ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА
(РОСАВИАЦИЯ)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра управления воздушным движением

Ю.С. Карчевский

ТЕОРИЯ УПРАВЛЕНИЯ ВОЗДУШНЫМ ДВИЖЕНИЕМ

ТЕОРИЯ УВД. ЧАСТЬ 2

Учебно-методическое пособие
по изучению дисциплины

для студентов II курса
направления 25.03.03
очной формы обучения

Москва
ИД Академии Жуковского
2025

УДК 351.814.3
ББК 0580.3
К27

Рецензент:

Печенежский В.К. – канд. техн. наук, доцент

Карчевский Ю.С.

К27 Теория управления воздушным движением. Теория УВД. Часть 2 [Текст] : учебно-методическое пособие по изучению дисциплины / Ю.С. Карчевский. – М.: ИД Академии Жуковского, 2025. – 40 с.

Данное учебно-методическое пособие издается в соответствии с рабочей программой учебной дисциплины «Теория управления воздушным движением» для студентов II курса направления 25.03.03 очной формы обучения.

Рассмотрено и одобрено на заседаниях кафедры 12.02.2025 г. и методического совета 28.02.2025 г.

УДК 351.814.3
ББК 0580.3

В авторской редакции

Подписано в печать 23.05.2025 г.
Формат 60x84/16 Печ. л. 2,5 Усл. печ. л. 2,325
Заказ № 1081/0325-УМП17 Тираж 25 экз.

Московский государственный технический университет ГА
125993, Москва, Кронштадтский бульвар, д. 20

Издательский дом Академии имени Н. Е. Жуковского
125167, Москва, 8-го Марта 4-я ул., д. 6А
Тел.: (499) 755-55-43
E-mail: zakaz@itsbook.ru

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дискретные задачи оптимизации над конечными множествами имеют конечное множество допустимых решений, которые можно перечислять и выбрать из них наилучшее, обеспечивающее получение экстремума целевой функции. Предметная область таких задач – процессы исследования операций, теория которой формируется уже несколько десятилетий.

Методы решения задач, которыми теория располагает сегодня – это, прежде всего, неклассические методы математического программирования, линейное, нелинейное и динамическое программирование, целочисленное и булево, геометрическое и выпуклое программирование, принципы оптимальности, минимаксные и максиминные методы и многие другие.

В самом деле, все рассматриваемые дискретные задачи всегда имеют конечное решение, которое можно получить простым перебором вариантов.

Другой вопрос, во что это выльется по затратам и по времени. Поэтому подавляющее большинство методов исследования операций ориентировано на сокращение переборов, где только это возможно.

В данном учебном пособии приведены алгоритмы и примеры решения основных практических задач Теории УВД таких как транспортная задача, задача о максимальном потоке в сети, задача о назначениях и другие.

Материалом для учебного пособия является курс лекций, прочитанных автором для студентов 2-го курса факультета управления на воздушном транспорте МГТУ ГА по направлению подготовки 25.03.03 «Аэронавигация».

1 ПОНЯТИЕ ДЕТЕРМИНИРОВАННЫХ ЗАДАЧ

Детерминированные задачи – это задачи, при которых считается, что каждая выбираемая руководителем (ЛПР) стратегия приводит к единственному, заранее известному результату.

Все задачи делятся на два типа: прямые и обратные.

Прямые отвечают на вопрос: что будет, если при заданных условиях, мы примем какое-то решение, то есть чему будет равен при данном решении показатель эффективности.

Для решения такой задачи строится математическая модель, позволяющая выразить показатель эффективности W через заданное условие α и решения (x), где:

X – не число, а группа параметров;

α – вектор условий, заранее известные факторы, обычно равенства или неравенства.

Обратные задачи отвечают на вопрос: как выбрать решение для того, чтобы показатель эффективности W обратился в max/min.

$W_{ОПТ}$ max/min – значение целевой функции, состоящей из всех значения W, то есть перед нами типичная математическая задача нахождения экстремума функции.

Если число возможных решений не велико, то можно просто вычислить W для каждого из них, сравнить между собой полученные значения W и непосредственно указать один (или несколько) оптимальных вариантов, для которых показатель эффективности W достигает max/min.

Такой способ нахождения оптимального решения – *простой перебор*. Но когда число возможных вариантов решений велико, поиск среди них оптимального простым перебором (то есть вслепую) затруднителен.

В этом случае применяют метод *направленного перебора*, если функция W имеет производную, то это – **вариационная задача**, самой простой из которых является задача на экстремумы (то есть задача на max/min).

Вариационные задачи решаются дифференцированием функциями W по всем аргументам X (то есть элементам решения).

Для чего приравниваем значения производных к 0 и решаем полученную систему алгебраических уравнений.

В зависимости от вида функции W и ограничений, накладываемых на элементы решения X, возникают разные задачи:

1. Если функция W линейно зависит от X, а ограничения, накладываемые на решение, имеют вид линейных равенств, возникает классическая задача линейного программирования (прогнозирования).

2. Если функция W выпукла, то применяются методы нелинейного программирования, в частности, квадратичного.

Для многоэтажного решения задачи применяется метод динамичного программирования (прогнозирования).

Таким образом, нахождение оптимального решения в детерминированном случае может представлять лишь вычислительные трудности.

Совсем по-другому дело обстоит, если задача содержит элемент неопределенности.

1.1 Выбор решения в условиях неопределенности

В реальных задачах показатель эффективности W (целевая функция) зависит от неизвестных фактов: W = W (x, α, … ω) – группа неизвестных фактов. Поэтому теперь, даже при заданных α и x, она уже не может быть вычислена и остается неопределенной.

Соответственно и задача поиска оптимального решения также теряет определенность, так как нельзя максимизировать неизвестную величину W. Понятно, что любое решение, принятое в условиях неопределенности хуже решения, принятого в заранее известных условиях.

Для того, чтобы решение принимать не наобум, существует ряд приемов решения задач по оптимизации, зависящих от природы неизвестных факторов, другими словами, мы должны определить с неопределенностью какого типа мы сталкиваемся в этой задаче.

Для этого необходимо классифицировать неопределенности.

Прежде всего, рассмотрим благоприятный для нас случай, когда неизвестные факторы показателя эффективности W являются случайными величинами и при этом имеют известные нам статистические характеристики (или их характеристики могут быть получены к нужному моменту времени).

Такие задачи называются *вероятностными*, а неопределенности, им присущие, называются *вероятностными неопределенностями*.

Например: пусть реорганизуется работа диспетчерского пункта с целью повышения его пропускной способности.

Нам не известно, какое количество воздушных судов (ВС) пройдет через данный диспетчерский пункт, когда они будут появляться, сколько времени будет обслуживаться каждое из них. Статистические характеристики нам неизвестны, но они могут быт получены путем обработки статистических данных.

Максимизировать или минимизировать случайную величину невозможно, при любых x она остается случайной, неконтролируемой.

Как быть?

Рассмотрим основные подходы, применяемые для разрешения подобной ситуации:

Если **неизвестные** случайные факторы мало отклоняются от своих средних значений, то их можно заменить средними значениями, и задача сводится к детерминированному случаю, то есть случайностью мы пренебрегаем.

Если **неизвестные** факторы существенно случайны, то нужно взять в качестве целевой функции (показателя эффективности W) его средние значения и выбрать такое решение X, при котором этот усредненный по условию α показатель обращается в max/min.

$W = M [W (x, \alpha, \dots \omega)] \rightarrow \max/\min,$

где M – математическое ожидание случайной величины W.

Такой подход называется *оптимизацией в среднем*.

Например, пусть в зоне ожидания находятся несколько ВС, если выбрать показателем эффективности (целевой функцией) время T, то, казалось бы, надо

выбрать такое решение, при котором среднее время ожидания будет min, однако, это не верно.

Дело в том, что время ожидания отдельных ВС не суммируется, слишком долгое ожидание одного из них не компенсируется почти мгновенно, обслуживанием другого, поэтому необходимо дополнить показатель эффективности дополнительным требованием: $P(T <= t_0) >= 0{,}995$,

где: P – вероятность события;

t_0 – постоянная, ограничивающая время пребывания ВС в зоне ожидания.

3. Случай, когда неизвестные факторы, не могут быть описаны статистическими методами.

Такая ситуация бывает в двух случаях:

1. Распределение вероятностей для случайных параметров к моменту принятия решения не может быть получено.

2. Распределения вероятностей для случайных параметров вообще не существует.

Пример 1: проектируется АС УВД, предназначенная дня обслуживания потоков воздушного движения (ВД), то есть она будет обслуживать случайные транспортных потоки, вероятностные характеристики этих потоков могли бы быть получены из статистики, если бы данная информационная система уже существовала и функционировала достаточно долгое время, но к моменту создания системы такой информации нет, а решение принимать надо.

Как поступить?

Применяем следующий прием – оставляем некоторые элементы аргумента X свободными, изменяемыми, затем выбираем *для начала* какой-то вариант решения, заведомо зная, что он не является оптимальным, и запускаем систему в работу, затем, по мере накопления опыта (статистических данных), целенаправленно изменяем свободные элементы решения, добиваясь, чтобы эффективность увеличивалась.

Такие совершенствующиеся в процессе применения алгоритмы называются *адаптивными*.

По мере накопления опта, такой алгоритм постепенно улучшается, подобно тому, как человек учится на ошибках.

Пример 2: разберем вторую ситуацию, когда у неопределенных факторов вообще не существует вероятностных характеристик.

Одним из методов решения подобных задач является *метод экспертных оценок*.

Идея метода состоит в следующем: собирается коллектив компетентных в данной области людей (экспертов), каждому из них предлагается ответить на какой-то конкретный вопрос, например оценить вероятность того или другого события, затем полученные ответы обрабатываются на подобие статистического материала, не смотря на субъективный характер оценок каждого эксперта. Усредняя оценки целого коллектива, можно набрать необходимую статистику и свести задачу к обычной вероятностной задаче.

Подводя итог, можно отметить, что при обосновании решения *в условиях неопределенности*, при любых наших действиях, элемент неопределенности сохраняется, поэтому нельзя предъявлять к точности решения слишком высокие требования, и вместо того, чтобы указать одно единственное решение, лучше выделить целую область приемлемых решений, которая оказывается несущественно лучше других. В пределах этой области и должны производить свой окончательный выбор лица, принимающие решения (ЛПР).

1.2 Многокритериальные задачи

Дня многих задач эффективность мне может быть оценена с помощью одного единственного показателя W_i. На помощь ему необходимо привлекать другие дополнительные показатели, поэтому задача становится многокритериальной.

Пример: реорганизуется работа центра АС УВД, стоит вопрос – как выбрать правильное решение? Чем при этом руководствоваться?

С одной стороны, нам бы хотелось повысить уровень безопасности полетов (БП), желательно увеличить пропускную способность ($\mu_З$) зоны секторов, а также пропускную способность ($\mu_{РА}$) района аэродрома. Затраты при этом хотелось бы минимизировать.

Вопрос: можно ли найти решение, одновременно удовлетворяющее всем требованиям. Сразу же можно ответить, что нет, т.к. решение, обращающее в максимум какой-то один показатель, как правило, не обращает ни в максимум, ни минимум другой.

Как же быть в этом случае? Как оценить эффективность решения по нескольким показателям?

1. Одним из **некорректных** подходов является путь сведения задачи к однокритериальной: составляют какую-то функцию, зависящую от всех показателей, и рассматривают ее как один обобщенный показатель, по которую оптимизируют решение.

Часто такой показатель имеет вид дроби, в числителе которой стоят все величины, увеличение которых желательно, а в знаменателе те, увеличение которых не желательно.

Этот способ объединения показателей в один нельзя рекомендовать, так как он основан на неявном допущении, что недостаток в одном показатель всегда может быть скомпенсирован за счет другого, а это неверно!

2. Другой **некорректный** подход заключается в составлении объединенного показателя эффективности, который представляет собой взвешенную сумму частных показателей:

$W(x, \alpha, \ldots \omega) = y_1 \times W_1 + y_2 \times W_2 + \ldots + y_n \times W_n$,

где $y_{1\ldots n}$ – весовые коэффициенты.

При использовании такого показателя нужно знать весовые коэффициенты y_n, а это не всегда возможно.

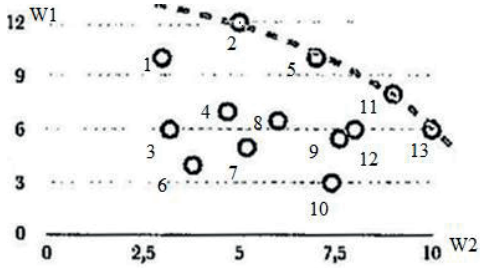
Произвольный выбор коэффициентов y_n приведет к тому, что оптимальное решение будет произвольным. Но подобный обобщенный показатель можно использовать для решения отдельных конкретных задач. Например, для оценки эффективности управления диспетчерами, так как коэффициенты $y_{1\ldots n}$ могут

быть получены на этапе обучения диспетчеров на тренажере при проигрывании типовых ситуаций, возникающих при УВД.

Как же решить *многокритериальную задачу*?

Для примера рассмотрим двухкритериальную задачу, то есть будем иметь две целевые функции W_1 и W_2.

Продемонстрируем данную ситуацию при помощи рисунка



Множество X состоит из конечного числа (13) возможных решений: x_1, x_2, x_3 … x_{13}.

Каждому решению соответствуют определенные значения показателя эффективность W_1 и W_2.

Изобразим каждое решение точкой на плоскости с координатами W_1 (ось ординат) и W_2 (ось абсцисс) и занумеруем точки соответственным номером решения.

Очевидно, что из всего множества X эффективными будут только решения, лежащие на правой верхней границе возможных решений (пунктирная линия).

Для всякого другого решения существует хотя бы одно доминирующее, для которого либо W_1, либо W_2, либо оба не пригодны. И только для решений, лежащих на правой верхней границе, доминирующих решений не существует, теперь ЛПР может выбрать тот вариант, который его интересует.

Существует один, часто применяемый способ, сведения многокритериальной задачи к однокритериальной – выделить один главный показатель эффективности W_i и обратить его в максимум, а на все остальные установить соответствующие ограничения.

При оптимизации зоны УВД можно потребовать увеличение пропускной способности ВС (μ) при сохранении существующей интенсивности ВД (λ).

При таком подходе все показатели переходят в разряд заданных условий (α).

Существует еще один путь построения компромиссного решения, который можно назвать *методом последовательных уступок*.

Предположим, что показатели W_1, W_2, … W_N расположены в порядке убывания важности.

Тогда, сначала оптимизируется самый важный критерий (ищем решение, обращающее в максимум первый важный показатель $W_1 \rightarrow \max$) и определяем его самое наилучшее значение W_1^* (*идеал*).

На следующем шаге допускается некоторое фиксированное ухудшение от этого оптимального значения на величину ΔW_1 (*уступка*) с целью улучшения

ситуации по второму по значимости критерию (для того, чтобы максимизировать показатель W_2).

Получаем *условный идеал* второго критерия W_2^*. Далее допускается уступка от него с целью оптимизации третьего критерия и так далее.

При уступке для максимизируемого критерия, от его идеала отнимаем соответствующую уступку $W_1 \geq W_1^* - \Delta W_1$.

При уступке для минимизируемого критерия – добавляем величину уступки: $W_1 \leq W_1^* + \Delta W_1$.

Обратите внимание на знаки неравенств и знаки перед уступками для максимизируемых и минимизируемых критериев.

Те значения вектора X^*, при которых достигается последнее оптимальное значение заданного параметра, и являются решением задачи многокритериальной оптимизации по методу последовательных уступок.

Данный способ хорош тем, что здесь сразу видно ценой какой уступки в одном показателе приобретается выигрыш и другом, и какова величина этого выигрыша.

2 МЕТОДЫ РЕШЕНИЯ ДЕТЕРМИНИРОВАННЫХ ЗАДАЧ

2.1 Линейное программирование (прогнозирование)

Значительная часть процессов может быть описана статическими математическим моделями, в который область допустимых значений (ОДЗ) вектора состояния процесса задается линейными зависимостями в виде неравенств или уравнений, то есть вектор состояний $X\{x_1, x_2, \ldots x_n\}$, целевая функция $W(x)$, ограничение в виде равенств $g_i(x) = 0$ и неравенств $h_i(x) = 0$ представляет собой линейные функции, то такая задача называется *задачей линейного программирования.*

Если же хотя бы одна из функций нелинейная, то рассматриваемая задача называется *задачей нелинейного программирования*.

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

Основными методами оптимизации статических моделей и процессов являются:

- метод линейного программирования;
- симплекс метод;
- венгерский метод.

Начнем с метода линейного программирования.

Линейное программирование – представляет собой математический метод для определения путей достижения лучших результатов (таких как максимальная прибыль и минимальная стоимость).

Линейное программирование является частным случаем математического программирования.

Особенности задач линейного программирования (ЗЛП):

- область допустимых решений задачи линейного программирования ограничена прямыми линиями, т.е. является многогранником;

– если решение ЗЛП единственно, то оно лежит в одной из вершин многогранника. Если решений множество, то они лежат на грани многогранника.

В ЗЛП может и не существовать решений, если область допустимых решений не замкнута.

Существуют два наиболее распространенных способа решения (ЗЛП): **графический метод** и **симплекс-метод.**

Графический метод существенно нагляднее и обычно проще для понимания и решения. Рассмотрим его более подробно.

2.2 Геометрическая интерпретация ЗЛП

Алгоритм решения:

1. Для каждого неравенства из системы ограничений на координатной плоскости начертить прямую.
2. Определить область допустимых значений.
3. Из точки координаты (0;0) провести вектор градиента, координаты градиента будут коэффициентом при переменных в выражении целевой функции.
4. Начертить линию-перпендикуляр к вектору градиента (линия уровня).
5. Перемещать линию уровня по направлению к линии градиента при поиске max и в обратном направлении при поиске min (если max – то от начала координат и наоборот).
6. Определить последнюю точку касания линии уровня и области допустимых значений переменных x_1, x_2 .
7. Подставить полученные координаты в выражение целевой функции.

2.2.1 Решение задачи линейного программирования графическим методом

$F = 3x_1 + 4x_2 \rightarrow \max$

$4x_1 + x_2 \leq 8$ (1)

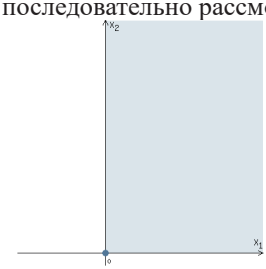
$x_1 - x_2 \geq -3$ (2)

$x_1 + x_2 \leq 2$ (3)

$x_1, x_2 \geq 0$

Обозначим условия ограничений как (1) (2) и (3).

Построим область допустимых значений (ОДЗ), для чего сразу построим систему координат. Очевидно, для нахождения области допустимых решений данной задачи, необходимо последовательно рассмотреть каждое неравенство.



Начнем с 1-го неравенства. Если знак ≤ заменить на знак равенства, то мы получим уравнение прямой, которую, как известно, можно построить по 2-м токам. Для этого сначала x_1 принимаем = 0, затем так же приравниваем к 0 x_2 и подставляем эти значения последовательно в уравнение (1).

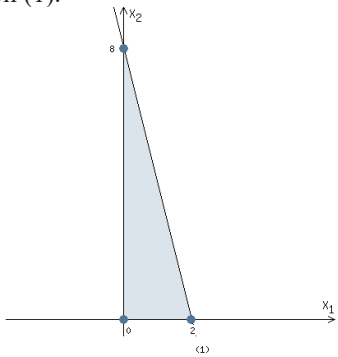
Вернемся к исходному неравенству.

$4 x_1 + x_2 \leq 8$

Преобразуем неравенство, оставив в левой части только x_2

$x_2 \leq -4 x_1 + 8$

Знак неравенства ≤. Следовательно, нас интересуют точки расположенные ниже построенной прямой (1).



Снова возьмём точку сначала координат (0; 0) и посмотрим, удовлетворяет ли она неравенству 2: 0 – 0 ≥ -3.

Неравенство выполняется, значит, точка с координатами (0; 0) принадлежит ОДЗ.

Вернемся к исходному неравенству.

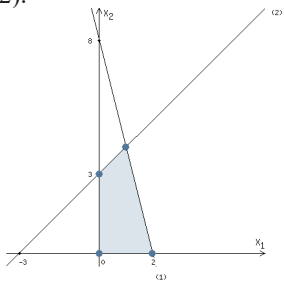
$x_1 - x_2 \geq -3$

Преобразуем неравенство, оставив в левой части только x_2

$- x_2 \geq - x_1 - 3$

$x_2 \leq x_1 + 3$

Знак неравенства ≤. Следовательно, нас интересуют точки расположенные ниже построенной прямой (2).



Перейдем к 3-му неравенству: $0 + 0 \leq 2$

Построим прямую: $x_1 + x_2 = 2$

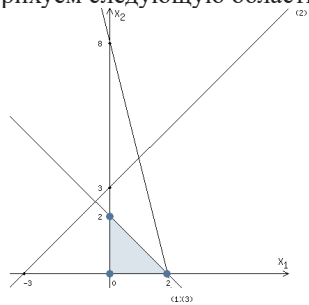
Нас интересуют точки расположенные выше или ниже построенной прямой (3)

Знак неравенства ≤. Следовательно, нас интересуют точки расположенные ниже построенной прямой (3).

Неравенство выполняется, значит, точка с координатами (0; 0) принадлежит ОДЗ.

Объединим данное условие с предыдущим рисунком. В итоге получим область допустимых решений, изображенную на рисунке.

Напомню, что еще должно выполняться условие не отрицательности переменных, поэтому заштрихуем следующую область.



Теперь в этой области необходимо найти точку, в которой целевая функция F принимает наибольшее значение:

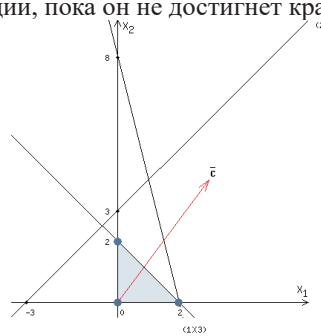
$$F = 3x_1 + 4x_2 \rightarrow \max$$

Строим вектор C = (3, 4), координатами которого являются коэффициенты функции F, обозначим его как вектор C. Он называется *целевым вектором* и показывает направление роста функции:

grad F = {3; 4} = C.

Начертим этот вектор на графике.

Теперь проведем к этому вектору перпендикуляр и начнем его сдвигать по направлению роста функции, пока он не достигнет крайней точки области.

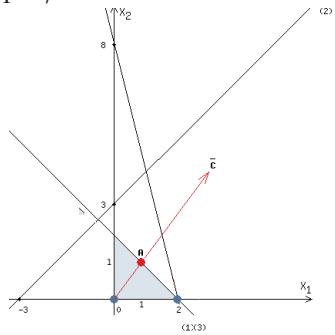


Будем перемещать «красную» прямую, перпендикулярно вектору C, от левого нижнего угла к правому верхнему.

«Красная» прямая называется линией уровня. В каждой точке линии уровня значение функции F есть величина постоянная.

В точке, в которой «красная» прямая в первый раз пересечет область допустимых решений, функция F достигает своего наибольшего значения (см. рисунок). Обозначим эту точку буквой A.

Координаты точки A (1,1) известны. (см. шаг 3)
Вычислим значение функции F в точке A (1,1).
F (A) = 3 × 1 + 4 × 1 = 7



Ответ: $F_{max} = F(1; 1) = 7$

2 РЕШЕНИЕ «ТРАНСПОРТНОЙ ЗАДАЧИ»

2.1 Постановка транспортной задачи. Транспортная таблица

Имеется *m* поставщиков $A_1, A_2, ..., A_m$, у которых сосредоточены запасы одного и того же груза в количестве $a_1, a_2, ..., a_m$ единиц, соответственно. Этот груз нужно доставить *n* потребителям $B_1, B_2, ..., B_n$, заказавшим $b_1, b_2, ..., b_n$ единиц этого груза, соответственно. Известны также все *тарифы перевозок* груза C_{ij} (стоимость перевозок единицы груза) от поставщика A_i к потребителю B_j.

Требуется составить такой план перевозок, при котором общая стоимость всех перевозок была бы минимальной.

Обозначим суммарный запас груза у всех поставщиков символом *a*, а суммарную потребность в грузе у всех потребителей — символом *b*.

Тогда

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

Условие транспортной задачи удобно записать в виде следующей *Транспортной таблицы 1*.

Таблица 1

запасы \ заказы		B_1	B_2	...	B_n
		b_1	b_2	...	b_n
A_1	a_1	c_{11}	c_{12}	...	c_{1n}
A_2	a_2	c_{21}	c_{22}	...	c_{2n}
...	...	...	...	...	...
A_m	a_m	c_{m1}	c_{m2}	...	c_{mn}

Транспортная задача – это задача линейного программирования (ЗЛП), имеющая следующий вид:

$$z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

Функция Z называется *целевой функцией*.

Матрица X с неотрицательными элементами $x_{i,j}$ называется *планом перевозок.*

Существует 2 вида ТЗ: *закрытая* (сбалансированная), когда от поставщиков будут вывезены все запасы груза, и все заявки потребителей будут удовлетворены.

И *открытая*, когда баланс нарушен.

В случае *открытой задачи* при $a < b$ весь груз будет вывезен, однако будут недопоставки груза *экономически невыгодным* потребителям. При $a > b$, наоборот, будут удовлетворены все потребители, но часть груза останется на складах *экономически невыгодных* поставщиков.

Математическая формулировка транспортной задачи заключается в нахождении плана перевозок $X = \{x_{ij}\}$, который удовлетворяет *системе ограничений:*

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i, i = 1,2,...,m, \\ \sum_{i=1}^{m} x_{ij} = b_j, j = 1,2,...,n, \end{cases} \qquad (4)$$

и доставляет минимум целевой функции *Z*.

План перевозок, реализующий минимум целевой функции *Z*, называется *оптимальным.*

Смысл первой группы равенств в системе ограничений (4) состоит в том, что суммарное количество груза, отправленное всем потребителям каждым поставщиком, равно запасу груза у этого поставщика.

Вторая группа равенств в системе ограничений (4) показывает, что суммарное количество груза, полученное каждым потребителем от всех поставщиков, равно заказу этого потребителя.

Сведение открытой транспортной задачи к закрытой

Решение транспортной задачи начинается с выяснения вопроса о том, является ли задача открытой или закрытой.

Если задача является открытой, то необходимо провести *процедуру закрытия задачи.* С этой целью при $a < b$ добавляем *фиктивного поставщика* A'_{m+1} с запасом груза $a'_{m+1} = b - a$.

Если же $a > b$, то добавляем *фиктивного потребителя* B'_{n+1} с заказом груза $b'_{n+1} = a - b$.

В обоих случаях соответствующие фиктивным объектам тарифы перевозок C_{ij} полагаем равными *нулю*. В результате суммарная стоимость перевозок Z не изменяется.

Первоначальный план перевозок

Транспортная задача относится к задачам линейного программирования, и ее можно было бы решить симплекс-методом. Но т.к. система ограничений транспортной задачи проще, чем система ограничений основной задачи линейного программирования (ОЗЛП), то это дает возможность вместо использования объемных симплекс-таблиц применить более удобный метод, который состоит из следующих этапов:

1. Составление первоначального плана перевозок.
2. Последовательные улучшения плана перевозок (перераспределение поставок) до тех пор, пока план перевозок не станет оптимальным.

Решение ОЗЛП начинается с нахождения опорного плана.

Для транспортной задачи такой план всегда **существует**!!!

Рассмотрим два метода составления *опорного плана (первоначального плана перевозок).*

При этом ненулевые элементы x_{ij} плана перевозок будем записывать в соответствующие пустые клетки транспортной таблицы с исходными данными задачи, а символом *(i, j)* будем обозначать клетку таблицы, содержащую информацию о перевозках груза от поставщика A к потребителю B.

2.2 Составление первоначального плана перевозок с помощью метода северо-западного угла

Составление первоначального плана перевозок начнем с перевозки запасов поставщика A_1. Будем за счет его запасов максимально возможно удовлетворять заказы сначала потребителя B_1, затем B_2 и так далее.

Таким образом, мы будем заполнять таблицу, начиная с клетки (1,1), и двигаться вправо по строке до тех пор, пока остаток запасов поставщика A_1 не окажется меньше заказа очередного потребителя. Для выполнения этого заказа используем остатки запаса первого поставщика, а недостающую часть добавим

из запасов поставщика A_2, то есть переместимся на следующую строку таблицы по столбцу, соответствующему указанному потребителю.

Далее аналогичным образом распределим запасы поставщика A_2, затем A_3 и так далее.

Проиллюстрируем это на следующем примере (Таблица 2).

Таблица 2

запасы \ заказы		B_1	B_2	B_3	B_4
		100	40	80	60
A_1	160	4 **100**	8 **40**	10 **20**	5
A_2	30	4	6	2 **30**	3
A_3	90	4	4	6 **30**	5 **60**

Теперь мы можем подсчитать общую стоимость всех перевозок по данному плану:

$Z = 4 \times 100 + 8 \times 40 + 10 \times 20 + 2 \times 30 + 6 \times 30 + 5 \times 60 = 1460$. Изложенный метод северо-западного угла прост в реализации, однако трудно надеяться, что он даст экономичный первоначальный план, поскольку при распределении перевозок мы совершенно не учитывали их стоимость.

2.3 Составление первоначального плана перевозок с помощью метода наименьшей стоимости

Построение плана начнем с клетки с наименьшим тарифом перевозок. При наличии нескольких клеток с одинаковыми тарифами – выбираем любую из них. Пусть это будет клетка (i, j).

Запишем в эту клетку элемент $X_{ij} = min(a_i, b_j)$.

Если $a_i < b_j$, то запасы поставщика A_i исчерпаны, а потребителю B_j требуется еще $b'_j = b_j - a_i$ единиц груза. Поэтому, не принимая более во внимание i-ю строку, снова ищем клетку с наименьшей стоимостью перевозок и заполняем ее с учетом изменившихся потребностей.

В случае $a_i > b_j$ из рассмотрения исключается j-й столбец, а запасы A_i полагаются равными $a'_i = a_i - b_j$. Продолжаем этот процесс до тех пор, пока все запасы не будут исчерпаны, а все потребности удовлетворены.

Заметим, что при наличии в таблице клеток с одинаковыми тарифами, планы, полученные с помощью этого метода, могут быть разными, однако они, несомненно, ближе к оптимальному плану, чем план, составленный по методу северо-западного угла.

Таблица 3

заказы / запасы		B_1 100	B_2 40	B_3 80	B_4 60
A_1	160	4 **100**	8	10	5 **60**
A_2	30	4	6	2 **30**	3
A_3	90	4	4 **40**	6 **50**	5

Сформируем теперь первоначальный план по методу наименьшей стоимости для уже рассмотренного нами примера и сравним результаты (Таблица 3).

Поскольку наименьший тариф (число 2) стоит в клетке (2,3), то запишем в эту клетку элемент $x_{23} = 30$. Тогда $b'_3 = 50$, а 2-ю строку таблицы можно больше не учитывать.

Среди оставшихся клеток имеются три клетки с наименьшим тарифом перевозок, равным 4: (1,1); (3,1) и (3,2).

Выберем любую из них, например, клетку (1,1) и запишем в нее число $x_{ij} = 100$. Получаем, что $a' = 60$, а 1-й столбец таблицы больше не рассматриваем.

Теперь наименьший тариф, равный 4, проставлен в клетке (3,2), поэтому $x_{32} = 40$, $b'_3 = 50$ и 2-й столбец больше не нужен. Далее выбираем клетку (1,4) с тарифом 5 и пишем в нее $x_{14} = 60$. Исключив из рассмотрения сразу 1-ю строку и 4-ый столбец (поскольку $a'_1 = b_4 = 60$), переходим к последней клетке (3,3), в которую записываем перевозку $x_{33} = 50$.

Найдем суммарную стоимость перевозок по этому плану:

$z = 4 \times 100 + 5 \times 60 + 2 \times 30 + 4 \times 40 + 6 \times 50 = 1220$.

Сравнивая это значение со стоимостью плана, полученного по методу северо-западного угла, видим, что $1220 < 1460$, то есть мы получили более выгодный план перевозок.

2.4 Вырожденные планы. Циклы и пополнение плана

Прежде, чем перейти к анализу оптимальности планов и способам их улучшения, выясним, каким требованиям должны удовлетворять составляемые планы. Для этого вернемся к системе ограничений (4). В соответствии с определением плана перевозок у матрицы $X = \{x_{ij}\}$ сумма элементов i-й строки равняется a_i, $i = 1, 2,..., m$, а сумма элементов j-о столбца равняется b_j, $j = 1, 2,..., n$. Условие закрытости транспортной задачи $a = b$ означает, что среди $m + n$ уравнений системы (4) независимыми являются только $m + n - 1$ уравнений, поэтому в любом базисном решении этой системы должно быть $m + n - 1$ базисных переменных. Поскольку свободные переменные в таком решении равны нулю, то в транспортной таблице им будут соответствовать пустые клетки.

Клетки таблицы, в которые записаны отличные от нуля перевозки, называются *базисными,* а остальные (пустые) – *свободными.*

План называется *вырожденным,* если количество базисных клеток в нем меньше, чем $m + n - 1$.

Если на каком-то этапе решения получился вырожденный план, то его необходимо *пополнить,* проставив в недостающем числе клеток нулевую перевозку и превратив, тем самым, эти клетки в базисные. Общий баланс и суммарная стоимость перевозок плана при этом не изменятся. Однако проводить пополнение плана, выбирая клетки произвольно, нельзя.

Рассмотрим условия, которым должен соответствовать пополненный план.

Циклом в транспортной таблице называется несколько клеток, соединенных замкнутой ломаной линией так, чтобы две соседние вершины ломаной были расположены либо в одной строке, либо в одном столбце.

Ломаная линия может иметь точки самопересечения, но не в клетках цикла.

План называется *ациклическим,* если его базисные клетки не содержат циклов.

Доказано, что оптимальные планы являются ациклическими, поэтому и первоначальный план также должен удовлетворять этому требованию.

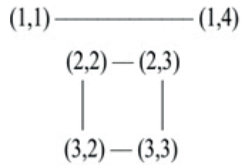
Заметим, что планы, полученные с помощью метода северо-западного угла и метода наименьшей стоимости, ациклические.

Однако если план оказался вырожденным, то при его пополнении требование ацикличности необходимо учитывать.

Возвращаясь к рассматриваемому примеру, видим, что первоначальный план, полученный по методу наименьшей стоимости, имеет 5 базисных клеток, однако

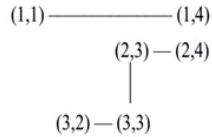
$m + n - 1 = 3 + 4 - 1 = 6$.

Значит, план нужно пополнить еще одной базисной клеткой. Попробуем выбрать для этого клетку (2,2). Соединив базисные клетки горизонтальными и вертикальными отрезками, получаем:



Видим, что пополненный таким образом план содержит цикл из клеток (2,2); (2,3); (3,3) и (3,2), следовательно, клетка (2,2) была выбрана неудачно.

Взяв вместо нее клетку (2,4), получим ациклический план.



Поэтому можно заполнить эту клетку, положив $x_{24} = 0$.

2.5 Проверка оптимальности плана и перераспределение поставок с помощью метода потенциалов

Для анализа полученных планов и их последующего улучшения удобно ввести дополнительные характеристики пунктов отправления и назначения, называемые *потенциалами.*

2.5.1 Вычисление потенциалов

Сопоставим каждому поставщику A_i и каждому потребителю B_j величины u_i и v_j, соответственно так, чтобы для всех базисных клеток плана были выполнены соотношения:

$$u_i + v_j = C_{ij},$$
$$i = 1, 2,..., m, \quad j = 1, 2,..., n. \qquad (5)$$

Поскольку число базисных клеток в плане равно $m + n - 1$ (вырожденные планы должны быть предварительно пополнены), то для определения потенциалов получается система из $m + n - 1$ уравнений с $m + n$ неизвестными.

Такая система имеет бесконечное множество решений. Нам требуется найти любое ее решение. Обычно для простоты полагают один из потенциалов равным нулю и затем вычисляют остальные.

В транспортной таблице для потенциалов $v_1,v_2,...,v_n$ заводится дополнительные строка, а для потенциалов u_1, u_2, … u_i – дополнительный столбец, куда проставляются найденные значения.

Проверка оптимальности плана

Для каждой свободной клетки плана вычислим разности:

$$\Delta C_{ij} = C_{ij} - (u_i + v_j) \qquad (6)$$

Запишем полученные значения в левых нижних углах соответствующих клеток. Заметим, что для базисных клеток выполнено соотношение $C_{ij} = 0$, и этим фактом можно пользоваться для контроля правильности нахождения потенциалов.

План является оптимальным, если все разности $C_{ij} > 0$.

В противном случае план можно улучшить «*перераспределением поставок»*

2.5.2 Перераспределение поставок

Найдем клетку с наибольшей по абсолютной величине отрицательной разностью ΔC_{ij} и построим цикл, в котором кроме этой клетки все остальные являются базисными. Такой цикл всегда существует и является *единственным*.

Заметим, что в новом плане суммы элементов по строкам и столбцам должны остаться прежними, поэтому изменение значения в одной клетке цикла повлечет за собой соответствующие изменения значений во всех остальных клетках этого цикла.

Так как в свободной клетке значение будет увеличено, то проставим в ее правом нижнем углу знак «+». Теперь пройдем по всей ломаной цикла, проставляя в правых нижних углах клеток поочередно знаки «плюс в кружке» и «минус в кружке».

Груз будет перераспределен по клеткам цикла на величину $\Delta x = \min x_{ij}$ следующим образом. В клетках со знаком «плюс» значение перевозки нужно

увеличить на величину Δx, а в клетках со знаком «минус» – уменьшить на величину Δx. Так как после пересчета у нас добавилась лишняя базисная клетка, то их количество необходимо сократить, убрав нуль в одной из клеток цикла. Если таких клеток получилось несколько, то свободной делаем ту из них, в которой тариф перевозок *максимален*.

После этого полученный план проверяется на оптимальность рассмотренным ранее способом ($C_{ij} > 0$). Перераспределение груза производим до тех пор, пока очередной план не станет оптимальным. На этом действие алгоритма завершается.

Рассмотрим, как нужно пользоваться методом потенциалов, на примере первоначального плана, полученного ранее по методу северо-западного угла.

Вначале проверим, не является ли этот план вырожденным.

Так как $m + n - 1 = 3 + 4 - 1 = 6$, и число базисных клеток в плане также равно 6, то план в пополнении не нуждается. Найдем потенциалы по базисным клеткам таблицы с помощью формул (7), положив $u_1 = 0$ и занесем полученные значения в таблицу.

$$\begin{cases} u_1 + v_1 = 4, \\ u_1 + v_2 = 8, \\ u_1 + v_3 = 10, \\ u_2 + v_3 = 2, \\ u_3 + v_3 = 6, \\ u_3 + v_4 = 5, \\ u_1 = 0, \end{cases} \Leftrightarrow \begin{cases} u_1 = 0, \\ u_2 = -8, \\ u_3 = -4, \\ v_1 = 4, \\ v_2 = 8, \\ v_3 = 10, \\ v_4 = 9, \end{cases} \qquad (7)$$

Вычислим теперь разности Δc_{ij} для свободных клеток и также проставим эти данные в левых нижних углах соответствующих клеток. В итоге получим следующую таблицу 4.

Таблица 4

запасы \ заказы		B_1	B_2	B_3	B_4	
		100	40	80	60	u
A_1	160	4: 100	8: 40	10: 20 (⊖)	5: – 4 (⊕)	0
A_2	30	4: 8	6: 6	2: 30	3: 2	– 8
A_3	90	4: 4	4: 0	6: 30 (⊕)	5: 60	– 4
	v	4	8	10	9	

Поскольку $\Delta c_{14} = -4 < 0$, то этот план не является оптимальным.

Перераспределим груз по циклу, обозначенному в таблице 4 пунктиром, на величину $\Delta x = \min(20,60) = 20$.

Для этого в клетках со знаком «плюс» увеличим поставки на 20 единиц, а клетках со знаком «минус» – поставки на столько же уменьшим. Для сохранения количества базисных клеток число 0 в клетке (1,3) не записываем, и она становится свободной.

Вычислив потенциалы и разности Δc_{ij} для нового плана, видим, что снова есть отрицательная разность $\Delta c_{32} = -4$.

Поэтому придется план улучшать еще раз. С этой целью перераспределим груз по циклу, отмеченному в таблице 5 пунктиром, на величину $\Delta x = \min(40,40) = 40$. Так как в результате в цикле получаются две клетки с нулевыми перевозками: (1,3) и (3,4), то сделаем свободной клетку (1,3), поскольку ее тариф перевозок больше. После перераспределения груза по циклу вычислим все необходимые разности Δc_{ij}.

Т а б л и ц а 5

запасы \ заказы		B_1	B_2	B_3	B_4	
		100	40	80	60	u
A_1	160	4: 100	8: 40 ⊖	10: 4	5: 20 ⊕	0
A_2	30	4: 4	6: 2	2: 30	3: 2	−4
A_3	90	4: 0	4: −4 ⊕	6: 50 ⊕	5: 40 ⊖	0
	v	4	8	6	5	

Как видим, все Δc_{ij} неотрицательны, значит, план оптимален (таблица 6). Найдем суммарную стоимость перевозок по этому плану:

$Z = 4 \times 100 + 5 \times 60 + 2 \times 30 + 4 \times 40 + 6 \times 50 = 1220$.

Т а б л и ц а 6

запасы \ заказы		B_1	B_2	B_3	B_4	
		100	40	80	60	u
A_1	160	4: 100	8: 4	10: 4	5: 60	0
A_2	30	4: 4	6: 6	2: 30	3: 2	−4
A_3	90	4: 0	4: 40	6: 50	5: 0	0
	v	4	4	6	5	

3. ОПРЕДЕЛЕНИЕ МАКСИМАЛЬНОГО ПОТОКА В СЕТИ

Задана сеть из P узлов, связанных направленными соединениями, среди которых выделены источник S и сток T (S ≠ T).

Необходимо определить максимально возможное количество данных, которое можно передать по этой сети из S в T. Также требуется выяснить, как можно устроить передачу этого максимально возможного количества данных.

Эту задачу можно рассматривать для сетей компьютеров, сетей аэропортов, железнодорожных, автомобильных дорог, сетей трубопроводов и т. д.

Пусть G = (V, E) – орграф.

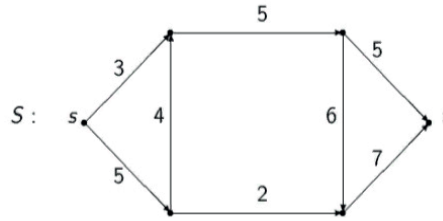
Для каждой вершины v ∈ V определим множества A(v) и B(v):

т. е. A(v) – множество всех дуг, исходящих из вершины v в графе G;

а B(v) – множество всех дуг, входящих в вершину v в графе G.

Если A (v) = 0, т. е. в вершину v дуги не входят, то вершина v называется **источником**. Если B (v) = 0, т. е. из вершины v дуги не исходят, то вершина v называется **стоком**.

Сетью S = (G; c) назовем орграф G с заданной функцией c: E —> R+, сопоставляющей каждой дуге e ∈ E неотрицательное действительное число c (e), называющееся пропускной способностью этой дуги e. Сеть S = (G; c):



Потоком в сети S = (G; c), где G = (V, E), называется такая функция f : E —> R_+, приписывающая каждой дуге e ∈ E неотрицательное число f (e), называемое потоком по этой дуге e, для которой выполняются свойства:

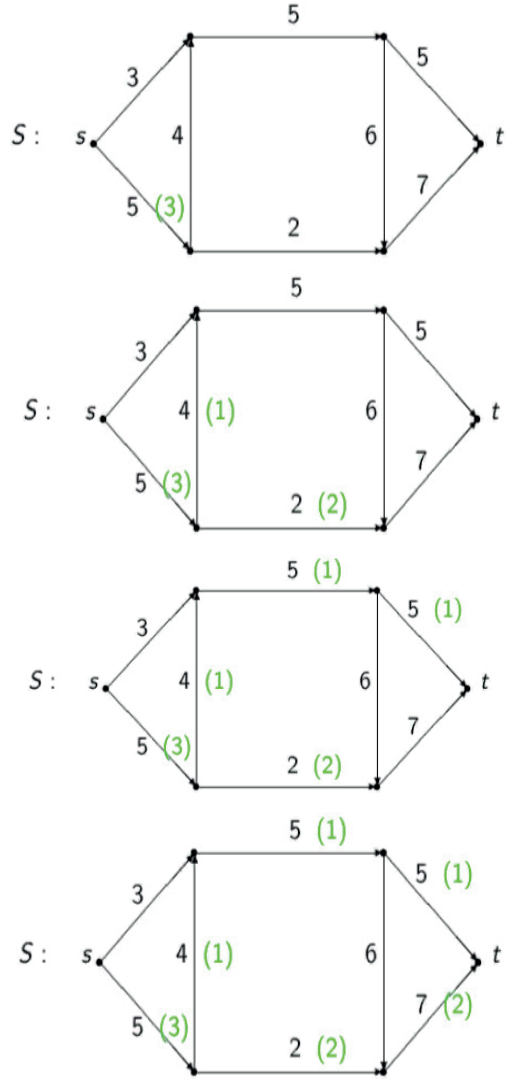
- для каждой дуги e ∈ E верно f (e) ≤ c (e), т. е. поток по любой дуге не превышает пропускную способность этой дуги;
- для каждой вершины v ∈ V, v ≠ s, t, верно, что поток, входящий в любую вершину, кроме источника и стока,
- без остатка выходит из нее (закон *сохранение потока*).

$$D_f(v) = \sum_{e \in A(v)} f(e) - \sum_{e \in B(v)} f(e) = 0,$$

При этом неотрицательное число $p_f = \sum_{e \in A(s)} f(e)$

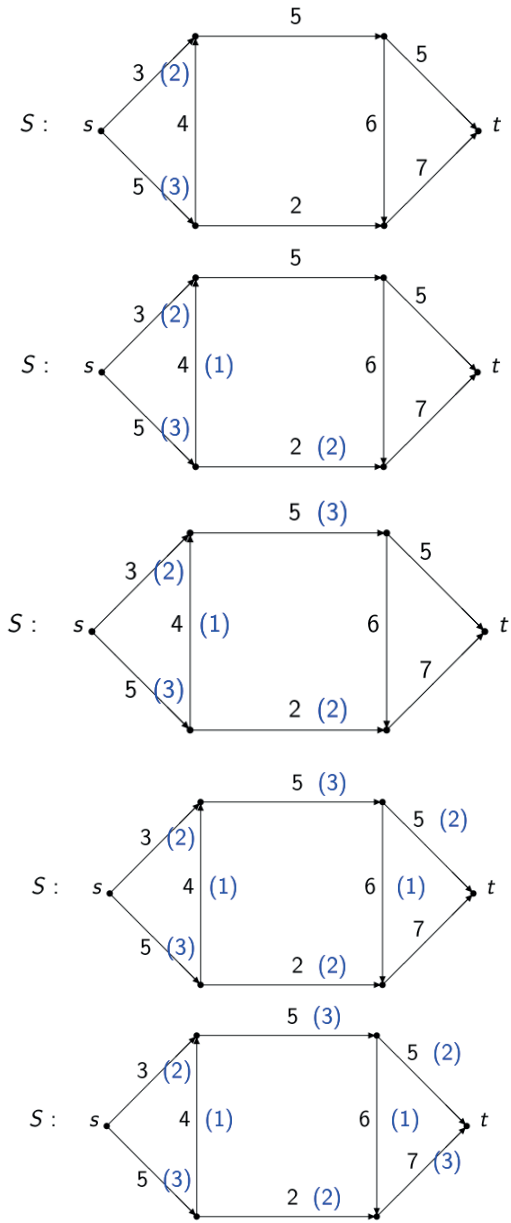
называется величиной потока f.

Найдем поток f1 в сети S = (G; c):



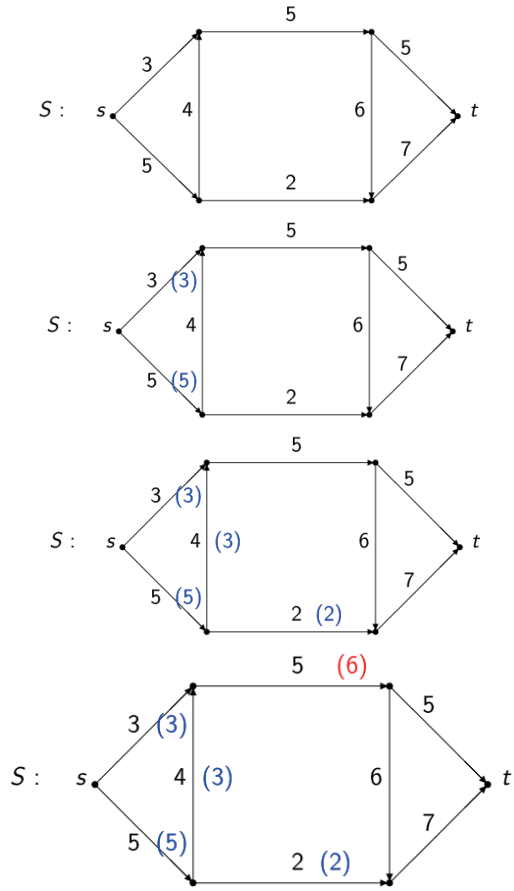
Получаем: $Pf_1 = 3$

Можно построить другой поток f_2 в этой же сети S = (G; c):



Получаем: $Pf_2 = 5$

Попытаемся построить поток f_3 в этой же сети S = (G; c):



Поток f_3 с $Pf_3 = 8$ построить не удалось, т.е. максимальный поток для данной сети $Pf_2 = 5$.

Часто для решения задачи о максимальном потоке в транспортной используют понятие разреза сети.

Разрезом сети называется множество, которому принадлежит исток, и не принадлежит сток. Т.е. разрез – это минимальное (в смысле отношения включения) множество дуг, удаление которых «разрывает» все пути, соединяющие исток и сток. Если удалить из сети S все дуги разреза R, то останется сеть, в которой нет ни одного направленного пути из s в t.

Пропускной способностью разреза R (X, Y) называется неотрицательное число:

$$c(R) = \sum_{e \in R} c(e),$$

т. е. число равное сумме пропускных способностей его дуг.

4 ОПРЕДЕЛЕНИЕ КРАТЧАЙШИХ МАРШРУТОВ В ТРАНСПОРТНОЙ СЕТИ

4.1 Алгоритм Дейкстры

Алгоритм Дейкстры – алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных.

Алгоритм работает только для графов без рёбер отрицательного веса.

Формулировка задачи

Имеется некоторое количество авиарейсов между городами мира, для каждого известна стоимость. Стоимость перелёта из аэропорта A в аэропорт B может быть не равна стоимости обратного перелёта из B в A. Найти маршрут минимальной стоимости (возможно, с пересадками) от Копенгагена до Барнаула.

Формальное определение

Дан взвешенный ориентированный граф G (V, E) без дуг отрицательного веса.

Надо найти кратчайшие пути от некоторой вершины a графа G до всех остальных вершин этого графа.

Каждой вершине из V сопоставим *метку* – минимальное известное расстояние от этой вершины до a.

Алгоритм работает пошагово – на каждом шаге он «посещает» одну вершину и пытается уменьшить ее метку.

Работа алгоритма завершается, когда все вершины посещены.

Подготовка к работе.

Метка самой вершины a полагается равной 0, метки остальных вершин – бесконечности. Это означает то, что расстояния от a до других вершин пока неизвестны.

Изначально все вершины графа помечаются как не посещённые.

Шаг алгоритма.

Если все вершины посещены, алгоритм завершается.

В противном случае, из ещё не посещённых вершин выбирается вершина x_i, имеющая минимальную метку.

Мы рассматриваем все возможные маршруты, в которых x_i является предпоследним пунктом. Вершины, в которые ведут ребра из x_i, называются соседями этой вершины.

Для каждого соседа вершины x_i, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки $L(x_i)$ и длины ребра $d(x_s; x_i)$, соединяющего x_i с этим соседом.

Если полученное значение длины меньше значения метки соседа, заменяем значение метки полученным значением длины.

Обновление пометок вершин графа производится по следующему выражению:

$$L'(x_i) = min\begin{bmatrix} L(x_i) \\ L(x_s) + d(x_s; x_i) \end{bmatrix}$$

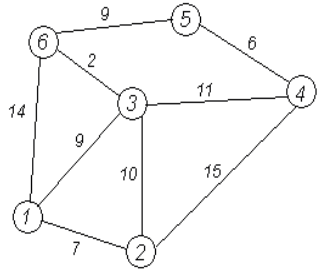
где $L'(x_i)$ – обновленная пометка;
$L(x_i)$ – вершина, для которой обновляется пометка;
$L(x_s)$ – «стартовая» вершина;
$d(x_s; x_i)$ – расстояние от «стартовой» вершины до обновляемой.

Рассмотрев всех соседей, помечаем вершину xi как посещённую и повторим шаг алгоритма.

4.1 Решение задачи

Рассмотрим выполнение алгоритма на примере графа, представленного на рисунке.

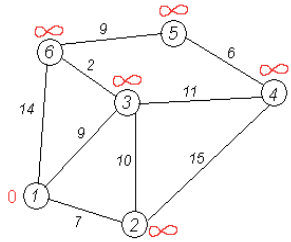
Пусть требуется найти кратчайшие расстояния от 1-й вершины до всех остальных.



Кружками обозначены вершины, линиями – пути между ними (ребра графа).

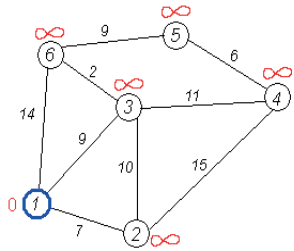
В кружках обозначены номера вершин, над ребрами обозначен их вес – длина пути.

Рядом с каждой вершиной красным обозначена метка – длина кратчайшего пути в эту вершину из вершины 1.



Первый шаг.

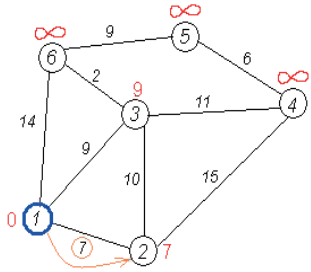
Минимальную метку имеет вершина 1. Её соседями являются вершины 2, 3 и 6.



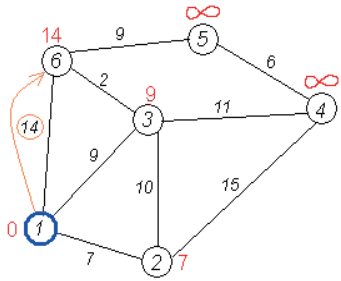
Первый по очереди сосед вершины 1 – вершина 2, потому что длина пути до неё минимальна.

Длина пути в неё через вершину 1 равна сумме значения метки вершины 1 и длины ребра, идущего из 1-й в 2-ю, то есть 0 + 7 = 7.

Это меньше текущей метки вершины 2, бесконечности, поэтому новая метка 2-й вершины равна 7.



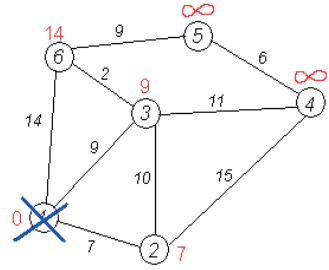
Аналогичную операцию проделываем с двумя другими соседями 1-й вершины – 3-й и 6-й.



Все соседи вершины 1 проверены.

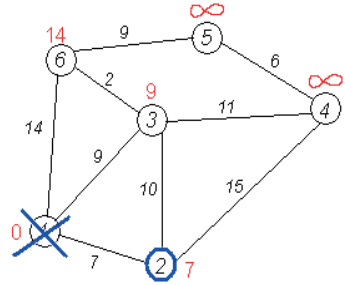
Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру не подлежит.

Вычеркнем её из графа, чтобы отметить, что эта вершина посещена.



Второй шаг.

Снова пытаемся уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2-ю вершину. Соседями вершины 2 являются вершины 1, 3 и 4.



Первый (по порядку) сосед вершины 2 — вершина 1.

Но она уже посещена, поэтому с 1-й вершиной ничего не делаем.

Следующий сосед – вершина 3, так как имеет минимальную метку.

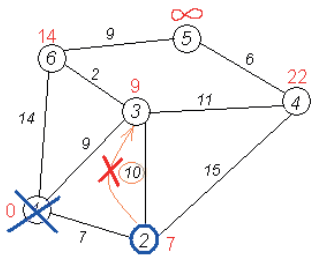
Если идти в неё через 2, то длина такого пути будет равна 17 (7 + 10 = 17).

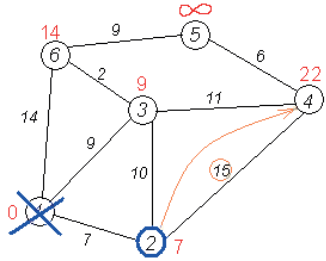
Но текущая метка третьей вершины равна 9, а это меньше 17, поэтому метка **не меняется.**

Ещё один сосед вершины 2 – вершина 4.

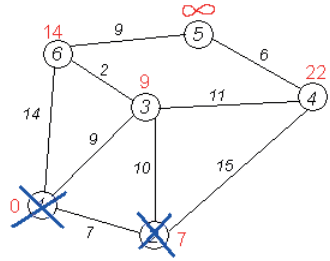
Если идти в неё через 2-ю, то длина такого пути будет равна сумме кратчайшего расстояния до 2-й вершины и расстояния между вершинами 2 и 4, то есть 22 (7 + 15 = 22).

Поскольку 22 < бесконечности, то устанавливаем метку вершины 4 равной 22.





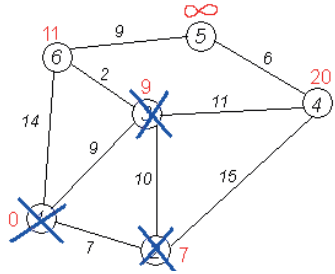
Все соседи вершины 2 просмотрены, замораживаем расстояние до неё и помечаем её как посещённую.



Третий шаг.

Повторяем шаг алгоритма, выбрав вершину 3.

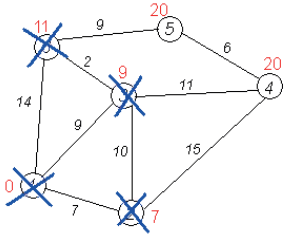
После её «обработки» получим такие результаты:

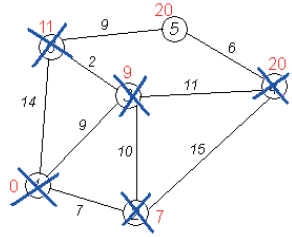


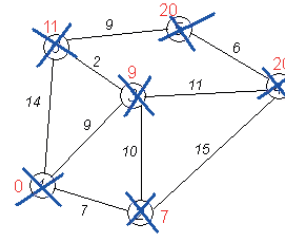
Дальнейшие шаги.

Повторяем шаг алгоритма для оставшихся вершин.

Это будут вершины 6, 4 и 5, соответственно порядку.







Завершение выполнения алгоритма.

Алгоритм заканчивает работу, когда все вершины посещены.

Результат работы алгоритма виден на последнем рисунке: кратчайший путь от вершины 1 до 2-й составляет 7, до 3-й – 9, до 4-й – 20, до 5-й – 20, до 6-й – 11.

Если в какой-то момент все не посещённые вершины помечены бесконечностью, то это значит, что до этих вершин нельзя добраться (то есть граф несвязный). Тогда алгоритм может быть завершён досрочно.

Также всегда существует вектор P, исходя из которого можно построить кратчайшие маршруты.

По количеству элементов этот вектор равен количеству вершин в графе, Каждый элемент содержит последнюю промежуточную вершину на кратчайшем пути между вершиной-источником и конечной вершиной.

В начале алгоритма все элементы вектора P равны вершине источнику (в нашем случае P = {1, 1, 1, 1, 1, 1}).

Далее на этапе пересчета значения метки для рассматриваемой вершины, в случае если метка рассматриваемой вершины меняется на меньшую, в массив P мы записываем значение текущей вершины W.

Например: у 6-ой вершины была метка со значением «14», при W = 1.

Далее при W = 3, метка 6-ой вершины изменилась на «11», следовательно мы запишем значение в вектор P — P [3] = 11.

В результате получим вектор P = {7, 9, 11, 20, 20}.

Зная, что в каждом элементе вектора P записана последняя промежуточная вершина на пути между источником и конечной вершиной, мы можем получить и сам кратчайший маршрут.

5 АЛГОРИТМ КРАСКАЛА, ПРИМА ДЛЯ НАХОЖДЕНИЯ МИНИМАЛЬНОГО ОСТОВНОГО ДЕРЕВА

Задача о нахождении минимального остовного дерева часто встречается в следующей постановке:

есть N городов, которые необходимо соединить дорогами так, чтобы можно было добраться из любого города в любой другой (напрямую или через другие города). Разрешается строить дороги между заданными парами городов, и известна стоимость строительства каждой такой дороги.

Требуется решить, какие именно дороги нужно строить, чтобы минимизировать общую стоимость строительства.

Эта задача может быть сформулирована в терминах теории графов как задача о нахождении минимального остовного дерева в графе, вершины которого представляют города, рёбра – это пары городов, между которыми можно проложить прямую дорогу, а вес ребра равен стоимости строительства соответствующей дороги.

Формальная постановка задачи

Имеется неориентированный взвешенный граф.

Назовем остовным деревом подграф, содержащий все вершины исходного графа, который является деревом.

Задача состоит в том, чтобы найти такое остовное дерево, сумма рёбер которого минимальна.

Формальная постановка задачи

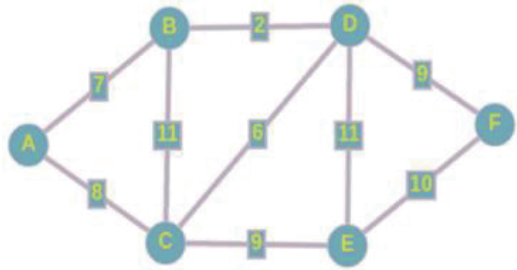
Имеется неориентированный взвешенный граф.

Назовем остовным деревом подграф, содержащий все вершины исходного графа, который является деревом.

Задача состоит в том, чтобы найти такое остовное дерево, сумма рёбер которого минимальна.

Неформальная постановка задачи

Представьте исходный граф без рёбер, теперь нам нужно как-то соединить все вершины между собой, чтобы можно было бы попасть из любой вершины в другую, не имея при этом циклов в получившемся графе с минимально возможной суммой весов включенных рёбер.



Механизм, по которому работает данный алгоритм, очень прост. На входе имеется пустой подграф, который и будем достраивать до потенциального минимального остовного дерева.

Будем рассматривать только связные графы, в другом случае при применении алгоритма Краскала мы будем получать не минимальное остовное дерево, а просто остовной лес.

5.1 Алгоритм Краскала

Вначале мы производим сортировку рёбер **по не убыванию** по их весам.

Добавляем i-ое ребро в наш подграф только в том случае, если данное ребро соединяет две разные компоненты связности, одним из которых является наш подграф. То есть, на каждом шаге добавляется минимальное по весу ребро, один конец которого содержится в нашем под графе, а другой – еще нет.

Алгоритм завершит свою работу после того, как множество вершин нашего подграфа совпадет с множеством вершин исходного графа.

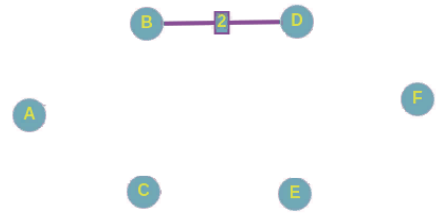
Данный алгоритм называется «жадным» из-за того, что мы на каждом шаге пытаемся найти оптимальный вариант, который приведет к оптимальному решению в целом.

Разберем конкретный пример по шагам.

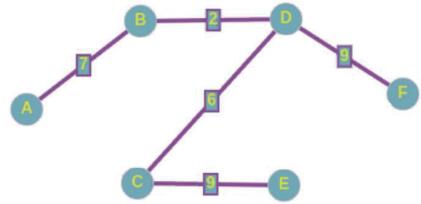
Из представленного ранее графа, выпишем все его ребра в отсортированном порядке:

1) D <--> B; w = 2
2) D <--> C; w = 6
3) A <--> B; w = 7
4) A <--> C; w = 8
5) C <--> E; w = 9
6) D <--> F; w = 9
7) F <--> E; w = 10
8) B <--> C; w = 11
9) D <--> E; w = 11

И начнем по списку добавлять эти ребра в наш остов:

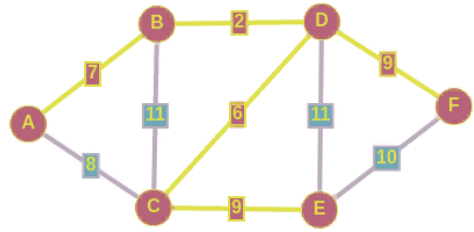


Подграф после добавления 1-го ребра



При добавлении в наше остовное дерево ребра A <--> C образовывается цикл, поэтому мы просто пропускаем данное ребро.

В итоге мы соединили все вершины ребрами с минимально-возможными весами, а значит, нашли минимальное остовное дерево для нашего исходного графа.



Повторим шаги алгоритма Краскала:

1. Удаляем все ребра, сортируем их по возрастанию;
2. Добавляем дугу с минимальным весом;
3. Если возник цикл в графе, берем другую дугу;
4. Повторяем шаг 2;
5. Если число компонент связности дойдет до 1, цикл завершается досрочно.

Замечание

Остовное дерево всегда имеет дуг на одну меньше, чем количество вершин.

Поэтому шаг 2 будет повторяться n-2 раз, где n – количество вершин графа.

5.2 Алгоритм Прима

Суть самого алгоритма Прима тоже сводится к «жадному» перебору рёбер, но уже из определенного множества.

На входе так же имеется пустой подграф, который и будем достраивать до потенциального минимального остовного дерева.

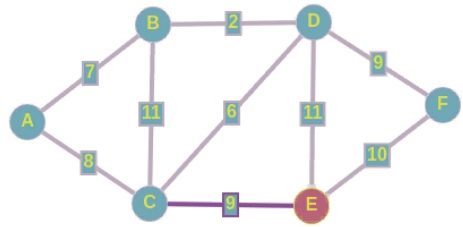
Изначально наш подграф состоит из одной любой вершины исходного графа.

Затем из ребер инцидентных этой вершине, выбирается такое минимальное ребро, которое связало бы две абсолютно разные компоненты связности, одной из которых и является наш подграф. То есть, как только у нас появляется возможность добавить новую вершину в наш подграф, мы тут же включаем ее по минимально возможному весу.

Продолжаем выполнять предыдущий шаг до тех пор, пока не найдем искомое минимальное остовное дерево (MST).

Разберем конкретный пример по шагам.

Выбираем чисто случайно вершину E, далее рассмотрим все ребра, исходящие из нее, включаем в наше остовное дерево ребро C <--> E; W = 9, так как данное ребро имеет минимальный вес из всех рёбер инцидентных множеству вершин нашего подграфа. Имеем следующее:



Подграф после добавления 1-го ребра

Теперь выборка производится из рёбер:

D <--> C; w = 6

A <--> C; w = 8

F <--> E; w = 10

B <--> C; w = 11

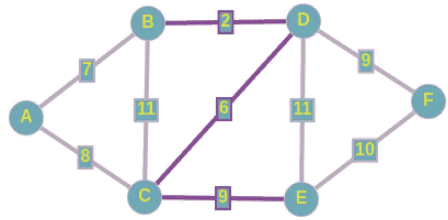
D <--> E; w = 11

То есть, в данный момент, мы знаем только о двух вершинах, соответственно, знаем обо всех ребрах, исходящих из них.

Про связи между другими вершинами, которые не включены в наш подграф, мы ничего не знаем, поэтому они на этом шаге не рассматриваются.

Добавляем в наш подграф ребро D <--> C и по аналогии добавляем ребро D <--> B.

Получаем следующее:



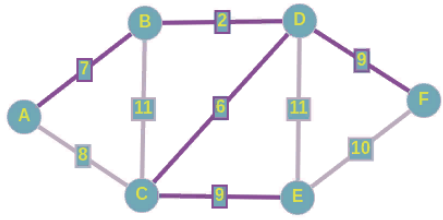
Подграф, полученный после добавления рассмотренных рёбер

Давайте добьем наш подграф до минимального остовного дерева.

Вы, наверное, уже догадались о том, по каким ребрам мы будем связывать наши оставшиеся вершины: A и F.

Проводим последние штрихи и получили тот же самый подграф в качестве минимального остовного дерева.

Но как мы раннее говорили, сам подграф ничего не решает, главное тут – множество рёбер, которые включены в наше остовное дерево.



Искомое минимальное остовное дерево

Суммарный вес искомого MST равен 33.

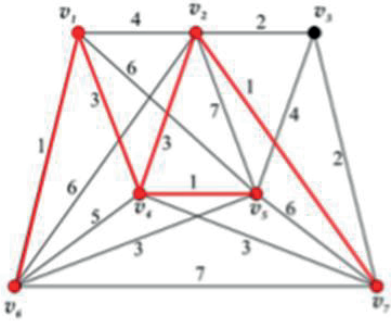
Повторим шаги алгоритма Прима:

1. Удаляем все ребра;
2. Выбираем вершину;
3. Добавляем инцидентную дугу с минимальным весом для данной вершины и соединяем новую;
4. Если возник цикл в графе, дуга игнорируется;
5. Если число компонент связности дойдет до 1, цикл завершается досрочно.

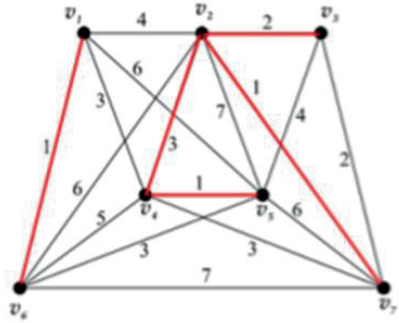
Сравнение алгоритмов Прима и Краскала

	Алгоритм Прима		**Алгоритм Краскала**
1	Удаляем все ребра.	1	Удаляем все ребра, сортируем по возрастанию.
2	Выбираем вершину.	2	Добавляем дугу с минимальным весом.
3	Добавляем инцидентную дугу с минимальным весом для данной вершины и соединяем новую.	3	Если возник цикл в графе, берем другую дугу.
4	Если возник цикл в графе, дуга игнорируется.	4	Повторяем шаг 2.
5	Если число компонент связности дойдет до 1, цикл завершается досрочно.	5	Если число компонент связности дойдет до 1, цикл завершается досрочно.

Алгоритм Прима



Алгоритм Краскала



6 МЕТОД ВЕТВЕЙ И ГРАНИЦ

Впервые «метод ветвей и границ» был предложен применительно к задаче целочисленного линейного программирования в 1960 г. Он применим как к полностью, так и частично целочисленным задачам.

В основе «метода ветвей и границ» лежит идея последовательного разбиения множества допустимых решений на подмножества (стратегия «разделяй и властвуй»).

На каждом шаге метода элементы разбиения подвергаются проверке для выяснения, содержит данное подмножество оптимальное решение или нет.

Если рассматривается задача на минимум, то проверка осуществляется путем сравнения нижней оценки значения целевой функции на данном подмножестве с верхней оценкой функционала. В качестве оценки сверху используется значение целевой функции на некотором допустимом решении.

Допустимое решение, дающее наименьшую верхнюю оценку, называют *рекордом*.

Если оценка снизу целевой функции на данном подмножестве не меньше оценки сверху, то рассматриваемое подмножество не содержит решения лучше рекорда и может быть отброшено.

Если значение целевой функции на очередном решении меньше рекордного, то происходит смена рекорда.

Будем говорить, что подмножество решений просмотрено, если установлено, что оно *не содержит решения лучше рекорда.*

Если просмотрены все элементы разбиения, алгоритм завершает работу, а текущий рекорд является *оптимальным решением*. В противном случае среди не просмотренных элементов разбиения выбирается множество, являющееся, в определенном смысле, перспективным. Оно подвергается разбиению (ветвлению).

Новые подмножества анализируются по описанной выше схеме. Процесс продолжается до тех пор, пока не будут просмотрены все элементы разбиения.

Метод используется для решения некоторых NP-полных задач, таких как: «*задача коммивояжёра»* или «*задача о ранце».*

Одна из самых известных и важных задач транспортной логистики – задача коммивояжера или «задача о странствующем торговце». Также в литературе встречается название «задача китайского почтальона».

Суть задачи сводится к поиску оптимального (кратчайшего, быстрейшего или самого дешевого) пути, проходящего через промежуточный пункты по одному разу и возвращающегося в исходную точку.

Например, нахождение наиболее выгодного маршрута, позволяющего коммивояжеру посетить со своим товаром определенные города по одному разу и вернуться обратно.

Мерой выгодности маршрута может быть минимальное время поездки, минимальные расходы на дорогу или минимальная длина пути.

В наше время, когда стоимость доставки часто бывает сопоставима со стоимостью самого товара, а скорость доставки – один из главных приоритетов, задача нахождения оптимального маршрута приобретает огромное значение.

Ее цель «задачи коммивояжера» заключается в нахождении самого выгодного маршрута, проходящего через все заданные точки (пункты, города) по одному разу, с последующим возвратом в исходную точку.

Особенности задачи в том, что она довольно просто формулируется и найти хорошие решения для нее также относительно просто, но вместе с тем поиск действительно оптимального маршрута для большого набора – непростой и ресурсоемкий процесс.

6.1 Общая характеристика задачи «задачи коммивояжера»

Виды данной задачи по симметричности ребер графа могут быть 2-х типов:

– симметричная – все пары ребер, соединяющих одни и те же вершины, имеют одинаковую длину (т.е. граф, представляющий исходные данные задачи, является неориентированным).

Иными словами, длина прямого пути от города А до города В и длина обратного пути от города В до города А, одинаковы. То же самое справедливо и в отношении остальных пар городов (А-С и С-А, В-С и С-D, и т. д.);

– асимметричная – длина пар ребер, соединяющих одни и те же города, может различаться (ориентированный граф). Для этого типа задачи коммивояжера прямой путь, например, из города А в город В может быть короче или длиннее обратного пути из города В в город А.

Типы задачи коммивояжера так же подразделяются по замкнутости маршрута:

– замкнутая – нахождение кратчайшего пути, проходящего через все вершины по одному разу с последующим возвратом в точку старта (маршрут получается замкнутым, закольцованным);
– незамкнутая – нахождение кратчайшего пути, проходящего через все вершины по одному разу и без обязательного возврата в исходную точку (маршрут получается разомкнутым).

Методы решения задачи коммивояжера довольно разнообразны и различаются применяемым инструментарием, точностью находимого решения и сложностью требуемых вычислений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Селезнева, С. Н. Математика. Лекция 8. Сеть. Поток в сети. Теорема о величине максимального потока в сети. Нахождение максимального потока в сети. [Электронный ресурс] / С.Н. Селезнева. Режим доступа: http://mk.cs.msu.ru, свободный. – Загл. с экрана.

2. Давыдов, Е.Г. Элементы исследования операций / Е.Г. Давыдов. Москва: Крокус, 2013.

3. Бирюков, Р.С. Методы оптимизации в примерах и задачах. Учебно-методическое пособие / Р.С. Бирюков и др. Нижний Новгород: НГУ, 2010.

4. Шапорев, С.Д. Дискретная математика. Курс лекций и практических занятий / С.Д. Шапорев. Санкт Петербург: БХВ-Петербург, 2006.

5. Партыка, Т.Л. Математические методы. Учебник / Т.Л. Партыка, И.И. Попов. Москва: Форум – ИНФРА, 2007.

6. Грешилов, А.А. Математические методы принятия решений. Учебное пособие для вузов / А.А. Грешилов. Москва: Изд-во МГТУ им. Н. Э. Баумана, 2006.

7.Венцель, Е.С. Исследование операций / Е.С. Венцель. Москва: Высшая школа, 2004.