, т.е. наиболее длинный путь между начальной и конечной вершиной сети. Длина пути равна сумме длительностей выполнения входящих в него элементарных работ и представляет собой наименьшую возможную длительность выполнения проекта.
в) Использовать знание критического пути для получения более экономичного и эффективного расписания.
3. Осуществлять контроль выполнения проекта. Для этого
а) Использовать полученную сеть и расписание для контроля выполнения проекта.
б) Вносить изменения в расписание в процессе выполнения проекта так, чтобы оно соответствовало реальным данным и отражало реальную ситуацию. ■

Некоторые работы не могут выполняться до тех пор, пора не завершится выполнение некоторых других работ. Это может быть обусловлено технологией процесса, физическими ограничениями или использованием информации, порождаемой другими работами. Как отмечалось в разделе "Топологическая сортировка", такие ограничения называются ограничениями предшествования. Работы, не связанные ограничениями предшествования, могут выполняться независимо друг от друга, параллельно во времени.

Отношения предшествования отражаются в сетевой модели проекта.

Наиболее распространены два типа сетевых моделей, называемые 1) *действие-на-вершине* и 2) *действие-на-дуге*.

В модели 1) вершины представляют собой элементарные работы проекта, а дуги указывают последовательность их выполнения. Вершинам приписаны длительности выполнения соответствующих работ.

В модели 2) дуги представляют собой элементарные работы проекта, а вершины соответствуют событиям "начало работы" и "конец работы". Дугам сопоставлены длительности выполнения соответствующих работ. Для отражения отношений предшествования между работами могут вводиться фиктивные дуги нулевой длины.

Рассмотрим модель 2) "действие-на-дуге". Как уже отмечалось, длительность выполнения проекта определяется длиной критического пути. Проект не может быть выполнен быстрее, чем длина критического (самого длинного) пути.

Алгоритм отыскания критического пути

Применяется для отыскания критического пути между выделенными вершинами s и t ацикличного орграфа $G = (V, A)$ с неотрицательными весами дуг p_{ij}, $(i, j) \in A$.

Метка вершины $Length(v)$ соответствует длине самого длинного пути из s в v.

Шаг 1. (Начало). Помечаем вершину s меткой $Length(s) = 0$. Полагаем множество помеченных вершин $S = \{s\}$.

Шаг 2. (Рекурсия). Находим непомеченную вершину v, в которую входят дуги только из помеченных вершин множества $B_v^0 \subseteq S$. Помечаем v меткой $Length(v) = \max\{Length(u) + p_{uv} | u \in B_v^0\}$, т.е. включаем v в S. Если максимум достигается на вершине $u^0 \in B_v^0$, то самый длинный путь в вершину v пришел из вершины u^0.

Если помечена вершина t, то **стоп:** длина критического пути равна $Length(t)$ и сам путь может быть восстановлен обратным просмотром уравнений рекурсии. Иначе повторяем Шаг 2. ■

После того, как критический путь построен, для каждой работы j могут быть вычислены *наиболее ранний* ES_j и *наиболее поздний* LS_j моменты ее начала такие, что начало работы в промежутке $[ES_j, LS_j]$ не приводит к увеличению длительности всего проекта. Начало работы ранее ES_j невозможно из-за ограничений предшествования, а ее начало после LS_j приведет к увеличению длительности проекта.

Знание указанных моментов позволяет реализовать расписание более гибко, поскольку есть возможность задержки начала выполнения некоторых работ без ущерба для длительности выполнения всего проекта.

Наиболее ранние моменты начала работ ES_j равны пометкам соответствующих вершин в приведенном выше алгоритме.
Наиболее поздние моменты начала работ подсчитываются при помощи процедуры прохождения сети от конечной вершины к начальной. При этом вершине t приписывается значение T длины критического пути, и далее

$$LS_i = \min\{LS_j - p_{(ij)} | j \in A_i^0, \text{ все вершины } A_i^0 \text{ помечены}\}.$$

Значение LS_i равно T минус длина наиболее длинного пути из вершины i в вершину t.

6 Задача коммивояжера и метод ветвей и границ

Предположим, что фирма должна провести рекламную кампанию во всех областных городах Беларуси. Она решила провести рекламную кампанию в каждом городе ровно по одному разу. Известно, сколько стоит переезд между двумя любыми городами. Фирма желает снизить свои дорожные расходы.
Математическая модель такой ситуации называется *задачей коммивояжера*, которая состоит в следующем. Задан *полный* ориентированный или неориентированный граф, для определенности, орграф. Орграф называется полным, если каждая пара вершин i и j соединена дугами (i, j) и (j, i). Известна стоимость c_{ij} каждой дуги (i, j). Требуется построить гамильтонов цикл (т.е. цикл, проходящий через каждую вершину ровно по одному разу) такой, что суммарная стоимость всех его дуг была минимальной.
Отметим, что есть взаимно однозначное соответствие между гамильтоновыми циклами (т.е. маршрутами коммивояжера) в полном графе с n вершинами и перестановками n элементов. Перестановка $\sigma = (i_1, \ldots, i_n)$ соответствует маршруту коммивояжера, проходящему через вершины $i_1, \ldots, i_n$ и включающему дуги $(i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n), (i_n, i_1)$.
Основным методом решения задачи коммивояжера является *метод ветвей и границ (МВиГ)*. Этот метод является универсальным и может применяться для решения практически всех дискретных экстремальных задач.
Основные принципы МВиГ для решения задачи минимизации состоят в следующем.
1) Ветвление. Исходная задача разбивается (ветвится) на две или более подзадачи таким образом, что решение исходной задачи является решением хотя бы одной из подзадач. Каждая из подзадач, в свою очередь, аналогичным образом ветвится на более мелкие подзадачи. Процесс может повторяться до тех пор, пока получаемая подзадача не становится тривиальной и ее решение может быть легко найдено.
Указанный процесс можно представить в виде так называемого *дерева ветвлений*. Вершины дерева ветвлений соответствуют подзадачам и разбиты на *уровни*. Исходная задача является вершиной нулевого уровня. Подзадачи, полученные из исходной задачи, являются вершинами первого уровня. Подзадачи, полученные из вершин первого уровня, являются вершинами второго уровня, и т.д. Дуги направлены из вершин уровня i в вершины уровня $i + 1$. В каждую вершину входит ровно одна дуга.

2) Построение нижних и верхних оценок минимального значения целевой функции. Для исходной задачи, а также для любой ее подзадачи могут быть найдены числа LB и UB такие, что

$$LB \leq F^* \leq UB,$$

где F^* – минимальное значение целевой функции в задаче или подзадаче, LB и UB – его нижняя и верхняя оценки соответственно.

Нижняя оценка может быть получена в результате решения *релаксированной* или ослабленной задачи, например, задачи коммивояжера, в которой стоимости всех дуг, входящих в данную вершину, заменены на минимальную из них, и аналогично изменены (уменьшены) стоимости всех дуг, выходящих из данной вершины.

Верхняя оценка может быть получена в результате отыскания любого допустимого решения рассматриваемой задачи и вычисления соответствующего значения целевой функции. Для задачи коммивояжера достаточно подсчитать суммарную стоимость дуг для какого-нибудь допустимого маршрута (допустимой перестановки). Наилучшая, т.е. наименьшая из всех имеющихся в наличии верхних оценок, называется (текущим) *рекордом*.

Может оказаться, что задача отыскания нижней и/или верхней оценки сама является сложной. В этом случае соответствующая оценка не используется и можно положить $LB = -\infty$, $UB = \infty$.

3) Отсеивание вариантов. В случае, если для некоторой подзадачи ее нижняя оценка превосходит либо равна рекорду, такую подзадачу можно далее не ветвить, так как ее решение будет заведомо хуже либо не лучше решения, соответствующего рекорду.

4) Оптимальное решение. Процесс вычислений прекращается, когда нет ни одной подзадачи, которая может продолжать ветвиться. В этом случае оптимальное решение соответствует текущему рекорду. ■

В приведенном методе остается неопределенным способ выбора подзадачи для ветвления. Такие способы могут быть различными. Наиболее употребимым является способ, при котором ветвят ту подзадачу, которой соответствует наименьшее значение нижней оценки либо ту, которая быстрее приведет к тривиальной подзадаче.

Различают методы ветвления *в глубину* и *в ширину*.

При ветвлении в глубину всегда ветвят одну из подзадач наибольшего уровня. Такое ветвление быстрее приведет к построению полного допустимого решения.

При ветвлении в ширину ветвят подзадачи одного уровня до тех пор, пока все такие подзадачи не будут рассмотрены. Такое ветвление позволяет получать больше нижних и верхних оценок и может привести к большему количеству отсеянных вариантов. Однако требуется больше памяти для хранения промежуточных результатов по сравнению с ветвлением в глубину.

Рассмотрим задачу проведения рекламной кампании в областных центрах Беларуси. Пусть матрица (таблица) стоимостей переезда между городами выглядит следующим образом:

город(i,j)	Минск(1)	Брест(2)	Витебск(3)	Гомель(4)	Гродно(5)	Могилев(6)
1	∞	350	250	350	320	200
2	400	∞	650	500	100	700
3	300	600	∞	320	400	150
4	370	580	280	∞	710	180
5	400	100	550	800	∞	670
6	220	650	170	210	590	∞

Будем решать задачу методом ветвления вглубь. Вначале построим "приведенную задачу", эквивалентную исходной. Используем процедуру "приведение таблицы", состоящую в вычитании наименьшего числа в каждом столбце (строке), затем наименьшего числа в каждой строке (столбце) и суммировании этих чисел (из каждого города нужно выехать и в каждый город нужно въехать). Получаем $\Delta_0 = (220 + 100 + 170 + 210 + 100 + 150) + (30+50) = 1030$. Далее можно работать с приведенной матрицей. Для получения реальной стоимости любого маршрута необходимо добавить к получаемой из приведенной таблицы стоимости величину Δ_0.

Приведенная таблица:

(i,j)	1	2	3	4	5	6
1	∞	200(250)	30(80)	90(140)	170(220)	0(50)
2	180	∞	480	290	0	550
3	80	500	∞	110	300	0
4	120(150)	450(480)	80(110)	∞	580(610)	0(30)
5	180	0	380	590	∞	520
6	0	550	0	0	490	∞

Для элементов приведенной таблицы оставим обозначения c_{ij}.

Очевидно, что нижняя оценка для приведенной задачи равна $LB(0) = 0$. Для отыскания верхней оценки построим какой-нибудь маршрут коммивояжера. Например, разумным будет включить в этот маршрут дугу минимальной стоимости, скажем, (2,5). Далее выбрать дугу минимальной стоимости вида $(5,j)$, $j \neq 2$. Это (5,1). Далее, выбрать дугу минимальной стоимости вида $(1,j)$, $j \neq 2,5$. Поступая аналогично, (1,6), (6,3), (3,4), и, без вариантов, (4,2). Получаем верхнюю оценку $UB(0) = 740$. Она является рекордом R.

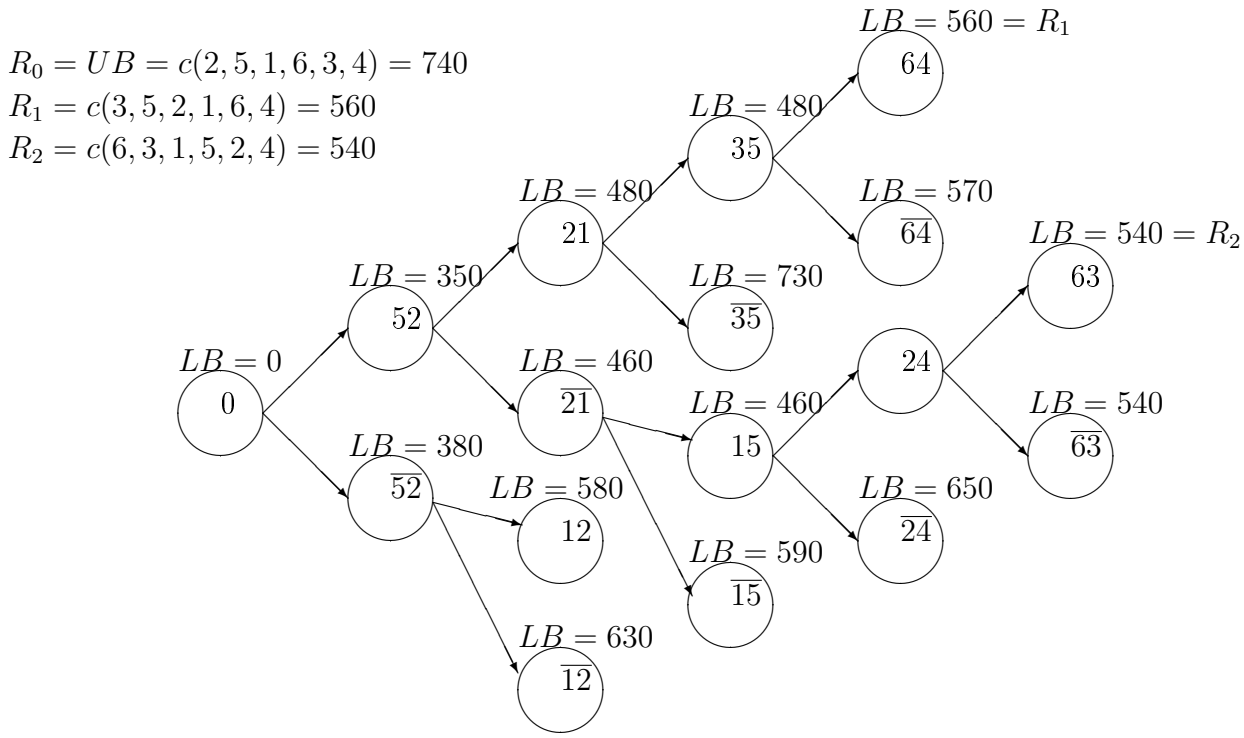
Отметим, что маршруту коммивояжера соответствует такое множество элементов таблицы, что в каждой строке и каждом столбце находится ровно один элемент этого множества. Однако, не каждое множество, содержащее ровно один элемент в каждой строке и каждом столбце является маршрутом коммивояжера. Такому множеству может соответствовать более одного цикла в графе.

Далее разветвим исходную задачу. Ветвление будем осуществлять по вхождению или невхождению некоторой дуги в маршрут. Обычно выбирают дугу 1) максимальной, но не равной ∞ стоимости либо 2) нулевой стоимости, которая дает наибольшее значение нижней оценки для подзадачи при ее невхождении в маршрут коммивояжера. Вначале воспользуемся способом 2). Дуги нулевой стоимости: (1,6), (2,5), (3,6), (4,6), (5,2), (6,1), (6,3), (6,4).

В подзадаче $P(\overline{1,6})$ дуга (1,6) не входит в маршрут. Полагаем стоимость этой дуги равной ∞. Какая-то дуга из 1 должна выйти и какая-то дуга должна войти в 6. Можно добавить

к нижней оценке сумму наименьших чисел в указанных строке и столбце $\Delta_{\overline{16}} = 30$ и вычислить $LB(\overline{1,6}) = LB(0) + \Delta_{\overline{1,6}} = 30$. Аналогично вычисляем $\Delta_{\overline{2,5}}$, $\Delta_{\overline{3,6}}$, $\Delta_{\overline{4,6}}$, $\Delta_{\overline{5,2}}$, $\Delta_{\overline{6,1}}$, $\Delta_{\overline{6,3}}$, $\Delta_{\overline{6,4}}$. Наибольшее значение равно $\Delta_{\overline{5,2}} = 380$. Следовательно, $LB(\overline{5,2}) = 380$. Выбираем дугу (5,2) для дальнейших ветвлений.

Нарисуем начальное дерево ветвлений, состоящее из 3 вершин - начальной 0 и вершин, обозначенных $(\overline{5,2})$ (соответствующая дуга не входит в маршрут коммивояжера) и (5,2) (входит), см. рис.



Ответ: $c^* = c(6,3,1,5,2,4) = 540 + \Delta_0 = 540 + 1030 = 1570$

Припишем вершинам соответствующие значения LB.

Рассмотрим подзадачу $P(5,2)$. Поскольку дуга (5,2) входит в маршрут, другие дуги не могут выйти из 5 и войти в 2. Вычеркиваем строку 5 и столбец 2. Кроме того, дуга (2,5) войти в маршрут не может. Полагаем стоимость этой дуги равной ∞. Вычисляем $\Delta_{5,2} = 350$. После приведения таблица имеет вид:

(i,j)	1	3	4	5	6
1	∞	30	90	0	0
2	0	300	110	∞	370
3	80	∞	110	130	0
4	120	80	∞	410	0
6	0	0	0	320	∞

Вычисляем $LB(5,2) = LB(0) + c_{5,2} + \Delta_{5,2} = 350$.

Будем ветвить подзадачу $P(5,2)$, так как размерность соответствующей матрицы меньше и быстрее можно получить тривиальную подзадачу. В приведенной таблице для этой

подзадачи находим дуги нулевой стоимости: (1,5), (1,6), (2,1), (3,6), (4,6), (6,1), (6,3), (6,4). Наибольшее значение $\Delta_{\overline{i,j}}$ при записи в указанные клетки ∞ равно $\Delta_{\overline{2,1}} = 110$. Выбираем дугу (2,1) для дальнейших ветвлений. В дерево ветвлений добавляем вершины (2,1) и $(\overline{2,1})$, следующие за вершиной (5,2).

В подзадаче $P(\overline{2,1})$ дуга (2,1) не входит в маршрут. Полагаем стоимость этой дуги равной ∞. Вычисляем $\Delta_{\overline{2,1}} = 110$ и $LB(\overline{2,1}) = LB(5,2) + \Delta_{\overline{2,1}} = 460$.

Рассмотрим подзадачу $P(2,1)$. Вычеркиваем строку 2 и столбец 1 из матрицы для подзадачи $P(5,2)$. Дуга (2,1) образует путь с уже введенной в маршрут дугой: (5,2),(2,1). Поэтому можно положить $c_{1,5} = \infty$. Вычисляем $\Delta_{2,1} = 130$. Полученная приведенная таблица имеет вид:

(i,j)	3	4	5	6
1	30	90	∞	0
3	∞	110	0	0
4	80	∞	280	0
6	0	0	190	∞

Вычисляем $LB(2,1) = LB(5,2) + \Delta_{2,1} = 480$.

Будем ветвить подзадачу $P(2,1)$. В приведенной таблице для этой подзадачи находим дуги нулевой стоимости: (1,6), (3,5), (3,6), (4,6), (6,3), (6,4). Наибольшее значение $\Delta_{\overline{i,j}}$ при записи в указанные клетки ∞ равно $\Delta_{\overline{3,5}} = 190$. Выбираем дугу (3,5) для дальнейших ветвлений. В дерево ветвлений добавляем вершины (3,5) и $(\overline{3,5})$, следующие за вершиной (2,1). Процесс останавливается для матриц 2×2. В этом случае можно найти решение либо определить, что оно не существует. Так для подзадачи $P(6,4)$ таблица имеет вид

(i,j)	3	6
1	∞	0
4	80	∞

По дереву ветвлений восстанавливаем построенный частичный маршрут: $(3,5,2,1)$ и $(6,4)$. Из таблицы видно, что единственный вариант - полный маршрут $(3,5,2,1,6,4)$ стоимости $LB(6,4) = LB(3,5) + c_{43} = 480 + 80 = 560$. Получили новый рекорд $R = 560$. Ветки $P(\overline{6,4})$ и $P(\overline{3,5})$ можно отсечь.

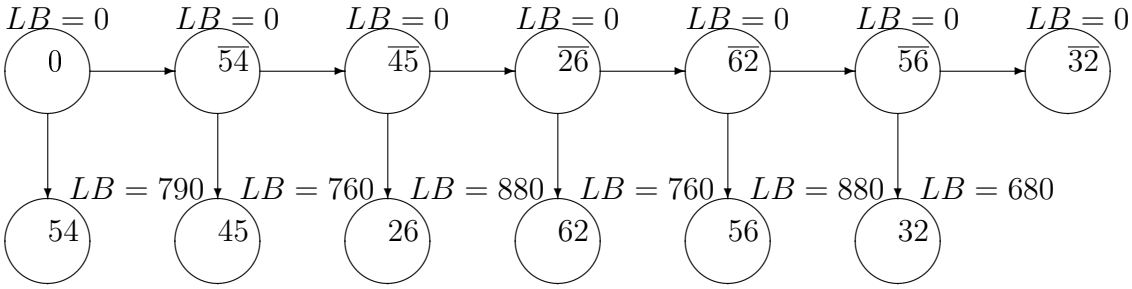
Приведенная таблица для подзадачи $P(\overline{2,1})$:

(i,j)	1	3	4	5	6
1	∞	30	90	0	0
2	∞	190	0	∞	260
3	80	∞	110	130	0
4	120	80	∞	410	0
6	0	0	0	320	∞

Приведенная таблица для подзадачи $P(\overline{5,2})$:

(i,j)	1	2	3	4	5	6
1	∞	0	30	90	170	0
2	180	∞	480	290	0	550
3	80	300	∞	110	300	0
4	120	250	80	∞	580	0
5	0	∞	200	410	∞	340
6	0	350	0	0	490	∞

На этом остановимся и покажем, как ветвить при выборе дуги максимальной стоимости. Будем выбирать дугу максимальной стоимости. Это дуга (5,4), $c_{5,4} = 590$.
Нарисуем начальное дерево ветвлений, состоящее из 3 вершин - начальной 0 и вершин, обозначенных $(\overline{5,4})$ (соответствующая дуга не входит в маршрут коммивояжера) и (5,4) (входит), см. рис.



В подзадаче $P(\overline{5,4})$ дуга (5,4) не входит в маршрут. Полагаем стоимость этой дуги равной ∞. Нижняя оценка не меняется: $LB(\overline{5,4}) = LB(0) = 0$.
В подзадаче $P(5,4)$ дуга (5,4) входит в маршрут. Вычеркиваем строку 5 и столбец 4. Полагаем стоимость дуги (4,5) равной ∞. Вычисляем $\Delta_{54} = 200$ и $LB(5,4) = c_{5,4} + 200 = 790 > R = 740$. Соответствующую ветку отсекаем.
Будем ветвить подзадачу $P(\overline{5,4})$. Выбираем дугу максимальной стоимости – $(4,5)$, $c_{4,5} = 580$. В дерево ветвлений добавляем вершины (4,5) и $(\overline{4,5})$.
В подзадаче $P(\overline{4,5})$ дуга (4,5) не входит в маршрут. Полагаем стоимость этой дуги равной ∞. Нижняя оценка не меняется: $LB(\overline{4,5}) = LB(0) = 0$.
В подзадаче $P(4,5)$ дуга (4,5) входит в маршрут. Вычеркиваем строку 4 и столбец 5. Вычисляем $\Delta_{45} = 180$ и $LB(4,5) = c_{4,5} + 180 = 760 > R = 740$. Ветку отсекаем.
Будем ветвить подзадачу $P(\overline{4,5})$. Выбираем дугу максимальной стоимости – $(2,6)$, $c_{2,6} = 550$. В дерево ветвлений добавляем вершины (2,6) и $(\overline{2,6})$.
В подзадаче $P(\overline{2,6})$ дуга (2,6) не входит в маршрут. Полагаем стоимость этой дуги равной ∞. Нижняя оценка не меняется: $LB(\overline{2,6}) = LB(\overline{4,5}) = 0$.
В подзадаче $P(2,6)$ дуга (2,6) входит в маршрут. Вычеркиваем строку 2 и столбец 6. Полагаем $c_{6,2} = \infty$. Вычисляем $\Delta_{2,6} = 330$ и $LB(2,6) = c_{2,6} + 330 = 880 > R$. Ветку отсекаем.
Будем ветвить подзадачу $P(\overline{2,6})$. Выбираем дугу максимальной стоимости – $(6,2)$, $c_{6,2} = 550$.
В подзадаче $P(\overline{6,2})$ дуга (6,2) не входит в маршрут. Полагаем стоимость этой дуги равной ∞. $LB(\overline{6,2}) = 0$.

В подзадаче $P(6,2)$ дуга (6,2) входит в маршрут. Вычеркиваем строку 6 и столбец 2. Вычисляем $\Delta_{6,2} = 210$ и $LB(6,2) = c_{6,2} + 210 = 550 + 210 = 760 > R = 740$. Ветку отсекаем.
Будем ветвить подзадачу $P(\overline{6,2})$. Соответствующая таблица имеет вид:

(i,j)	1	2	3	4	5	6
1	∞	200	30	90	170	0
2	180	∞	480	290	0	∞
3	80	500	∞	110	300	0
4	120	450	80	∞	∞	0
5	180	0	380	∞	∞	520
6	0	∞	0	0	490	∞

7 Имитационное моделирование на примере метода Монте-Карло

Рассмотрим ситуацию, когда операция представляет собой случайный процесс и показателем эффективности является вероятность какого-либо события или математическое ожидание какой-либо случайной величины. Более того, предположим, что процесс невозможно описать аналитически. В этом случае для моделирования процесса применяется *имитационное моделирование*, наиболее ранним представителем которого является *метод Монте-Карло*.
Имитационное моделирование представляет собой вычислительный эксперимент, результаты которого интерпретируются с позиций математической статистики. Рассмотрим имитационное моделирование на примере метода Монте-Карло (М-К).
Идея метода М-К состоит в следующем. Вместо того, чтобы описывать случайное явление с помощью аналитических зависимостей, производится эксперимент, называемый розыгрышем, дающий случайный результат. В результате розыгрыша получаем одну реализацию случайного явления. Проведя эксперимент достаточно большое количество раз, получим статистический материал, который можно обрабатывать методами математической статистики.
Для сложных операций, в которых участвует много элементов (машин, систем, людей, коллективов) и в которых случайные факторы сложным образом взаимодействуют друг с другом, метод М-К как правило оказывается проще и адекватнее аналитического.
Рассмотрим пример, типичный для применения метода М-К. На плоскости нарисована некоторая фигура. Команда из n человек (участников операции) должна выполнить следующее. Каждый человек, не видя фигуры и не зная действий других участников, должен нарисовать на плоскости круг заданного радиуса r. Операция считается успешной, если круги покрыли не менее 50% площади фигуры. Требуется определить вероятность того, что операция будет успешной.
Решение этой задачи аналитически с помощью методов теории вероятностей чрезвычайно сложно. Значительно проще решить эту задачу методом М-К.

Для этого проведем следующий эксперимент. Разыграем координаты n точек на плоскости с помощью какого-либо механизма случайного выбора. Например, пусть m – количество точек на плоскости. Разобьем интервал $[0,1]$ на m последовательных непересекающихся частей одинаковой длины (события выбора центра круга независимы и равновероятны). С помощью ЭВМ или по таблице выберем n (псевдо)случайных чисел из $[0,1]$. Интервалы, в которые попали эти числа, указывают координаты n точек на плоскости. Нарисуем соответствующие круги и определим площадь покрытия фигуры. Если эта площать не менее 50%, то будем считать эксперимент успешным.

При большом количестве экспериментов N вероятность W успеха операции может быть приближенно оценена как частота успешных экспериментов:

$$W \approx \frac{M}{N}, \text{ где } M \text{ – число успешных экспериментов.}$$

С помощью метода М-К можно получать не только вероятности событий, но и математические ожидания случайных величин. Например, если в рассматриваемом примере требуется найти не вероятность успеха операции, а математическое ожидание $M[S]$ площади S покрытия фигуры кругами, то можно воспользоваться следующим.

В соответствии с законом больших чисел (теорема Чебышева) при большом количестве независимых опытов среднее арифметическое полученных значений случайной величины почти наверняка мало отличается от ее математического ожидания. Пусть S_i – площать покрытия фигуры в эксперименте i. Тогда

$$M[S] \approx \frac{\sum_{i=1}^{N} S_i}{N}.$$

Аналогично может быть найдена дисперсия случайной величины, т.е. мат.ожидание квадрата отклонения случайной величины от ее мат.ожидания. В нашем примере дисперсия $D[S]$ площади покрытия фигуры может быть приближенно вычислена по формуле

$$D[S] = M[(S - M[S])^2] \approx \frac{\sum_{i=1}^{N}(S_i - M[S])^2}{N}.$$

Таким образом, метод М-К имитирует влияние случайной величины на процесс проведения операции с помощью специально организованного розыгрыша или жребия. Далее проводится достаточно большое число таких розыгрышей и интересующие нас характеристики случайного явления находятся опытным путем:

вероятности – как частоты событий,

математические ожидания – как средние арифметические значения соответствующих случайных величин,

дисперсии – как среднеквадратичные отклонения случайных величин от их среднеарифметического значения.

Недостатком метода является необходимость проведения большого числа розыгрышей.

(Необходимые понятия: Функция распределения случайной величины X: $F(x) = P(0 < X < x)$. Плотность распределения $f(x)$ непрерывной случайной величины: должна существовать $f(x)$ такая, что $P(a \le X < b) = \int_a^b f(x)dx = F(b) - F(a)$. Случайная величина распределена по показательному закону, если ее плотность распределения $f(x) = \lambda e^{-\lambda x}$, $x > 0$.

8 Основы теории сложности вычислений

В теории сложности вычислений вводится понятие трудноразрешимой задачи. Существование эффективного (быстрого) алгоритма решения хотя бы одной из трудноразрешимых задач влечет существование такого рода алгоритма для любой из таких задач. Из этого следует, что для трудноразрешимых задач существование эффективных алгоритмов решения маловероятно.

Задачи распознавания и экстремальные комбинаторные задачи. Под *задачей распознавания (свойств объектов)* будем понимать некоторый общий вопрос, предполагающий ответ "да", если рассматриваемый в этом вопросе объект обладает свойствами, описанными в формулировке вопроса. Объект и его свойства описываются с помощью некоторых параметров, конкретные значения которых в общей формулировке вопроса отсутствуют. Параметрами могут быть множества, графы, числа, функции и т.п. *Пример задачи* получается при замене всех имеющихся в общей формулировке задачи параметров их конкретными значениями. Пример называется *да-примером*, если при соответствующих значениях параметров объект обладает указанными в формулировке задачи свойствами.

Экстремальная комбинаторная задача состоит в следующем. На конечном множестве X' определен функционал $F(x)$, $x \in X'$. Необходимо в заданном подмножестве X множества X' отыскать такой элемент x^*, что $F(x^*) = \min\{F(x)|x \in X\}$ (задача минимизации), либо такой x^0, что $F(x^0) = \max\{F(x)|x \in X\}$ (задача максимизации).

Сопоставим экстремальной задаче B следующую задачу распознавания B': определить, существует ли такой элемент x' в заданном множестве X, что $F(x') \leq y$ (либо $F(x') \geq y$ в случае задачи максимизации) для данного действительного числа y. Очевидно, если x^* решение задачи минимизации B, то элемент $x' \in X$ такой, что $F(x') \leq y$, существует тогда и только тогда, когда $F(x^*) \leq y$. Таким образом, экстремальная задача не проще соответствующей задачи распознавания.

Разумные схемы кодирования. Поскольку задачи интересуют нас с точки зрения разработки алгоритмов их решения и реализации этих алгоритмов на ЭВМ, входные данные задачи должны быть каким-то образом закодированы. Схема кодирования сопоставляет цепочку символов каждому примеру задачи.

Оценка вычислительной сложности алгоритма решения задачи – это функция от длины записи входной информации задачи, т.е. она непосредственно зависит от выбранной схемы кодирования.

Можно подобрать схему кодирования, которая дает очень длинные коды примеров задачи, так что почти любой алгоритм будет эффективным относительно длин таких входов и мы не сможем определить, какой алгоритм лучше. Кроме того, схема кодирования может выдавать коды существенно разной длины для разных примеров одной и той же задачи. В этом случае непонятно как определять вычислительную сложность алгоритма – на одних примерах он будет эффективным, на других нет.

Эти и другие противоречия будут устранены, если пользоваться так называемыми *разумными схемами кодирования.* Будем говорить, что схема кодирования является разумной, если она удовлетворяет следующим неформальным условиям.

(а) *Код примера задачи должен быть компактным и не должен содержать необязательную информацию или символы.*
(б) *Числа из примера задачи должны быть представлены в двоичной, либо какой-нибудь иной системе счисления с основанием, отличным от единицы.*
Система счисления с основанием, равным 1, называется унарной. В этой системе число 3 представляется в виде 111, число 4 в виде 1111 и т.д. (как зарубки у древних людей). В унарной системе счисления для представления целого числа x требуется x единиц памяти. В двоичной системе счисления все числа представляются в виде последовательностей двух чисел: 0 и 1. Число 3 представляется в виде 11, число 4 в виде 100, число 5 в виде 101 и т.д. Для представления целого числа x требуется $O(\log_2 x)$ нулей и единиц (единиц памяти). Функция $O(x)$ определяется следующим образом:

$$\lim_{x\to\infty} \frac{O(x)}{x} = const.$$

(в) *Схема кодирования должна обладать свойством декодируемости,* т.е. для каждого параметра задачи должен существовать полиномиальный алгоритм, способный выделить код этого параметра из кода любого примера задачи.
(г) *Схема кодирования должна обеспечивать однородность при кодировании различных примеров одной и той же задачи.* Однородность означает, что схема кодирования должна обеспечивать для любых двух различных примеров задачи получение кодов, "близких" по длине, если эти примеры содержат близкие по величине числа и количество параметров в обоих примерах приблизительно одно и то же.
Выполнение условия однородности может быть обеспечено соблюдением следующего простого правила. Схема кодирования не должна заниматься распознаванием специфических свойств кодируемого примера задачи. При построении кода конкретного примера задачи схема может использовать лишь те свойства, которые непосредственно указаны в формулировке задачи и являются общими для всех примеров данной задачи. Таким образом, *код примера задачи должен содержать лишь ту информацию, которую содержит общая формулировка задачи, дополненная конкретными значениями параметров задачи.*
Условие (г) предотвращает появление слишком коротких кодов для отдельных примеров задачи. С другой стороны, условие (а) не дает возможности появляться слишком длинным кодам.
Вычислительная (временная) сложность алгоритмов. Полиномиальные и псевдополиномиальные алгоритмы. Под временной сложностью алгоритма понимается количество *элементарных операций*, необходимое для его реализации. Для алгоритмов, предназначенных для реализации на ЭВМ, под элементарными понимаются такие машинные операции, как сложение, умножение, сравнение двух чисел, запись либо считывание числа, расположенного по известному адресу, и т. п.
Для любого примера I некоторой задачи введем в рассмотрение две функции:
– функция $Length[I]$, определяющая длину кода примера I при использовании разумной схемы кодирования и

– функция $Max[I]$, определяющая величину наибольшего числового параметра в I. Если задача не содержит чисел, то полагаем $Max[I] = 1$ для любого примера I.
Как уже отмечалось, оценка вычислительной сложности алгоритма решения задачи – это функция от длины записи входной информации задачи.
Алгоритм решения задачи называется *полиномиальным*, если его временная сложность не превосходит некоторого полинома от функции $Length[I]$ для любого примера I этой задачи. Соответствующая задача называется *полиномиально разрешимой*.
Полиномиально разрешимые задачи образуют класс $\mathcal{P}$.
Алгоритм решения задачи называется *псевдополиномиальным*, если его временная сложность не превосходит некоторого полинома от двух функций: $Length[I]$ и $Max[I]$, - для любого примера I этой задачи. Соответствующая задача называется *псевдополиномиально разрешимой*.
Класс $\mathcal{NP}$ задач. NP-полные и NP-трудные задачи. Задача принадлежит классу $\mathcal{NP}$, если она может быть решена за полиномиальное время на так называемой недетерминированной машине Тьюринга, которая реально не существует. На такой машине за полиномиальное время можно реализовать как полиномиальные так и некоторые неполиномиальные алгоритмы.
Задача П называется NP-*трудной*, если любая задача из класса $\mathcal{NP}$ *полиномиально сводится* к ней, т.е. имея полиномиальный алгоритм решения NP-трудной задачи, можно получить полиномиальный алгоритм решения любой задачи из класса $\mathcal{NP}$. Задача распознавания П называется NP-*полной*, если она NP-трудна и принадлежит $\mathcal{NP}$.
Понятие полиномиальной сводимости транзитивно. Поэтому для доказательства NP-трудности некоторой задачи достаточно полиномиально свести к ней некоторую NP-трудную задачу:
для любого примера известной NP-трудной задачи за полиномиальное от его размерности время построить пример нашей задачи и показать, что решение NP-трудной задачи существует тогда и только тогда, когда существует решение построенного примера нашей задачи.
Экстремальная задача называется NP-трудной, если соответствующая ей задача распознавания является NP-трудной.
Из существования полиномиального алгоритма решения какой-нибудь NP-трудной задачи следует существование полиномиальных алгоритмов решения для всех задач из класса $\mathcal{NP}$, включая все NP-полные задачи.
Как уже отмечалось, все полиномиально разрешимые задачи полиномиально разрешимы на недетерминированной машине Тьюринга. Поэтому

$$\mathcal{P} \subseteq \mathcal{NP}.$$

Однако неясно $\mathcal{P} = \mathcal{NP}$ или $\mathcal{P} \neq \mathcal{NP}$.
Поскольку ни для одной NP-трудной задачи не разработан полиномиальный алгоритм решения, в настоящее время общепринятой является гипотеза о том, что

$$\mathcal{P} \neq \mathcal{NP}.$$

Для того, чтобы ее опровергнуть, достаточно построить полиномиальный алгоритм решения любой (какой-нибудь одной) NP-трудной задачи. Список таких задач включает десятки тысяч задач из различных областей науки.
За разработку полиномиального алгоритма решения любой NP-трудной задачи установлен приз в 1.000.000 долларов.
NP-трудные в сильном смысле задачи. Наряду с разделением задач на NP-трудные и полиномиально разрешимые, NP-трудные задачи, в свою очередь, подразделяются на NP-трудные в сильном смысле задачи и задачи, имеющие псевдополиномиальные алгоритмы решения. NP-трудные в сильном смысле задачи не имеют псевдополиномиального алгоритма решения.
Примеры NP-трудных задач.
NP-трудная задача, имеюшая псевдополиномиальный алгоритм решения:
РАЗБИЕНИЕ:
Заданы целые положительные числа $a_1, a_2, \ldots, a_r$ и A такие, что $\sum_{j=1}^{r} a_j = 2A$. Требуется определить, существует ли множество $X \subset N = \{1, 2, \ldots, r\}$ такое, что $\sum_{j \in X} a_j = A$.

NP-трудная в сильном смысле задача:
3-РАЗБИЕНИЕ:
Заданы целые положительные числа $a_1, a_2, \ldots, a_{3r}$ и A такие, что $A/4 < a_j < A/2$, $j = 1, \ldots, 3r$, и $\sum_{j=1}^{3r} a_j = rA$. Требуется определить, существует ли разбиение множества $\{1, 2, \ldots, 3r\}$ на r непересекающихся подмножеств $X_1, X_2, \ldots, X_r$ такое, что $\sum_{j \in X_l} a_j = A$, $l = 1, \ldots, r$.
Нетрудно заметить, если решение задачи 3-РАЗБИЕНИЕ существует, то каждое из множеств $X_1, \ldots, X_r$ содержит в точности три элемента.

9 Динамическое программирование

Динамическое программирование (ДП) является универсальным методом решения экстремальных задач.
Существует несколько интерпретаций метода ДП. Мы рассмотрим ДП как процесс построения множеств частичных решений исходной задачи и выбора из этих множеств *доминирующих* решений таких, что хотя бы одно из них может быть "достроено" до оптимального.
При таком подходе процесс решения некоторой задачи может быть организован следующим образом. Для определенности, рассмотрим задачу минимизации

$$f(x) \to \min, \ x \in X.$$

Будем формировать множества $X_0, X_1, \ldots, X_n$ частичных решений, где множество X_{k+1} получается из множества X_k путем "достраивания" каждого решения из X_k. Оптимальное решение x^* выбирается из множества X_n.
С каждым частичным решением $x \in X_k$ будем связывать некоторый набор параметров $B = (k, b_1, \ldots, b_i)$, называемых *переменными состояния*. Набор B переменных состояния называется *состоянием*.

Пусть $X_k(B)$ обозначает множество частичных решений $x \in X_k$ в состоянии (соответствующих состоянию) B. Пусть $B(x)$ обозначает состояние частичного решения x.

В методе ДП с каждым состоянием B связывается функция *доминирования* $F(B)$ такая, что частичное решение, находящееся в состоянии B и максимизирующее (либо минимизирующее) $F(B)$, *доминирует* все остальные частичные решения в состоянии B, т.е. это частичное решение может быть достроено до полного допустимого решения с наименьшим значением целевой функции среди всех полных допустимых решений, достроенных из любого частичного решения в состоянии B.

Из определения функции $F(B)$ следует, что в каждом множестве $X_k(B)$ достаточно выбрать доминирующее решение для дальнейших построений.

Обозначим через S множество всех возможных состояний. Это множество называется *пространством состояний.*

В методе ДП рекуррентно вычисляют значения $F(B)$ и находят состояние, соответствующее оптимальному решению. Затем восстанавливают само оптимальное решение.

Для применения метода ДП необходимо корректно определить

1) пространство состояний S,

2) функцию доминирования $F(B)$, $B \in S$,

3) способ построения частичных решений $x' \in X_{k+1}$ из частичных решений $x \in X_k$,

4) метод вычисления состояния $B(x')$, используя состояние $B(x)$, если $x' \in X_{k+1}$ "достроен" из $x \in X_k$, и

5) рекуррентную формулу для вычисления значений $F(B)$.

Пример алгоритма ДП. В качестве примера рассмотрим следующую задачу. Служащий должен выполнить n работ к заданному сроку d, начиная с момента времени ноль. Для каждой работы j задана длительность выполнения p_j и штраф w_j, выплачиваемый с случае завершения j после d. Все параметры являются неотрицательными целыми числами. Требуется найти такую последовательность выполнения работ, чтобы суммарный штраф был наименьшим.

При заданной последовательности работ $(i_1, \ldots, i_n)$ легко определить момент завершения выполнения каждой работы i_j: $C_{i_j} = \sum_{k=1}^{j} p_{i_k}$.

Введем в рассмотрение переменные U_j: $U_j = 1$, если работа j является запаздывающей ($C_j > d$), и $U_j = 0$, если j является ранней ($C_j \leq d$). Требуется найти такую последовательность работ, что значение $\sum_{j=1}^{n} w_j U_j$ минимально.

Заметим, что на значение целевой функции влияет лишь информация о том, какие работы являются запаздывающими, а какие ранними. Поэтому существует оптимальная последовательность работ, в которой все ранние работы упорядочены произвольно и все запаздывающие работы упорядочены произвольно и расположены после последней ранней работы. Тогда задача может быть сформулирована следующим образом:

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

при условиях

$$\sum_{j=1}^{n} p_j(1-U_j) \le d,$$

и

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

Эта задача NP-трудна.

Пусть $U^* = (U_1^*,\dots,U_n^*)$ – оптимальный вектор для этой задачи и W^* – соответствующее значение целевой функции.

Сформулированная задача, в свою очередь, может быть проинтерпретирована в терминах ДП следующим образом.

Будем формировать частичные решения, определяя переменные $U_j = 0$ либо $U_j = 1$ для $j = 1,\dots,n$. Будем говорить, что частичное решение $(U_1,\dots,U_k)$ находится в состоянии (k,a), если определены значения переменных $U_1,\dots,U_k$ и значение ограничения $\sum_{j=1}^{k} p_j(1-U_j) = a$.

Переменные состояния могут принимать значения $k = 0,1,\dots,n$ и $a = 0,1,\dots,d$.

Из начального состояния $(0,0)$ можно перейти в состояние $(1,0)$, при котором $U_1 = 1$, либо в состояние $(1,p_1)$, где $U_1 = 0$, если $p_1 \le d$.

Рассмотрим некоторое состояние (k,a), $k \in \{2,\dots,n\}$. В это состояние можно перейти только из состояния $(k-1,a)$, если $U_k = 1$, либо из состояния $(k-1,a-p_k)$, если $U_k = 0$.

Определим функционал доминирования $F(k,a)$ как значение целевой функции $\sum_{j=1}^{k} w_j U_j$, вычисленное для решений в состоянии (k,a). Из определения функционала $F(k,a)$ следует, что

$$W^* = \min\{F(n,a) | a = 0,1,\dots,d\}.$$

Значения $F(k,a)$ могут быть вычислены рекуррентно. Для инициализации рекуррентных вычислений положим $F(0,0) = 0$ и $F(k,a) = \infty$ для всех $(k,a) \ne (0,0)$.

Из определения функции $F(k,a)$ следует, что общее *рекуррентное соотношение* может быть записано как

$$F(k,a) = \min\{F(k-1,a) + w_k, F(k-1,a-p_k)\}, \tag{2}$$

$$k = 1,\dots,n,\ a = 0,1,\dots,d.$$

При этом, если $F(k,a) = F(k-1,a) + w_k$, то в частичном решении, соответствующем состоянию (k,a), переменная $U_k = 1$. Если же $F(k,a) = F(k-1,a-p_k)$, то в этом решении $U_k = 0$.

Обратный ход. Оптимальное решение U^* может быть найдено с помощью следующей процедуры, называемой *обратный ход.* Эта процедура использует таблицу размерности $n \times (d+1)$, в каждой ячейке (k,a) которой хранится информация о том, на первой или второй компоненте достигается минимум в (2) для указанных значений k и a. На каждой итерации этой процедуры проверяется, на первой или второй компоненте достигается минимум в (2) для определенных значений k и a. На первой итерации полагаем $k = n$ и $a = a^*$, где a^* определяется из $W^* = F(n,a^*)$. Если минимум в (2) достигается на первой компоненте, то полагаем $U_k^* = 1$, $k := k-1$, оставляем a без изменений и переходим

к следующей итерации процедуры обратного хода. Если минимум достигается на второй компоненте, то полагаем $U_k^* = 0$, $a := a - p_k$, $k := k-1$, и переходим к следующей итерации. После выполнения n итераций будет построен оптимальный вектор $U^* = (U_1^*, \ldots, U_n^*)$.

Время работы и требуемая память. Эффективность алгоритма ДП определяется его временной сложностью и объемом требуемой памяти. Анализ алгоритма для рассматриваемой задачи показывает, что вычисление $F(k, a)$ требует выполнения одной операции сложения и одной операции сравнения для каждой пары (k, a). Таким образом, временная сложность этого алгоритма равна $O(nd)$.

Что касается объема памяти, то следует отметить, что для вычисления оптимального значения целевой функции W^* на каждом шаге k достаточно хранить таблицу размерности $2 \times (d+1)$ со значениями $F(k-1, a)$ и $F(k, a)$, $a = 0, 1, \ldots, d$. Для отыскания оптимального вектора U^* достаточно применить процедуру обратного хода, которая требует $n(d+1)$ дополнительных единиц памяти.

Из приведенного анализа можно сделать вывод, общий для всех алгоритмов ДП: *временная сложность и объем требуемой памяти алгоритма ДП непосредственно зависят от мощности соответствующего пространства состояний.*

Выбор пространства состояний. Прямые и обратные алгритмы ДП. Как правило, существует несколько вариантов для выбора пространства состояний. Например, для рассматриваемой задачи можно ввести состояния (k, w) такие, что частичное решение находится в состоянии (k, w), если определены значения переменных $U_1, \ldots, U_k$ и $\sum_{i=1}^{k} w_i U_i = w$. В этом случае функционал $F(k, w)$ определяется как наименьшее значение ограничения $\sum_{i=1}^{k} p_i(1 - U_i)$ для решений в состоянии (k, w). Рекуррентное соотношение можно записать в виде

$$F(k, w) = \min \begin{cases} F(k-1, w) + p_k, & \text{если } F(k-1, w) + p_k \le d, \\ F(k-1, w - w_k) \end{cases}$$

Соответствующий алгоритм ДП требует $O(n \sum_{k=1}^{n} w_k)$ единиц времени и памяти.

Алгоритмы ДП, в которых частичные решения формируются от начала к концу, как это делается в приведенных выше алгоритмах, называются *прямыми*. Альтернативный подход состоит в построении частичных решений от конца к началу. Такие алгоритмы называются *обратными*. Например, рассматриваемая задача может быть решена в пространстве состояний (k, a) следующим образом. Полагаем $F(n+1, 0) = 0$, все остальные начальные значения $F(k, a)$ равными бесконечности и вычисляем

$$F(k, a) = \min\{F(k+1, a - p_k), F(k+1, a) + w_k\}$$

для $k = n, n-1, \ldots, 1$ и $a = 0, 1, \ldots, d$. Оптимальное значение целевого функционала равно $W^* = \min\{F(1, a) |\ a = 0, 1, \ldots, d\}$.

Числовой пример решения задачи о рюкзаке методом ДП.

$$F(x) = x_1 + 2x_2 + x_3 + x_4 + 2x_5 + x_6 + 2x_7 \Rightarrow \max,$$

$$B(x) = 2x_1 + x_2 + 2x_3 + 2x_4 + x_5 + x_6 + x_7 \le 5,\ x_i \in \{0, 1\}.$$

X_0	(0)
$F(x)$	0
$B(x)$	0

X_1	(0)	(1)
$F(x)$	0	1
$B(x)$	0	2

X_2	(0,0)	(0,1)	(1,0)	(1,1)
$F(x)$	0	2	1	3
$B(x)$	0	1	2	3

X_3	(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)
$F(x)$	0	1	2	3	1	2	3	4
$B(x)$	0	2	1	3	2	4	3	5

3 предпоследние частичные решения решения можно дальше не рассматривать, так как есть доминирующие. Далее достраиваем только доминирующие.

X_4	(0,0,0,0)	(0,0,0,1)	(0,0,1,0)	(0,0,1,1)	(0,1,0,0)	(0,1,0,1)	(0,1,1,0)	(0,1,1,1)	(1,1,1,0)	(1,1,1,1)
$F(x)$	0	1	1	2	2	3	3	4	4	5
$B(x)$	0	2	2	4	1	3	3	5	5	7

X_5	(0,0,0,0,0)	(0,0,0,0,1)	(0,0,0,1,0)	(0,0,0,1,1)	(0,1,0,0,0)	(0,1,0,0,1)	(0,1,1,1,0)	(0,1,1,1,1)
$F(x)$	0	2	1	3	2	4	4	6
$B(x)$	0	1	2	3	1	2	5	6

X_6	(0,0,0,0,0,0)	(0,0,0,0,0,1)	(0,0,0,1,0,0)	(0,0,0,1,0,1)	(0,0,0,1,1,0)	(0,0,0,1,1,1)	(0,1,0,0,1,0)	(0,1,0,0,1,1)
$F(x)$	0	1	2	3	3	4	4	5
$B(x)$	0	1	1	2	3	4	2	3

X_7	(0,0,0,0,0,0,0)	(0,0,0,0,0,0,1)	(0,0,0,0,0,1,0)	(0,0,0,0,0,1,1)	(0,0,0,1,0,0,0)	(0,0,0,1,0,0,1)
$F(x)$	0	2	1	3	2	4
$B(x)$	0	1	1	2	1	2

X_7	(0,0,0,1,0,1,0)	(0,0,0,1,0,1,1)	(0,1,0,0,1,0,0)	(0,1,0,0,1,0,1)	(0,1,0,0,1,1,0)	**(0,1,0,0,1,1,1)**
$F(x)$	3	5	4	6	5	**7**
$B(x)$	2	3	2	3	3	4

Оптимальное решение $x^* = (0, 1, 0, 0, 1, 1, 1)$ со значением целевой функции $F(x^*) = 7$.