ТЕОРИЯ СИСТЕМ И СИСТЕМНЫЙ АНАЛИЗ. ТЕОРИЯ А2. ОПТИМИЗАЦИЯ НА СЕТЯХ

1. Пояснение к теме. Задачи конечномерной оптимизации. Их сетевые постановки

В технических и, прежде всего, в экономических системах часто возникает проблема выбора одного из очень многих вариантов решения некоторой производственной проблемы. Примером, такой проблемы может быть задача **эффективной маршрутизации**: как проехать на автомобиле из указанного начального пункта по сети дорог до другого заданного пункта на этой сети. При этом желательно выбрать не первый попавшийся маршрут, а достаточно выгодный для экономии затрат на топливо. Ставя эту практическую задачу математически, можно искать самый дешевый маршрут. Если все дороги по качеству одинаковы, то самый дешевый маршрут является самым коротким. Такая, уже строго поставленная задача, называется **задачей оптимальной маршрутизации**. Казалось бы, что эта задача может быть решена простым перебором всех возможных маршрутов, поскольку этих маршрутов конечное число. В пользу такого совсем простого подхода говорит возможность применения ЭВМ, которая в принципе способна перебрать огромное число вариантов за разумное время. Рассмотрим и проанализируем пример подсчета числа вариантов в одной из задач маршрутизации.

Пример 1. Рассматриваемая сеть дорог представляет собой семейство из (n+1)-ой параллельной прямой, пересеченное другим семейством из (n+1)-ой параллельной прямой. Каждый отрезок прямой между двумя точками пересечения требует для проезда известную сумму денег. Требуется узнать, сколько существует вариантов для проезда из юго-западного угла A в северо-восточный угол B сети, изображенной на рис. 1. При этом ограничим себя теми маршрутами, в которых из каждой точки можно двигаться либо вверх, либо направо. Такое ограничение в ряде случаев разумно. Если брать маршрут между пунктами A и B не из этого семейства, то всегда найдется путь из указанного семейства с меньшим числом звеньев.

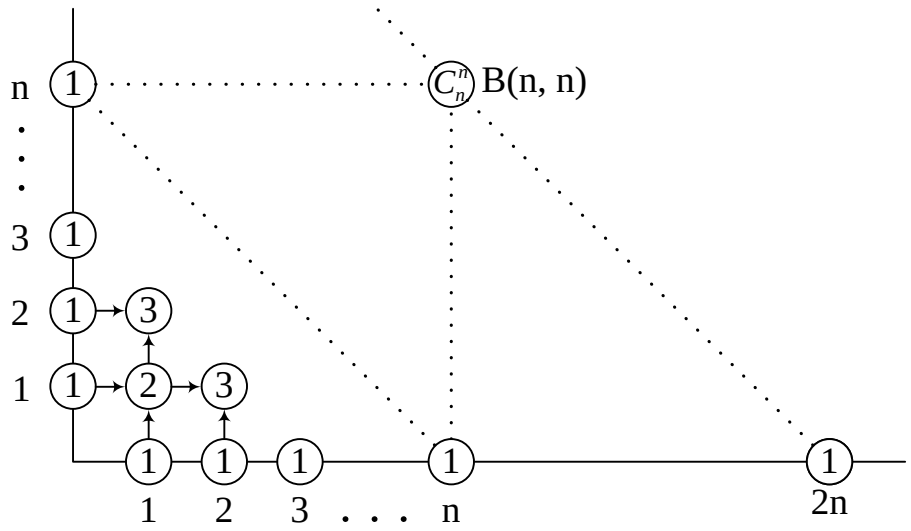


Рис.1. Схема маршрутов к примеру 1

Решение

Из точки A(0,0) в точку с координатой (0,1) можно попасть единственным способом за один шаг. Также единственным способом можно попасть из точки A(0,0) в точки с координатами $(0, m)$и $(k, 0)$, соответственно, за m и за k шагов. В точку с координатами (1,1) можно попасть двумя способами – из точки (0,1) и из точки (1,0). Таким образом, заполнена первая диагональная цепь точек, состоящая из точек (0,1), (1,1) и (1,0). Этим точкам соответствуют следующие числа способов попадания: 1, 2, 1. Рассмотрим следующую цепь с координатами вершин (0,2), (1,2), (2,1) и (2,0). Двум крайним точкам этой цепи уже найдены числа способов попадания – по одному. В точку (1,2) можно попасть из точки (1,1) и из точки (0,2) – всего тремя способами, потому что в точку (1,1) можно было попасть двумя способами, а в (0,2) – одним. Рассуждая аналогично, получим, что в точку (2,1) можно попасть также тремя способами. В выполненных расчетах прослеживается конструкция треугольника Паскаля (см. раздел Комбинаторики Теория А из блока тем Дискретная математика):

1-2-1, 1-3-3-1. Несложно доказать с помощью метода математической индукции и свойств сочетаний, что в точку с координатами (n, m) можно попасть C_{n+m}^m способами. В частности, число путей из пункта А в пункт В равно

$$C_{2n}^n = \frac{(2n)!}{(n)!\,(n)!} = \frac{(n+1) * (n+2) * \ldots * (2n)}{1 * 2 * \ldots * n} > 2 * 2 * \ldots * 2 = 2^n.$$

Пусть $n = 100$. В таком случае число путей больше числа $2^{100} > 10^{30}$.

Оказалось, что число вариантов чрезвычайно велико. Непосредственно перебрать такое число вариантов совершенно невозможно даже с помощью ЭВМ. Этот пример характерен для современных проблем экономики и техники. Сталкиваясь с подобными проблемами, опытный, но не владеющий современными математическими и вычислительными методами, руководитель, перебирает часть вариантов, отбрасывая некоторые, совсем уж негодные. При этом его выбор, может оказаться неудачным, потому что

лучший вариант и близкие к нему по качеству хорошие варианты могут оказаться им нерассмотренными. В таких случаях целесообразно прибегать к научным методам оптимизации. Однако при этом, не следует впадать в часто встречающуюся ошибку, думая, что пакет прикладных программ во всех случаях найдет решение в любой сложной ситуации и подскажет план действий любому, пусть даже и не знающему теорию, человеку. Многообразие возникающих жизненных проблем может оказаться существенно шире ряда классических, уже решенных математически задач. При столкновении с новой задачей следует сначала выяснить, не сводится ли она к уже рассмотренной, классической проблеме. Например, в уже рассмотренной задаче оптимальной маршрутизации могут быть разнообразные варианты. Перечислим некоторые классические задачи и их варианты.

Задача оптимальной маршрутизации

1. **Классический вариант**. Требуется проехать из заданного пункта сети дорог в другой пункт этой сети, выбрав кратчайший по длине маршрут.
2. Требуется проехать из заданного пункта сети дорог в другой пункт этой сети, выбрав самый дешевый маршрут. Эта задача сразу сводится к классической задаче заменой длин ветвей на стоимость их проезда.
3. Требуется проехать из заданного пункта сети дорог в другой пункт этой сети, выбрав маршрут, по которому можно проехать за кратчайшее время. Эта задача сразу сводится к классической задаче заменой длин ветвей на времена их проезда.
4. Требуется проехать из заданного пункта сети дорог в другой пункт этой сети, выбрав маршрут с наименьшим числом пересадок с одного вида транспорта на другой. Такая задача сводится к классической заменой всех длин ветвей на нули и учреждением новых ветвей в пунктах пересадок, которые оцифровываются единицами.
5. Требуется проехать из заданного пункта сети дорог в другой пункт этой сети, выбрав маршрут с наименьшим временем пересадок с одного вида транспорта на другой. Такая задача сводится к классической заменой всех длин ветвей на нули и учреждением новых ветвей в пунктах пересадок, которые оцифровываются временами ожидания транспорта.
6. Требуется проехать из заданного пункта сети дорог в другой пункт этой сети, выбрав маршрут, по которому можно проехать за кратчайшее время с учетом ожиданий в некоторых пунктах (например, на границе государств, или ожидания при пересадках на другой вид транспорта). Эта задача сразу сводится к классической задаче заменой длин ветвей на времена их проезда и добавлением специальных ветвей с временами ожидания в указанных пунктах.

7. Требуется проехать из заданного пункта сети дорог в другой пункт этой сети, выбрав кратчайший по времени проезда маршрут с учетом того, что это время проезда может меняться во время суток из-за расписания движения транспорта и его вынужденного ожидания при пересадках.

8. К задаче оптимальной маршрутизации сводятся задача нахождения лучшего варианта прокладки по местности газопроводов, автомобильных и шоссейных дорог. Для этого на план местности наносится координатная сетка рис. 2.

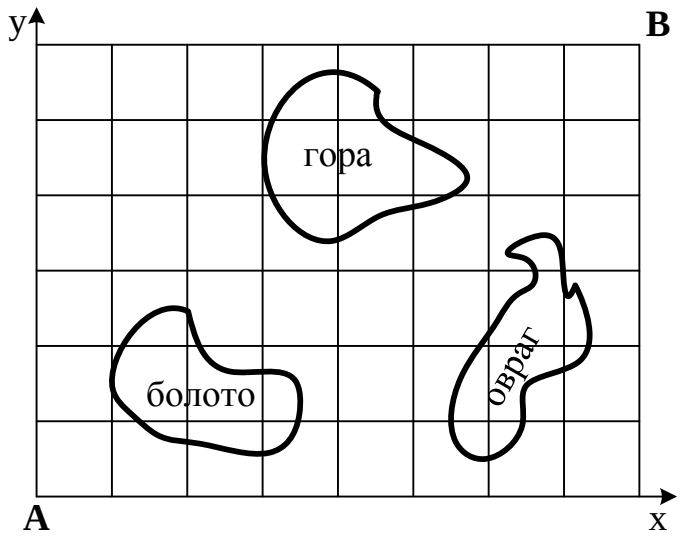


Рис.2 Координатная сеть для нахождения самого дешевого маршрута железной дороги на местности между пунктами A и B

При этом каждый горизонтальный и каждый вертикальный участок сети оцифровывается величиной стоимости строительства. З учитывается рельеф местности (холмы надо срывать, низины и болота засыпать), другие особенности местности (необходимость строительства моста, эстакады через реку, или овраг, наличие леса и т.п.). Далее решается задача под номером два.

9. К задаче оптимальной маршрутизации сводятся задача нахождения оптимального плана набора высоты самолетом. Требуется набрать заданную высоту, потратив наименьший объем горючего. Эта задача математически мало отличается от задачи 8.

Транспортная задача в сетевой форме

1. **Классическая транспортная задача линейного программирования в сетевой форме**. Задана сеть, для каждой ветви которой указана стоимость провоза по ней единицы груза. По сети требуется

осуществить перевозку груза одного и того же вида и при этом найти план перевозок наименьшей стоимости. При этом в части вершин сети находятся пункты отправления грузов (ПО – склады товара) и пункты назначения (ПН – магазины, заказчики товара). Величины a_i запасов в пунктах отправления заданы, Также заданы величины заказов b_j всех пунктов назначения. Кроме этих пунктов существуют остальные вершины графа – промежуточные пункты. В классической транспортной задаче соблюдается условие баланса – сумма запасов в пунктах отправления равна сумме заказов пунктов назначения. Известны стоимости провоза единицы груза c_k по каждой ветви графа

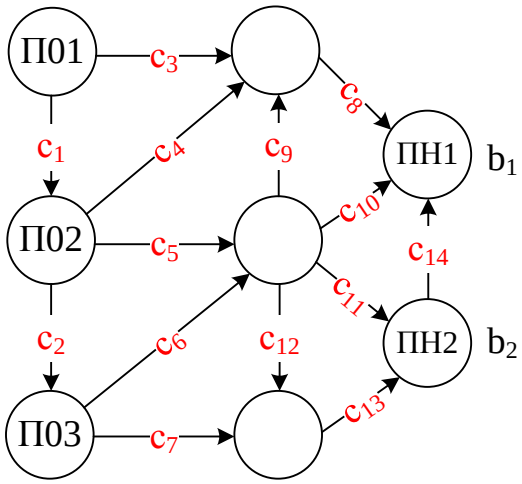


Рис.3. Схема движения в примере классической транспортной задачи

2. **Несбалансированная транспортная задача.** Эта задача отличается от классической задачи тем, что отсутствует баланс. При этом возможны два варианта. В первом из них суммарные запасы пунктов отправления превосходят суммарные запасы пунктов назначения. Такая задача легко сводится к классической добавлением фиктивного пункта назначения (помойки). Считается при этом, что стоимость сброса из каждого пункта отправления груза в «помойку» фиктивно очень велика (чтобы туда сбрасывался продукт только в крайнем случае). Другой вариант отсутствия баланса заключается в том, что суммарный запаса продуктов в пунктах отправления меньше суммарных заказов пунктов назначения. Такая задача сводится к классической задаче добавлением фиктивного пункта отправления. Кроме того, в такой несбалансированной задаче может быть и еще ограничения специального вида. Могут быть указана часть пунктов отправления, заказы которых обязаны быть удовлетворены, или удовлетворены, по крайней мере, в некоторой части (например, хотя бы, на половину). Такая задача называется задача с предписанными поставками.

3. **Транспортная задача с ограничениями на пропускную способность ветвей сети.** Эта задача отличается от классической задачи тем, что для части ветвей указаны объемы перевозок, превосходить которые недопустимо.

4. **Транспортная задача с запрещенными коммуникациями.** Эта задача отличается от классической задачи тем, что граф не является полным. Другими словами, между некоторыми вершинами графа отсутствуют ветви. Именно такая задача является базисной транспортной задачей в сетевой форме.

Задача об оптимальном назначении в сетевой форме

1. Классическая задача об оптимальном назначении. Даны n различных станков и n типов деталей и известна стоимость изготовления на каждом из станков детали каждого вида. Требуется найти наиболее дешевый вариант изготовления деталей, при котором каждый станок будет изготовлять ровно один вид деталей. Вариант: имеется n должностей и n человек, причем известна производительность каждого человека на каждой должности; требуется найти наилучший вариант назначения людей на должности, при котором их суммарная производительность труда, выраженная в деньгах, будет наибольшей.

2. Та же задача, но не сбалансированная. Имеется n должностей и m человек.

3. Та же задача, но некоторые люди вообще не могут быть назначены на некоторые работы. Предварительно представляет интерес входная задача: «существует ли назначение вообще?».

Задача оптимального распределения ресурсов

1. **Классическая задача оптимального распределения ресурсов.** Имеется некоторая сумма средств a для возможного финансирования n не связанных между собой проектов. Если вложить $x_i \geq 0$ средств в $i-$ый проект, то будет получен доход в объеме $g_i(x_i)$, а следовательно и экономический эффект

$$f_i(x_i) = g_i(x_i) - x_i.$$

Требуется так распределить общий объем средств, чтобы общий экономический эффект от всех финансируемых проектов был наибольшим. При этом допускается возможность отсутствия финансирования некоторых из проектов.

2. Задача, которая отличается от классической задачи оптимального распределения ресурсов тем, что допускается недоиспользование общего объема выделяемых средств.

3. Задача, которая отличается от классической задачи оптимального распределения ресурсов тем, что функции прибыли f_i зависят не только от своего, но и от других переменных, что означает существование связей между проектами.

Задача сетевого планирования производства (PERT и МКП)

1. Планируется производство продукции сложного вида, требующей выполнения целого ряда различных и связанных между собой программ. Например, это может быть строительство нового завода, или планирования производства крупного корабля, строительство большого моста, нового вида ракет и т. п. При этом известно расчетное время каждого элемента плана. Например, при строительстве может быть известно время разработки плана строительства и чертежей, известно время подведения коммуникаций к строительной площадке, время строительства жилищ и элементов инфраструктуры для рабочих и т.п. Известна, последовательность этих работ. Заданы номинальные времена производства каждой из работ. Требуется найти время выполнения всего всей программы строительства объекта, а также требуется найти все те работы, затяжка производства которых приводит к срыву планов и увеличению сроков осуществления всего проекта. Кроме того для всех остальных работ требуется найти резервы времени, что при затяжке выполнения работ на меньшие величины, чем резервы времени, еще не происходит срыва плана времени завершения всего проекта.

2. Дополнительно требуется найти все те виды работ, затяжка выполнения которых на заданное время приводит к увеличению сроков осуществления всего проекта.

Задача коммивояжера

1. Бродячий торговец (коммивояжер) должен обойти по сети дорог несколько конкретных пунктов расположения его клиентов и вернуться в исходную точку начала своего пути (на базу своих продуктов). Требуется выбрать кратчайший маршрут и при этом последовательность обходимых пунктов может быть произвольной.

2. К такой задаче сводится целый набор конкретных производственных проблем. Примером такой проблемы может быть выбор пути обслуживания роботом (смазка) станков в цеху завода.

3. Другим примером служит задача составления поезда маневровым локомотивом на большой железнодорожной станции. При этом требуется выбрать вагоны с различных путей, причем некоторые из вагонов находятся между другими вагонами, которые не входят в состав данного поезда. В этом случае маневровому локомотиву приходится перемещать и

такие вагоны. Нужно выбрать наилучший по объему маневровой работы вариант составления поезда из заданного числа вагонов.

Исключительно редко удается найти аналитическое решение сетевой оптимизационной задачи. В большинстве случаев для их решения используются алгоритмы, предназначенные для ЭВМ. Некоторые алгоритмы стали классическими в силу своей эффективности и распространенности решаемых ими задач. Перейдем к рассмотрению алгоритмов решения сетевых оптимизационных задач.

2. Динамическое программирование Ричарда Беллмана

2.1 Пояснение к теме

Правильный перевод книги американского математика и известного сотрудника корпорации RAND Р.Беллмана, вышедшей в 1950 году на его родине, правильно переводится с американского диалекта английского языка как Поэтапное планирование. Этот метод решения многочисленных задач сетевой оптимизации является одним из самых распространенных вариантов сокращения объема перебора вариантов при нахождении оптимального плана. Сначала рассмотрим некоторые примеры применения метода Беллмана, а затем сделаем небольшие обобщения и выводы.

Пример 2. (Частный случай древнекитайской игры Ним).

Рассмотрим следующую игру двух лиц (задача оптимизации в теории игр весьма существенна). Имеется некоторое число палочек (например, на рис.4 имеется десять палочек). Играя поочередно, каждый из двух игроков своим ходом обязан снять одну, или две, или три палочки. Тот игрок, который снял последнюю палочку, считается проигравшим.

Попытаемся найти лучшую стратегию игры. При этом временно отвлечемся от заданного числа палочек. Сначала рассмотрим ту ситуацию, когда перед ходом игрока осталась всего одна палочка. Здесь игрок проиграл, согласно условию игры, он обязан убрать эту последнюю палочку. Теперь рассмотрим ситуацию, в которой перед ходом игрока осталось ровно две палочки. В этой ситуации он может убрать одну, после чего останется единственная палочка и его оппонент, второй игрок, окажется в безнадежном положении с одной палочкой. Если осталось три или четыре палочки, то также игрок может своим ходом оставить ровно противнику ровно одну палочку и победить в игре. Если же перед ходом игрока осталась пять палочек, то эта положение для него стратегически проиграно. Как бы он не сыграл, он попадает на выигрышную ситуацию противника, отмеченную на рис.4 значком плюс.

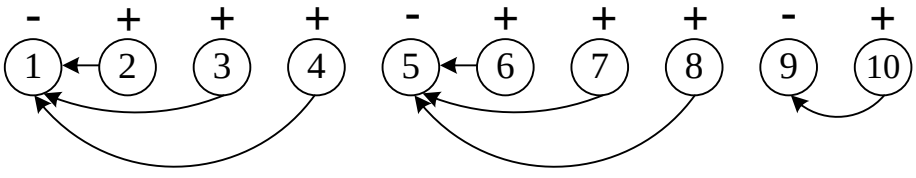


Рис. 4. Граф состояний игры примера 2

В этом случае при дальнейшей правильной игре выиграет его оппонент. Если же осталось шесть, семь или восемь палочек, то игрок своим ходом может довести их число до пяти и поставить противника в стратегически проигранное положение. С помощью рисунка 4 можно сделать верное предположение, что стратегически проигранные ситуации в игре соответствуют числам палочек, образующих арифметическую прогрессию с общим членом $a_n = 4n - 3$. Такие ситуации отмечены на рис.4 знаком минус. Все остальные положения отмечены знаком плюс. Игроку, находящемуся перед иным числом палочек следует играть на такое ближайшее число палочек, которое и которое отмечено знаком минус. В этом состоит выигрышная стратегия. В частности, если палочек было десять, то нужно, убрав одну, довести их число до девяти – ситуации, отмеченной знаком минус. На любой следующий ход противника следует ответить доведением числа палочек до пяти. На любой дальнейший ход противника следует ответить доведением числа палочек до одной и победить.

Пример 3. (Игра в шахматного короля). Рассмотрим игру двух лиц на шахматной доске размерами 8х8. Единственная фигура «король», в отличие от шахматной фигуры, может быть передвинута игроком во время его хода только на соседнее верхнее поле, или на соседнее правое поле, или по диагонали на одно поле на северо-восток. В начале игры король находится в левом нижнем поле. Противники играют только этой фигурой, перемещая ее своим ходом поочередно. В силу описанных условий игры, она закончится в правом верхнем углу доски. Тот из игроков, который поставит «короля» своим ходом на это поле, считается проигравшим.

Найдем выигрывающую стратегию в этой игре. Рассмотрим рис.5

+	-	+	-	+	-	+	-
-	-	-	-	-	-	-	+
-	+	-	+	-	+	-	-
-	-	-	-	-	-	-	+
-	+	-	+	-	+	-	-
-	-	-	-	-	-	-	+
-	+	-	+	-	+	-	-
	-	-	-	-	-	-	+

Рис.5. Выигрышная стратегия к примеру 3

Если «король» находится на одно поле левее последней клетки (находящейся в верхнем правом углу доски), то такое положение перед ходом противника является проигрышным, а для другого игрока –

выигрышем. То же относится и к положению «короля» на клетке, граничащей с последней, но расположенной под ней. Эти клетки отметим знаком плюс. На них следует играть. Клетки, из которых за один ход противник может попасть в клетки, отмеченные знаком плюс, являются проигрышными, они обозначаются знаком минус. На них играть не следует. Если клетка такова, что из нее противник может попасть только на клетки со знаком минус, то они обозначаются знаком плюс. Так можно заполнить всю доску, что и сделано на рис.5. Выигрышная стратегия начинающего игрока состоит в игре на клетки со знаком плюс. Если начинает игру противник, то он выиграет при выборе правильной стратегии. Но стоит ему хотя бы раз отклониться от нее, выигрыш может выпасть из его рук.

Пример 4. (Простейшая задача оптимальной маршрутизации). На рис.6 представлена сеть, для которой предлагается решить задачу нахождения кратчайшего маршрута из вершины А в вершину В.

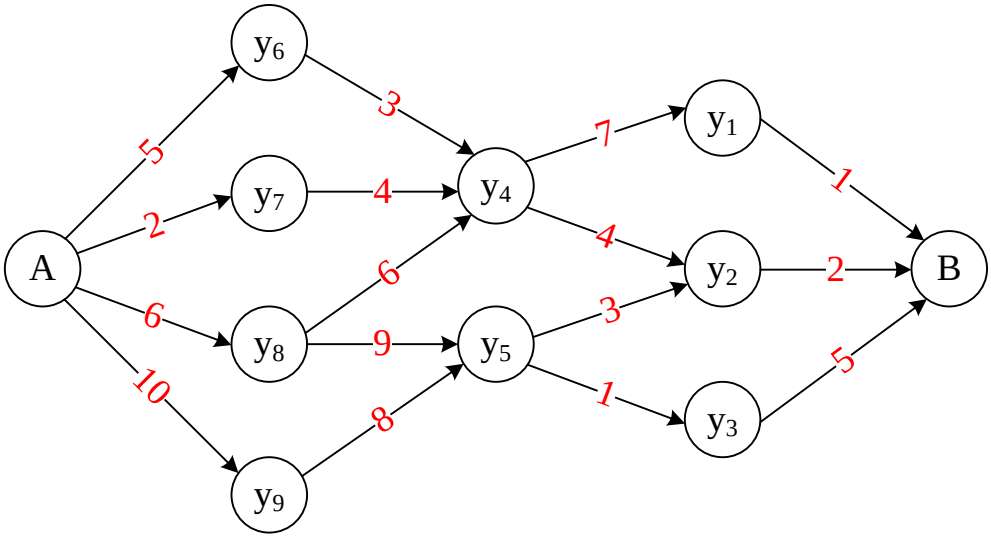


Рис. 6. Орграф к примеру 4

Решение

Отметим важную особенность предложенной в примере сети. Сеть содержит три вертикальные набора вершин. Первый набор состоит из вершин y, y_2, y_3. Второй набор состоит из вершин y_5 и y_6. Третий набор состоит из вершин y_6, y_7, y_8 и y_9. Вершины каждого из наборов не имеют общих ветвей. Из вершин первого набора можно попасть в конечную вершину маршрута В за один шаг (по единственной ветви). Из вершин второго набора можно попасть в конечную вершину маршрута В за два шага (по цепи из двух ветвей, первая из которых приводит в одну из вершин первого набора). Из вершин третьего набора можно попасть в конечную вершину маршрута В за три шага (по цепи из трех ветвей, проходя одну вершину второго набора и одну вершину первого набора). Эти особенности позволяют решить поставленную задачу оптимальной маршрутизации следующим образом. Будем искать ту наименьшую длину маршрута, который ведет из каждой вершины в пункт В. Для каждый из вершин первого набора существует единственный путь в вершину В. Поэтому $y_1 =$

$1, y_2 = 2$ и $y_3 = 5$. Из вершины y_4 можно идти в вершину y_1. В этом случае длина цепи будет равна 7+1=8. Это хуже, чем идти их этой вершины в вершину y_2. В последнем случае длина цепи составляет 4+2=6. Поэтому длина кратчайшего пути из четвертой вершины в вершину В равна шести: $y_4 = 6$. Рассуждая аналогично, найдем длину кратчайшего пути из пятой вершины в вершину В, как минимальную из двух сумм, каждая из которых составлена из длины ветви и длины кратчайшего пути до В из вершины, в которую ведет эта ветвь. Так получим, что $y_5 = 5$. Действуя таким же образом, найдем все длины кратчайших путей из всех оставшихся вершин третьего их набора: $y_6 = 9, y_7 = 10. y_8 = 12, y_9 = 13$. На последнем шаге алгоритма найдем ответ, выбрав лучший из четырех вариантов. Получим $y_A = 12$. Всеми найденными длинами кратчайших маршрутов обозначены вершины сети на рис.7.

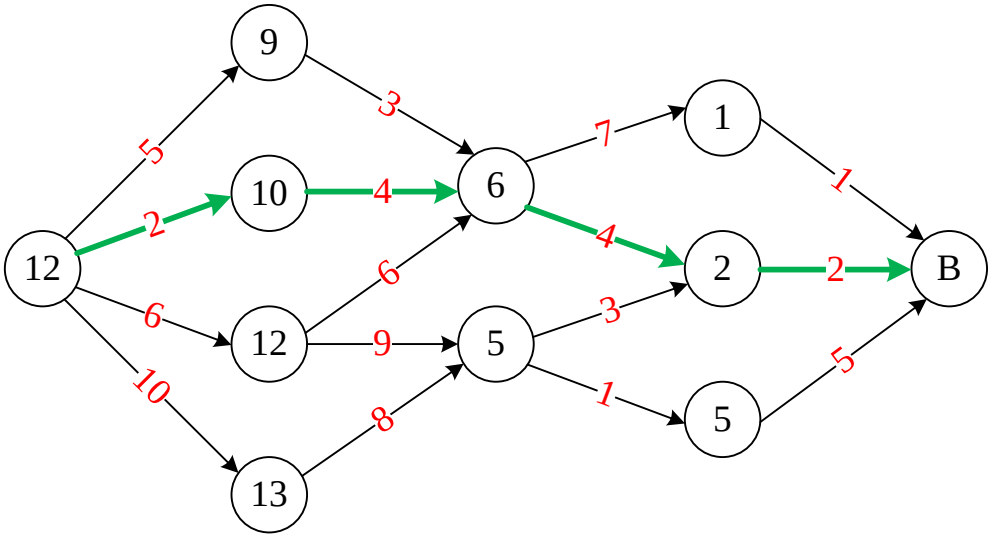


Рис.7. Оптимальный маршрут в примере 4, полученный с помощью алгоритма динамического программирования Р.Беллмана

Длина кратчайшего маршрута найдена. Однако представляет интерес и сам маршрут. Его нахождение осуществляется с помощью обратного хода алгоритма Беллмана. Если вычесть из длины y_A длину ветви кратчайшего маршрута в точку В, то должна быть получена длина кратчайшего маршрута в эту точку, из конца этой ветви. При проверке четырех вершин третьего набора (см. рис 7) найдем ту ветвь, которая удовлетворяет этому условию. Эта ветвь, ведет в седьмую вершину. Из нее идет единственная ветвь, которая приводит в четвертую вершину. Оттуда имеется два пути. Но лишь один из них удовлетворяет тому условию, что сумма длины ветви и длины пути из вершины конца ветви в В, равна длина пути из начала ветви в В. Так находится вторая вершина, из которой существует единственный путь в В. Наилучший маршрут найден. На рис.7 он показан зеленым цветом. Последовательность его вершин такова:

$$A \to 7 \to 4 \to 2 \to B.$$

Предложенная в примере 4 задача и алгоритм, решающий ее, является базовыми для решения многих задач. Выделенные наборы вершин соответствуют этапам (шагам) работы алгоритма. Часто эти этапы имеют физический или экономический смысл. Поэтому алгоритм Беллмана и называется динамическим программированием (поэтапным планированием).

Отметим две важные особенности трех последних задач и методов их решения.

1) Первая особенность заключается в том, что метод решения задачи дал не только ответ на поставленный вопрос, но и позволил решить целое семейство родственных задач. Во втором примере найдено решение задачи не только с десятью палочками, но и с произвольным числом палочек. В третьем примере, можно начинать игру на доске не только из левого нижнего угла, но из любой точки доски. В четвертом примере найдены длины кратчайших маршрутов в точку В не только из точки А, но и из каждой вершины сети. Во всех примерах оказалось, что удобнее решить не отдельную задачу, а сразу все задачи некоторого семейства. При решении каждой новой задачи требуется найти такое семейство, одним из элементов которого является как элемент исходная задача. Такой способ решения проблем называется **методом инвариантного погружения** задачи в семейство других, родственных ей задач**.** Этот метод первым применил Лагранж при нахождении оценки остаточного члена в формуле Тейлора.

2) Другая особенность рассмотренных задач заключается в специфическом виде функции цели, точнее, функционала. Во всех трех примерах искалось не число, а последовательность действий. Во втором и третьем примерах искалась стратегия выигрыша – последовательность ходов. В четвертом примере искался наилучший маршрут – последовательность вершин и ветвей сети. Но качество ответа в примерах можно было охарактеризовать одним числом. В игровых примерах – единицей в случае победы, прихода в конечную точку противника, в задаче оптимальной маршрутизации – длиной маршрута. Зависимость числа от функции (в данном случае от последовательности – функции натурального аргумента) называется функционалом. В случае задачи оптимальной маршрутизации функционал был аддитивным. Это значит, что его значение получается в виде суммы чисел на отдельных этапах. Примером аддитивных функционалов служат расход топлива поезда с дизельным локомотивом при его движении и прохождении отдельных участков пути. В случае игровых примеров функционал называется терминальным. Это означает то, что его характеризует только конечная точка в игре. Примером терминального функционала служит точность попадания в цель ракеты, выраженная в расстоянии ее до центра мишени в момент взрыва.

2.2 Принцип оптимальности Беллмана

2.2.1 Пояснение к теме. Формулировка принципа оптимальности динамического программирования

Принцип оптимальности динамического программирования Р.Беллмана: часть оптимальной траектории является также оптимальной траекторией.

Примером оптимальной траектории на пересеченной местности является набитая тропа в тайге. Он используется путешественниками и зверями не только при движении из начального пункта в конечный пункт, но и для движения между промежуточными пунктами тропы. Путник, по недомыслию сошедший с тропы, может оказаться на неоптимальном маршруте, и даже попасть в затруднительное положение, оказавшись на крутом обрыве, в чащобе или болоте.

Легко проверить, что принцип оптимальности динамического программирования верен для всех трех рассмотренных выше примеров. Оказывается, что этот принцип оптимальности верен для положительных аддитивных функционалов, для терминальных функционалов и для линейных выпуклых комбинаций положительных аддитивных и терминальных функционалов. Поскольку такие виды функционалов широко распространены, то и динамическое программирование стало обычным методом не только в руках прикладного математика, но и экономиста и инженера.

2.2.2 Доказательство принципа оптимальности для аддитивных функционалов

На рис.8 представлен вид сверху на гористую поверхность.

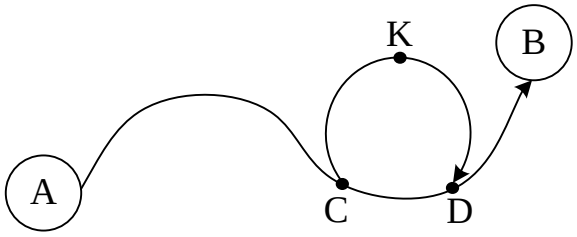


Рис. 8. К доказательству принципа оптимальности Беллмана для аддитивных функционалов.

Пусть там кратчайшим путем между точками кривая A и B служит кривая тропа ACDB (в горах кратчайший путь не прямая, по дороге возможны подъему и пропасти). Если кратчайшем путем между точками C и D этой тропы была бы не кривая CD, а вычурная дуга CKD, то кратчайшем путем из A в B служила бы дорога ACKDB, а не указанная кривая. При этом эта дорога была бы настолько экономнее исходной, насколько ACKDB

короче CD. Но, по условию это не так. Полученное противоречие завершает доказательство.

2.2.3 Доказательство принципа оптимальности для терминальных функционалов

На рис. 9 представлен орграф, в котором следует найти маршрут, начинающийся в точке A, некоторая вершина которого должна быть самой близкой к точке B.

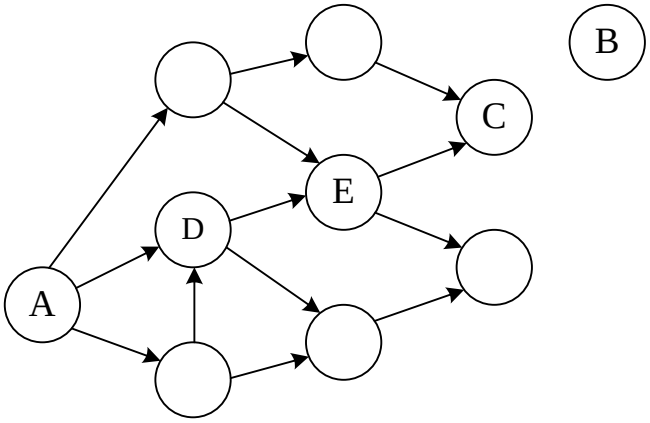


Рис. 9. К доказательству принципа оптимальности Беллмана для терминальных функционалов

Пусть это маршрут ADEC. При этом точка C является самой близкой к точке B на связном орграфе. Если взять произвольную точку маршрута, например точку D, и сойти с маршрута, то не возникнет возможности попасть в другую точку орграфа, более близкую к точке B, чем точка C, потому что такой не существует. Следовательно, сходить с оптимального маршрута нецелесообразно. Доказательство завершено.

2.3 Пример применения метода динамического программирования к планированию строительства

Пример 5. Пусть требуется выбрать оптимальный план строительства промышленного объекта. В первый год, возможно, выполнить полностью весь проект за 12 млрд. рублей. Но, за год можно выполнить и 80% всего объема работ, потратив на это 7 млрд. руб. Существуют варианты выполнения 60% объема работ со стоимостью 5 млрд. руб. и 40% объема работ за 3 млрд. рублей, а также и 20% объема работ за 30 млрд. рублей. Во второй год могут быть выполнены работы в различных объемах, с использованием различных объемов средств, в зависимости от стартового на конец предыдущего года объема выполненных работ. И т.д. до пятого года. Орграф, соответствующий описанной ситуации, показан на рис.10. С помощью простейшего алгоритма динамического программирования, описанного в примере 4, решена задача оптимальной маршрутизации и найден лучший план, соответствующий минимальной стоимости проекта. Этот план изображен в виде зеленой цепи на рис. 10. Его стоимость

составила 7,5 млрд. рублей, а продолжительность строительства равна трем годам. При этом за первый год следует выполнить 40% объема работ за 3млрд.руб., за второй год еще 20% за 2млрд.руб и за последний третий год 40% объема работ за 2,5млрд.руб.

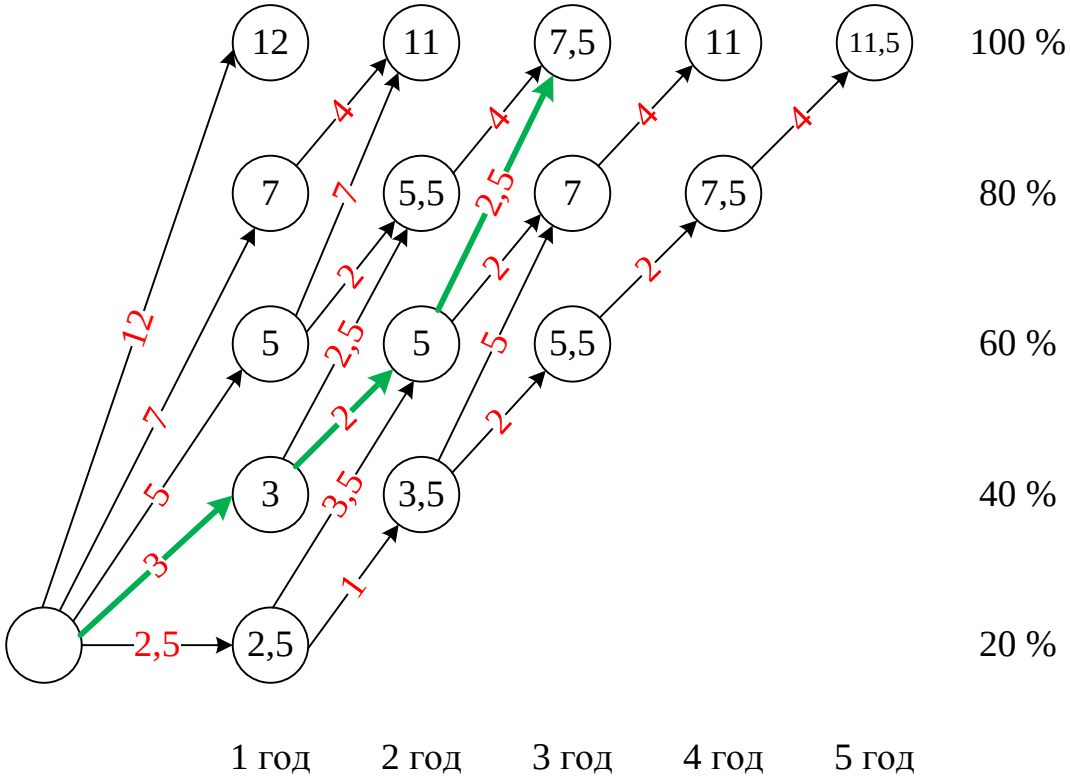


Рис. 10. Орграф к примеру 5

2.4 Решение задачи оптимального распределения ресурсов методом динамического программирования

2.4.1 Пояснение к теме. Дискретизация непрерывных оптимизационных задач. Бич размерности Беллмана

Дискретные сетевые задачи оптимизации широко используются для численного и приближенного нахождения решения целого ряда непрерывных оптимизационных задач. В их число входят задачи линейного и нелинейного программирования, различные вариационные задачи, задачи оптимальной адаптации и задачи оптимального управления. Подробнее остановимся на проблемах решения задач нелинейного программирования с помощью сетевых методов дискретной оптимизации. Постановка задачи нелинейного программирования (правильнее, планирования по нелинейным конечномерным моделям) выглядит следующим образом. Требуется найти глобальный максимум (в другом случае, минимум) заданной функции n переменных $F(x_1 \dots, x_n)$ с учетом системы ограничений, как правило, представляющей из себя систему нестрогих неравенств. Иногда ограничения представляют собой смешанную систему из нескольких нестрогих

неравенств и нескольких алгебраических уравнений. Задача нелинейного программирования может быть записана следующим образом:

$$\begin{cases} F(x_1 \dots, x_k) \to max \\ g_i(x_1 \dots, k) \le 0, \;\; i = 1, \dots, m; g_j(x_1 \dots, x_k) = 0, \;\; j = 1, \dots, m < k. \end{cases} \quad (1)$$

Для решения такой задачи можно применить метод дискретизации с некоторым шагом $h > 0$, считая, что значения всех переменных содержат целое число шагов. Например, если в задаче присутствует всего одна переменная, то при дискретизации на отрезке $[a, b]$ она будет принимать только следующие значения:

$$x = a, a + h, a + 2h, \dots, a + (n - 1)h = b. \quad (2)$$

Шаг h выбирается исследователем из соображений получения необходимой, но разумной точности ответа. Из формулы (2) можно найти число шагов n:

$$n = 1 + \left[\frac{|b - a|}{h}\right]. \quad (3)$$

В формуле (3) квадратными скобками показана целая часть действительного числа (наибольшее целое число, не превосходящее того, что стоит под знаком целой части). При реальных расчетах число шагов может меняться от десяти до нескольких десятков тысяч, но обычно оно не превосходит одну тысячу. Задачу нелинейного программирования с одним переменным (одномерную задачу оптимизации) нетрудно решить, т.к. после дискретизации она оказывается сведенной к перебору конечного и для ЭВМ небольшого, числа значений n. Однако, в случае наличия в задаче нескольких переменных (случай многомерной задачи), ее вычислительная трудность резко усложняется. Если переменных только два, и каждое из них может принимать n значений, то всевозможных соединений этих значений получается уже $n * n = n^2$. В случае k переменных (тогда говорят, что размерность задачи равна k) число всевозможных соединений становится равным n^k. Такое число может быть чрезвычайно большим. При достаточно умеренных параметрах: $n = 100$ и $k = 10$ получается число вариантов равное 10^{20}, которое превышает возможности по быстродействию ЭВМ. Из этого примера следует, что при большой размерности задачи, последняя становится не по силам ЭВМ. Это явление Беллман назвал «бичом размерности». Для преодоления влияния «бича размерности» используются различные численные методы. Общего универсального подхода для всех задач не существует. Но для отдельных типов задач разработаны эффективные алгоритмы. Одним из часто встречающихся типов задач, решаемых с помощью идей динамического программирования, является задача оптимального распределения ресурсов.

2.4.2 Применение динамического программирования к решению задачи оптимального распределения ресурсов. (Теория вопроса)

Рассмотрим классическую задачу оптимального распределения ресурсов.

$$\begin{cases} F(x_1, \dots, x_k) = \sum_{i=1}^{k} f_i(x_i) \to max \\ x_1 + x_2 + \cdots + x_k = a > 0 \\ x_i \geq 0, \qquad i = 1, \dots, k. \end{cases} \tag{4}$$

Произведем дискретизацию в математической модели (4):

$$x_1 = n_1 h, \dots, x_k = n_k h, \qquad n_i = 1, \dots, n, \qquad a = mh. \tag{5}$$

Подставим равенства (5) в систему (4) и получим:

$$\begin{cases} F(n_1 h, \dots, n_k h) = \sum_{i=1}^{k} f_i(n_i h) \to max \\ (n_1 + n_2 + \cdots + n_k)h = mh \Rightarrow n_1 + n_2 + \cdots + n_k = m. \\ i = 1, \dots, k. \end{cases} \tag{6}$$

В соответствии с методикой динамического программирования инвариантно погрузим задачу (6) в множество задач при $m = 1, 2, \dots$и k=1,2,…Далее для разработки алгоритма решения задачи (6) применяется принцип оптимальности Беллмана для аддитивных функционалов. Это позволительно сделать, потому что оптимизируемая (функция цели) $F(x_1, \dots, x_k)$, в силу первого равенства из системы (4), является суммой. Сначала рассматривается случай $k = 1$. В нем очевиден ответ: $x_1 = a, F = f_1(a)$. Экономический смысл этого этапа состоит в том, что все средства передаются первому проекту, а остальные проекты ничего не получают: $x_2 = \cdots = x_k = 0$. Затем рассматривается вариант, когда средства передаются двум первым проектам и оптимально распределяются между ними, в то время как остальным проектам никаких средств не оставляется. Для этого частного, но базисного для расчета случая, получим следующую задачу:

$$\begin{cases} F = f_1(n_1 h) + f_2(n_2 h) \to max \\ (n_1 + n_2)h = mh \Rightarrow n_1 + n_2 = m, \; n_1 = 0,1,.., m. \end{cases} \tag{7}$$

Исключим из уравнений (7) целочисленное переменное n_2:

$$\begin{cases} F = f_1(n_1 h) + f_2[(m - n_1)h] \to max \\ n_1 = 0,1,..,m. \end{cases} \quad (8)$$

Полученная задача (8) является одномерной (!) задачей целочисленной оптимизации. Ее можно эффективно решить на ЭВМ простым пассивным перебором. В ряде случаев можно решить задачу (8) и вручную без привлечения ЭВМ. Но следует ее решить для различных чисел m от единицы до номинального значения $m = \frac{a}{h}$. При таких решениях будет построена функция $f_{12}(m)$ оптимального распределения ресурсов между первыми двумя проектами в зависимости от их общего объема финансирования.

После этого строится функция $f_{123}(m)$ оптимального распределения ресурсов между тремя проектами:

$$\begin{cases} F = f_1(n_1 h) + f_2(n_2 h) + f_3(n_3 h) \to max \\ n_1,\ n_2,\ n_3 = 0,1,..,m. \end{cases} \quad (9)$$

Но при оптимальном распределении ресурсов между тремя проектами, некая часть будет передана первым двум из них. Эта часть должна быть оптимально распределена между двумя первыми проектами. Поэтому задача (9) эквивалентна следующей задаче оптимального распределения ресурсов:

$$\begin{cases} F = f_{12}(n_1 h) + f_3(n_3 h) \to max \\ n_1 + n_3 = 0,1,..,m. \end{cases} \quad (10)$$

Замечательным является то, что подобная задача является задачей оптимального распределения ресурсов между двумя (!) проектами – третьим и объединенным первым и вторым проектом. Но такая задача уже была решена выше. Ее решение выгодно оформить как подпрограмму и снова использовать. Таким образом, можно найти и оптимальное распределение ресурсов f_{1234} между первыми четырьмя проектами как наилучшее распределение ресурсов между четвертым проектом и оптимальным объединенным первым, вторым и третьим проектом. И вообще, можно найти оптимальное распределение ресурсов $f_{1\ldots k}$ между всеми проектами как наилучшее распределение ресурсов между последним по номеру проектом и оптимальным объединенным первым, вторым и т.д. – $(k-1)-$ым проектом. При этом, в частности, будет получено и решение исходной задачи.

2.4.3 Применение динамического программирования к решению задачи оптимального распределения ресурсов. (Пример)

Для окончательного уяснения алгоритма решения задачи оптимального распределения ресурсов рассмотрим конкретный числовой пример, в котором производится распределение ресурсов между тремя проектами.

Пример 6. Будем считать, что общий объем средств выделяемых на все проекты равен единице. Функции экономического эффекта заданы следующими:

$$f_1(x_1) = 3x_1 - 5x_1^2, f_2(x_2) = 2x_2 - 3x_2^2, f_1(x_1) = x_3 - x_3^{.2}.$$

Поэтому задача оптимального распределения ресурсов математически записывается следующим образом:

$$\begin{cases} F = 3x_1 - 5x_1^2 + 2x_2 - 3x_2^2 + x_3 - x_3^{.2}. \rightarrow max \\ x_1 + x_2 + x_3 = 1, \qquad x_1, x_2, x_3 \geq 0. \end{cases} \tag{11}$$

Для решения задачи введем шаг дискретизации $h = 0{,}1$. Вычислим значения всех предложенных функций с этим шагом и запишем их в первую таблицу 1.

Таблица 1

Исходные данные к расчету оптимального распределения ресурсов

x	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$f_1(x)$	0	0,25	0,4	0,45	0,4	0,25	0	0,3	0,8	1,3	2
$f_2(x)$	0	0,17	0,28	0,33	0,32	0,25	0,12	0,1	0,3	0,6	1
$f_3(x)$	0	0,09	0,16	0,21	0,24	0,25	0,24	0,21	0,16	0,9	0

Часто аналитические значения функций неизвестны. Но, тогда известны их табличные данные, полученные экспериментально. В таком случае таблица 1 содержит исходную информацию для дальнейших расчетов.

На первом этапе расчета найдем оптимальное распределение ресурсов между первыми двумя проектами. При этом сделаем десять отдельных расчетов $f_{12}(mh)$ при

$$m = 0{,}1, \dots, 10, h = 0{,}1.$$

При $\boldsymbol{m = 0}$ из таблицы 1 следует, что $f_{12}(0) = 0$.

При $\boldsymbol{m = 1}$ существует всего два варианта:

$f_{12}(0{,}1) = max(f_1(0{,}1), f_2(0{,}1)) = \mathbf{0{,}25}$, $\boldsymbol{y_1 = 0{,}1}$, $\boldsymbol{y_2 = 0}$.

При $\boldsymbol{m = 2}$ существует три варианта $(y_1, y_2) = (0{,}2, 0)$;

$$(\boldsymbol{y_1, y_2}) = (\mathbf{0, 1, 0, 1})$$ и

$(y_1, y_2) = (0, 0{,}2)$, оптимальный вариант второй $\boldsymbol{f_{12}(0,1) = 0{,}42}$.

При $\boldsymbol{m = 3}$ существует уже четыре варианта. Продолжая этот процесс расчетов, заполним второй, третий и четвертый столбцы таблицы 2.

Таблица 2

Результаты расчета оптимального распределения ресурсов

a=mh	f_{12}	y_1	y_2	f_{123}	x_1	x_2	x_3
0,1	0,25	0,1	0	0,25	0,1	0	0
0,2	0,42	0,1	0,1	0,42	0,1	0,1	0
0,3	0,57	0,2	0,1	0,57	0,2	0,1	0
0,4	0,68	0,3	0,1	0,68	0,3	0,1	0
0,5	0,73	0,3	0,2	0,77	0,3	0,1	0,1
0,6	0,75	0,3	0,3	0,84	0,3	0,1	0,2
0,7	0,75	0,3	0,4	0,89	0,3	0,2	0,2

0,8	0,72	0,4	0,4	0,94	0,3	0,2	0,3
0,9	0,65	0,4	0,5	0,97	0,3	0,2	0,4
1	0,52	0,4	0,6	**0,99**	**0,3**	**0,4**	**0,4**

На следующем этапе расчета повторяем алгоритм расчета смешанного плана двух проектов, одним из которых возьмем рассчитанный f_{12}, а вторым f_3. Результатом этого расчета являются числа в последних четырех столбцах таблицы 2. Ответом задачи служат последние четыре числа этой таблицы:

$$F_{max} = 0{,}99, x_1 = 0{,}3, x_2 = 0{,}4, x_3 = 0{,}3.$$

Из последнего расчета и из соображений предыдущего параграфа следует, что весь расчет представляет собой $k-1$ этап, на каждом из которых производится одномерная максимизация функции, вычисление которой состоит из операции сложения двух чисел. Такой расчет занимает доли секунды на современной ЭВМ даже при числе k элементарных функций, входящих в функцию цели, равным нескольким сотням. Эффективность алгоритма связана с тем, что задачу максимизации функции нескольких переменных здесь счастливо удалось свести к нескольким задачам максимизации функции одного переменного. Удалось избежать действия «бича размерности» благодаря использованию метода динамического программирования.

3. Алгоритмы решения задачи оптимальной маршрутизации

3.1 Пояснение к теме

В первом параграфе была поставлена задача нахождения кратчайшего маршрута между двумя вершинами сети. В примере 4 был предложен алгоритм решения этой задачи для простейшей сети, которая распадается на несколько последовательных рядов вершин. Здесь будут рассмотрены и обоснованы основные алгоритмы решения задачи оптимальной маршрутизации для сети произвольного вида, а также специального случая ациклического графа.

3.2.Алгоритм Форда-Беллмана

3.2.1 Пояснение к теме

Сначала рассмотрим пример, который показывает, что предложенный для примера 4 алгоритм, не может быть использован для произвольной сети.

Пример 7. На рис.11 показана сеть, состоящая всего из четырех вершин.

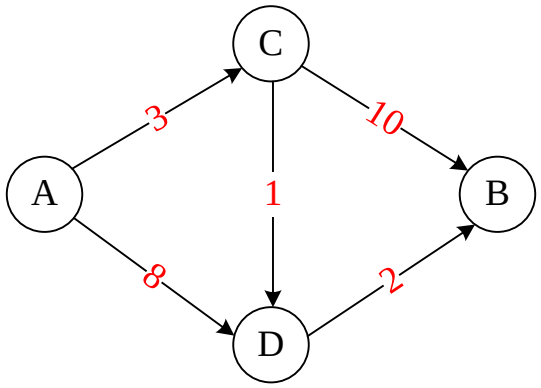


Рис. 11. Сеть к примеру 7

Кратчайший путь из вершины С в вершину В состоит из двухзвенной цепи CDB, длиной 3, что меньше прямого однозвенного пути СВ длиной 10. Такой путь невозможно найти с помощью простейшего алгоритма примера 3.

Кратчайший путь ACB из вершина А в вершину В состоит из трех звеньев т имеет длину 3+1+2=6.

Из рассмотрения примера 7 может возникнуть идея о том, что целесообразно сравнивать по длине маршруты, состоящие из одинакового числа ветвей орграфа. На основе этой идеи разработан алгоритм Форда-Беллмана.

3.2.2 Описание алгоритма Форда-Беллмана

Номер вершины В будем считать последним и обозначать его k. Обозначим длину ветви из вершины i в вершину j как L_{ij}.

1-ый этап (подготовительный). Исключим из рассмотрения все петли (если они имеются) в графе, потому что ими не целесообразно пользоваться, только увеличивая длину маршрута. Установим на каждой вершине новую условную петлю, с нулевой длиной. Тогда, остановка на месте в одной вершине может быть трактована как один пустой шаг алгоритма.

2-ой этап (подготовительный). Исключим из рассмотрения все не минимальные параллельные ветви в графе, если параллельные ветви в мультиграфе имеются. Напомним, что параллельными называются такие ветви в орграфе, начала и концы которых совпадают. Это мероприятие связано с тем, что, минимизируя длину маршрута, вообще не следует пользоваться более длинными ветвями.

3-й этап. Найдем все вершины, из которых можно попасть в вершину В по одной ветви (за шаг). Пусть это вершины с номерами $i_{11}, \dots, i_{m_1 1}$. Найдем длину маршрута B_{i1} из этих вершин следующим образом:

$$B_{i1} = L_{ik}.$$

Действительно, длина маршрута из единственной ветви равна длине этой ветви. Положим число ветвей равным единице (k=1).

4-ый этап. k=k+1.

5-ый этап. Из всех возможных вершин найдем кратчайшие пути, состоящие из (k+1)-ой ветви (сначала будет две ветви). Для этого из каждой вершины найдем ветви, ведущие в вершины, для которых найден путь из k ветвей минимальной длины. Сложив длину этого пути с длиной ветви, Получим длину возможного пути, ведущий из рассматриваемой вершины в конечный пункт B. Найдем все такие длины путей, ведущие из рассматриваемой вершины, и выберем из них наименьший. Перейдем к следующей вершине, пока не исчерпаем их все.

Все это время проверяем наличие изменений в оценке длин кратчайших путей за k м за k+1 ветвь пути.

6-ой этап. Если изменения произошли, то переходим к четвертому этапу. Если изменения не произошли, то переходим к седьмому этапу. Длина кратчайшего маршрута найдена.

7-ой этап. Обратный ход алгоритма для нахождения самого кратчайшего маршрута, совпадающий с обратным ходом алгоритма, описанного в примере 4.

Пример 8. На рис. 12 приведена сеть для нахождения кратчайшего пути из вершины X_1 в вершину X_2. Требуется найти кратчайший путь методом Форда-Беллмана.

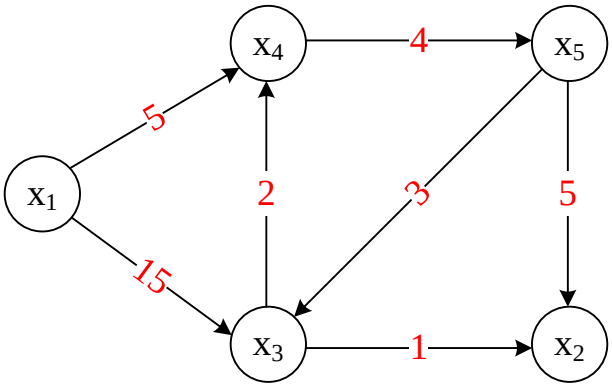


Рис.12. Сеть к примеру 8

Решение

На рис. 13 А, 13 Б, 13 В и 13 Г показаны оптимальные длины маршрутов, соответственно, из одной, двух, трех и четырех ветвей, ведущие из каждой вершины орграфа во вторую вершину. Все результаты получены с помощью алгоритма Форда-Беллмана.

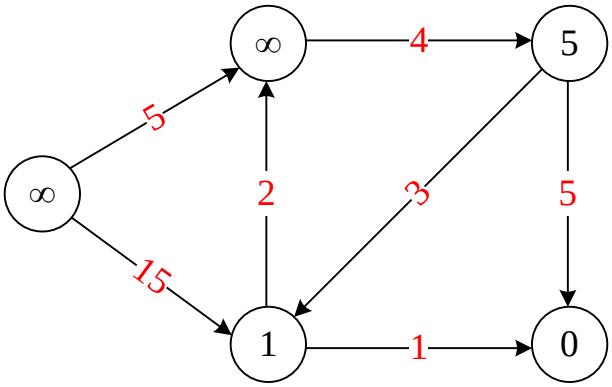


Рис.13 А. Результаты расчета длин маршрутов, состоящих из одной ветви

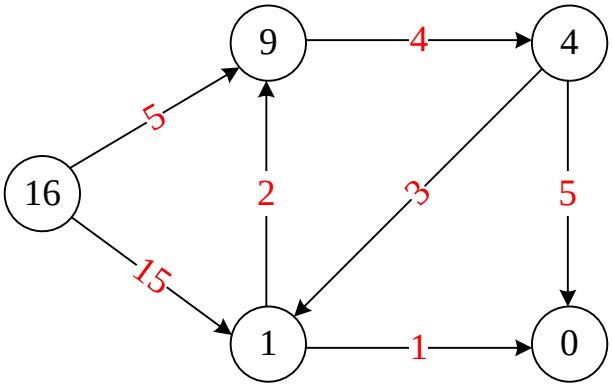


Рис.13 Б. Результаты расчета длин маршрутов, состоящих из двух ветвей

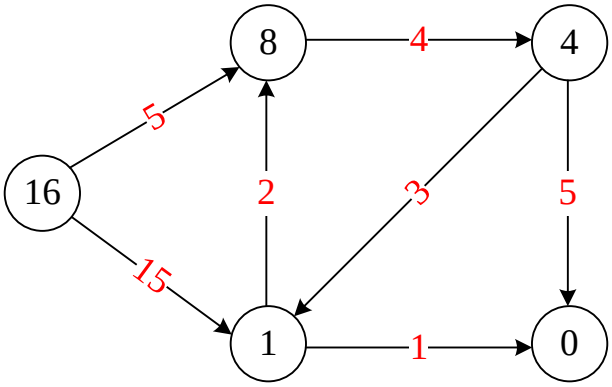


Рис.13 В. Результаты расчета длин маршрутов, состоящих из трех ветвей

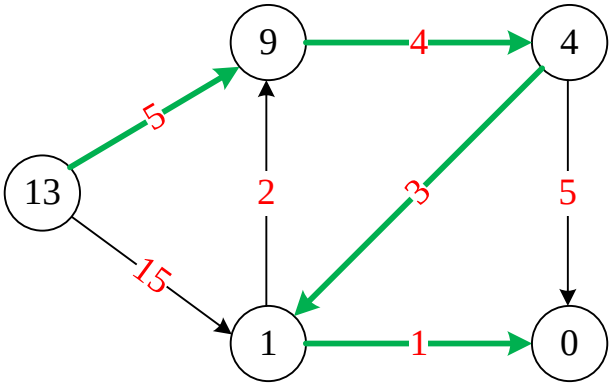


Рис.13 Г. Результаты расчета длин маршрутов, состоящих из четырех ветвей.

Зеленым цветом показан оптимальный маршрут из первой вершины во вторую. Это маршрут 1-4-5-3-2.

Алгоритм Форда-Беллмана является универсальным алгоритмом для решения задач маршрутизации. Он работает во всех случаях, когда эта задача имеет смысл, в том числе и в тех, когда некоторые длины ветвей отрицательны, но не образуют циклов со всеми отрицательными ветвями.

3.3 Алгоритм Шимбелла-Оттермана

3.3.1 Пояснения к теме. Шимбеловские операции с числами

Этот алгоритм является матричной версией алгоритма Форда-Беллмана. Вся исходная информация для работы алгоритма Шимбелла-Оттермана заключена в матрице смежности орграфа. Размерность этой квадратной матрицы равна числу вершин графа. Элементами матрицы служат длины ветвей. При этом все элементы главной диагонали равны нулям:
$L_{ii} = 0, i = 1, \dots, n$, а элемент орграфа L_{ij} равен длине ветви, ведущей из i-ой вершины орграфа в его j-ую его вершину. Если такая ветвь отсутствует, то длина считается бесконечностью.

Определим две алгебраические операции над положительными числами, которые используются в рассматриваемом алгоритме. Эти операции будем называть шимбелловскими, хотя они известны давно математикам, специалистам по абстрактной алгебре. Шимбелловским сложением $a \oplus b$ двух чисел неотрицательных a и b называется наименьшее из этих чисел:

$$a \oplus b = min(a, b). \quad (12)$$

Шимбелловским умножением $a \oplus b$ двух неотрицательных чисел a и b называется обычная сумма этих чисел:

$$a \otimes b = a + b. \quad (13)$$

Докажем, что Шимбелловские операции обладают некоторыми свойствами операций алгебраического поля.

1) Шимбелловская операция сложения коммутативна, потому что

$$\min(a, b) = \min(b, a) \Rightarrow a \oplus b = b \oplus a. \quad (14)$$

2) Существует ноль, им служит бесконечность:

$$a \oplus \infty = \infty \oplus a = a. \quad (15)$$

Последнее равенство верно, потому что

$$\min(a, \infty) = \min(\infty, a) = a.$$

3) Шимбелловская операция сложения коммутативна, потому что коммутативна операция обычного сложения:

$$a + b = b + a \Rightarrow a \otimes b = b \otimes a. \quad (16)$$

4) Шимбелловской единицей служит обычный ноль:

$$0 \otimes a = a \otimes 0 = a. \quad (17)$$

5) Имеет место дистрибутивность шимбелловского умножения относительно шимбелловского сложения:

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c). \quad (18)$$

Для доказательства (см. задачу 7) достаточно проверить равенство

$$a + min(b, c) = min[(a + b), (a + c)].$$

Замечание 1. (Линейное шимбелловское уравнение) Рассмотрим линейное уравнение с шимбеловскими операциями:

$$x = a \otimes x + b. \quad (19)$$

В обычных обозначениях уравнение (19) принимает следующий вид:

$$x = min[(a + x), b]. \quad (20)$$

Равенство возможно в двух случаях: либо $x = x + a \leq b,$ либо $x = b \leq a + x$. Первый случай возможен только при $a = 0$. Второй случай возможен при положительных значениях a. Поэтому, при не положительных значениях параметра a уравнение (19) не имеет решения.

Замечание 2. (Сведение задачи оптимальной маршрутизации к задаче решения линейных уравнений орграфа в шимбелловских операциях).

Пусть к первой вершине некоторого орграфа подходят ветви от остальных вершин, в том числе имеется и петля от первой вершины. Тогда шимбелловское уравнение орграфа примет следующий вид:

$$x_1 = (a_1 \otimes x_1) \oplus (a_2 \otimes x_2) \oplus \ldots \oplus (a_k \otimes x_k). \quad (21)$$

Переходя к обычным операциям, получим, что

$$x_1 = min[(a_1 + x_1), (a_2 + x_2), \ldots, (a_k + x_k)]. \quad (22)$$

В силу первого замечания величина a_1 должна быть положительна, иначе уравнение (22) не имеет решения. Но в этом случае $x_1 \neq a_1 + x_1$. Поэтому, уравнение (22) оказывается эквивалентным следующему уравнению:

$$x_1 == min[(a_2 + x_2), \ldots, (a_k + x_k)]. \quad (23)$$

Но уравнению (23) соответствует ситуация задачи оптимальной маршрутизации, когда значение наименьшей длины маршрута из первой вершину в некоторую другую вершину В равно наименьшей из сумм длин исходящих из первой вершины ветвей и наименьших длин маршрутов из тех вершин, где эти ветви заканчиваются. Находится и объяснение требования отсутствие петель с отрицательными и нулевыми значениями. В случае существования отрицательных циклов можно, многократно проходя их, получить маршрут сколь угодно малой общей длины. Такая ситуация должна быть отвергнута.

Замечание 3. Из сказанного следует, что алгоритм Форда-Беллмана соответствует алгоритму исключения вершины, но в шимбелловских операциях.

3.3.2 Формулировка алгоритма Шимбелла-Оттермана. Пример расчета

Матрица смежности орграфа является матрицей достижимостей из каждой вершины в каждой другой вершины за движение по одной ветви.

Алгоритм Шимбелла-Оттермана

Для того, чтобы попасть из некоторых вершин вершин в другие вершины, согласно соображением предыдущего пункта, следует перемножить две матрицы смежности, но шимбелловским образом. Для получения длин маршрутов, состоящих из n ветвей, следует шимбелловски возвести в $n-$ую степень матрицу смежности графа.

Пример 8. Решим задачу примера 7 с помощью алгоритма

Шимбелла-Оттермана. Матрица смежности для этого графа имеет следующий вид:

$$A=\begin{vmatrix} 0 & \infty & 15 & 5 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 4 \\ \infty & 5 & 3 & \infty & 0 \end{vmatrix}.$$

Нахождение длин оптимальных маршрутов между всеми парами вершин с помощью алгоритма Шимбелла-Оттермана

Возведем в квадрат матрицу A, с использованием шимбелловских операций:

$$A^2=B=A\otimes A=\begin{vmatrix} 0 & \infty & 15 & 5 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 4 \\ \infty & 5 & 3 & \infty & 0 \end{vmatrix}\otimes\begin{vmatrix} 0 & \infty & 15 & 5 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 4 \\ \infty & 5 & 3 & \infty & 0 \end{vmatrix}=$$

$$=\begin{vmatrix} 0 & 16 & 15 & 5 & 9 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & 6 \\ \infty & 9 & 7 & 0 & 4 \\ \infty & 4 & 3 & 5 & 0 \end{vmatrix}.$$

Покажем, как был рассчитан элемент квадрата матрицы, стоящий в ее первой строке на втором месте. При умножении матриц действует правило «умножения строчки на столбец»:

$$b_{12}=(0\otimes\infty)\oplus(\infty\otimes 0)\oplus(15\otimes 1)\oplus(5\otimes\infty)\oplus(\infty\otimes 5)=$$
$$=min(0+\infty,\infty+0,15+1,\quad 5+\infty,\infty+5)=min(\infty,\infty,16,\infty,\infty)=16.$$

Попутно будем заполнять матрицу переходов Q_2. При умножении матриц можно всякий раз при умножении $i-$ой строки на $j-$ый столбец

записывать значения номеров строк m, в которых образовалась минимальная сумма. Эти номера можно записывать в матрицу переходов на место (i, j). Число матриц переходов равно числу степеней матриц. Если требуется найти оптимальный путь x в точку y, то сначала находятся по последней матрице перехода элемент $m_1 = q_{xy}$. Затем, по предпоследней матрице переходов находится элемент $m_2 = q_{m_1 y}$ и так далее. Оптимальный маршрут можно записать следующим образом:

$$x, m_1, m_2, \dots, m_k, y.$$

В приведенном выше единичном расчете минимум дало третье слагаемое. Поэтому элемент $q_{12} = 3$. Заполненная матрица переходов имеет следующий вид:

$$Q_2 = \begin{vmatrix} - & 3 & 1 & 1 & 4 \\ - & - & - & - & - \\ - & 3 & - & 3 & 4 \\ - & 5 & 5 & - & 4 \\ - & 3 & 3 & 1 & - \end{vmatrix}.$$

Возведем в куб матрицу A, с использованием шимбелловских операций:

$$A^3 = C = B \otimes A = \begin{vmatrix} 0 & 16 & 15 & 5 & 9 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & 6 \\ \infty & 9 & 7 & 0 & 4 \\ \infty & 4 & 3 & 5 & 0 \end{vmatrix} \otimes \begin{vmatrix} 0 & \infty & 15 & 5 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 4 \\ \infty & 5 & 3 & \infty & 0 \end{vmatrix} =$$

$$= \begin{vmatrix} 0 & 14 & 12 & 5 & 9 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & 6 \\ \infty & 8 & 7 & 0 & 4 \\ \infty & 4 & 3 & 5 & 0 \end{vmatrix}.$$

Заполненная матрица переходов имеет следующий вид:

$$Q_3 = \begin{vmatrix} - & 4 & 4 & 1 & 1 \\ - & - & - & - & - \\ - & 3 & - & 3 & 3 \\ - & 5 & 4 & - & 4 \\ - & 5 & 5 & 5 & - \end{vmatrix}.$$

Возведем в четвертую степень матрицу A, с использованием Шимбелловских операций:

$$A^4 = A \otimes C = \begin{vmatrix} 0 & \infty & 15 & 5 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 4 \\ \infty & 5 & 3 & \infty & 0 \end{vmatrix} \otimes \begin{vmatrix} 0 & 14 & 12 & 5 & 9 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & 6 \\ \infty & 8 & 7 & 0 & 4 \\ \infty & 4 & 3 & 5 & 0 \end{vmatrix} =$$

$$= \begin{vmatrix} 0 & 13 & 12 & 5 & 9 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & 6 \\ \infty & 8 & 7 & 0 & 4 \\ \infty & 4 & 3 & 5 & 0 \end{vmatrix}.$$

Заполненная матрица переходов имеет следующий вид:

$$Q_4 = \begin{vmatrix} - & 4 & 4 & 1 & 1 \\ - & - & - & - & - \\ - & 3 & - & 3 & 3 \\ - & 4 & 4 & - & 4 \\ - & 5 & 5 & 5 & - \end{vmatrix}.$$

Заметим, что число итераций в алгоритме Форда-Беллмана и в его матричном варианте алгоритме Шимбелла-Оттермана не может быть большим числа вершин графа, потому что лучший маршрут не может дважды заходить в одну и ту же вершину. Если бы возник такой цикл, то его можно отбросить и уменьшить длину маршрута. Другой критерий останова алгоритма состоит в том, что если некоторая степень матрицы смежностей оказывается равной степени с показателем, меньшим на единицу, то алгоритм следует прекратить, потому что дальнейшие расчеты уже не приведут к новым результатам. В данном примере $A^5 = A^4$. Расчет завершен. Длина кратчайшего маршрута из первой вершины во вторую равна элементу последней матрицы, стоящему в ее первой строке на втором месте. Эта длина равна 13 м совпадает с расчетом примера 7.

Алгоритм Шибелла-Оттермана является универсальным матричным алгоритмом оптимальной маршрутизации. Он работает во всех случаях, когда эта задача имеет смысл, в том числе и в тех, когда некоторые длины ветвей отрицательны, но не образуют циклов со всеми отрицательными ветвями.

Нахождение оптимальных маршрутов между всеми парами вершин с помощью алгоритма Шимбелла-Оттермана

Найдем оптимальный маршрут. По трем матрицам переходов получим:

$$1 \to 4 \to 5 \to 3 \to 2.$$

Этот ответ также совпал с ответом примера 7 .

3.3.3 Модификация алгоритма Шимбелла-Оттермана

Алгоритм Шимбелла-Оттермана находит сразу все наилучшие маршруты из каждой вершины в каждую другую вершину. Однако, когда

начальная и конечная вершины заданы, такая большая информация избыточна. В этом случае можно предложить другой, более экономный по числу вычислений, модифицированный алгоритм Шимбелла-Оттермана. Для его применения сначала следует выделить начальную вершину, конечную вершину и совокупность остальных промежуточных вершин орграфа. Для последних составляется квадратная матрица смежности $\mathbb{A}$, Для начальной вершины составляется матрица-строка $\mathbb{B}$ ветвей переходов из нее в промежуточные вершины. Для конечной вершины составляется матрица-столбец $\mathbb{C}$ ветвей, ведущих из промежуточных вершин в конечную вершину. Модифицированный алгоритм заключается в последовательном вычислении чисел, являющихся следующими произведениями перечисленных матриц:

$$p_1 = \mathbb{B} \otimes \mathbb{A} \otimes \mathbb{C}, \qquad p_3 = \mathbb{B} \otimes \mathbb{A}^2 \otimes \mathbb{C}, \dots, p_k = \mathbb{B} \otimes \mathbb{A}^k \otimes \mathbb{C}. \qquad (24)$$

Эти числа равны длинам оптимальных маршрутов, соответственно, из двух ветвей, трех ветвей и т. д. – из k ветвей. Алгоритм нахождения маршрутов не отличается от алгоритма в обычном (не модифицированном) алгоритме Шимбелла-Оттермана. Заметим, что для расчета по формулам (24) не нужно всякий раз вычислять степень матрицы. Вместо этого трудоемкого процесса, можно один раз умножить $\mathbb{A} \otimes \mathbb{C}$. При этом получится матрица-столбец, для вычисления которой затрачивается в k раз меньше элементарных вычислений, чем при умножении двух квадратных матриц. При этом k равно порядку квадратной матрицы (числу промежуточных вершин графа).

Пример 9. Найдем с помощью модифицированного алгоритма Шимбелла-Оттермана решение задачи примера 7.

$$\mathbb{A} = \begin{vmatrix} 0 & 2 & \infty \\ \infty & 0 & 4 \\ 3 & \infty & 0 \end{vmatrix}, \mathbb{B} = |15 \quad 5 \quad \infty|, \mathbb{C} = \begin{vmatrix} 1 \\ \infty. \\ 5 \end{vmatrix}$$

Сделаем расчеты:

$$\mathbb{D} = \mathbb{A} \otimes \mathbb{C} = \begin{vmatrix} 1 \\ 9 \\ 4 \end{vmatrix}, q_1 = \mathbb{B} \otimes \mathbb{A} \otimes \mathbb{C} = \mathbb{B} \otimes \mathbb{D} = 14, \qquad m_3 = 4, m_2 = 5.$$

$$q_2 = \mathbb{B} \otimes \mathbb{A}^2 \otimes \mathbb{C} = \mathbb{B} \otimes \mathbb{A} \otimes D = 13,\ m_1 = 4, m_2 = 5, m_3 = 3.$$

Оптимальный маршрут

$$1 \to 4 \to 5 \to 3 \to 2.$$

и его протяженность, равная 13, совпали с расчетами в примерах 7 и 8.

3.4 Алгоритм Минти

Этот алгоритм, разработанный за год до алгоритма Форда-Беллмана, является самым удобным, если отсутствует какая-либо информация о структуре графа. Алгоритм Минти находит оптимальные маршруты из всех вершин графа в заданную конечную вершину.

1-ый этап. Конечная вершина обозначается нулем, потому что длина пути из нее к себе равна нулю. Она объявляется известной вершиной. Остальные вершины объявлены как неизвестные.

2-ой этап. Находятся все вершины, связанные с хотя бы одной известной вершиной одной дугой. Множество этих вершин назовем «забором». Для каждой вершины забора найдем минимальную сумму длины ветви и значения переменной известной вершины, инцидентной этой ветви. Среди всех таких чисел находится наименьшее. Это наименьшее число и является длиной оптимального маршрута из указанной вершины в конечную вершину. Т.е. справедлив тезис «наименьшая переменная в заборе является сразу найденной». Докажем это утверждение. Если бы существовал иной лучший путь из этой вершины в известные, то он бы проходил через другую вершину забора, что заведомо хуже, потому что значение переменной той вершины большее. Найденную вершину относим к числу известных вершин.

3-ий этап. Проверим, все ли вершины стали известными. Если нет, то переходим ко второму этапу. Если переменные всех вершин стали известны, то эти переменные равны длинам оптимальных путей. Далее переходим к расчету оптимального маршрута из выделенной вершины в конечную вершину. Это можно сделать с помощью алгоритма, описанного в примере 4.

Пример 10. Найдем оптимальный маршрут в примере 7 методом Минти. На рис. 14.А показан первый этап алгоритма Минти. Здесь последняя вершина означена нулем.

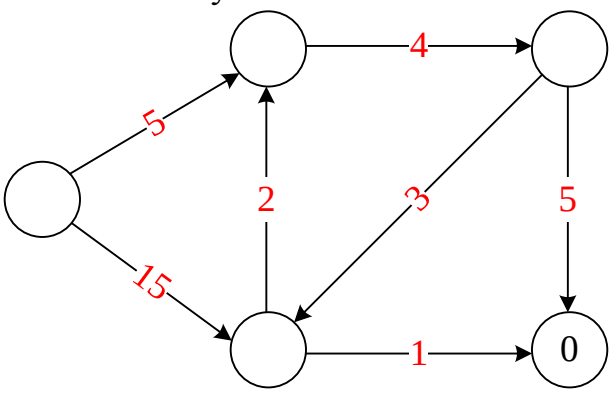


Рис.14.А. Первый этап алгоритма Минти

Первым «забором» служат две вершины, из которых можно попасть в конечную вершину. Меньшее значение ветви, исходящей из одной из них оцифровывается такая вершина.

На рисунках 14.Б, 14.В, 14.Ги 14.Д. показаны все шаги алгоритма Минти.

Вторым забором служит две вершины, из каждой из которых по одной единственной ветви можно попасть в одну из двух известных вершин. Минимальный путь из одной из них (в данном случае, не из начальной) оказался равным четырем (3+1=длина ветви плюс величина переменной известной вершины, инцидентной ветви).

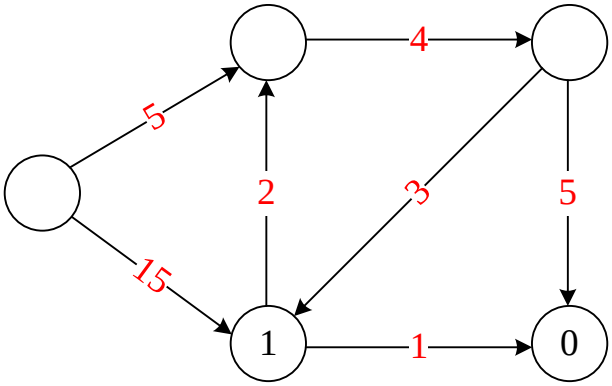


Рис.14.Б. Второй этап алгоритма Минти

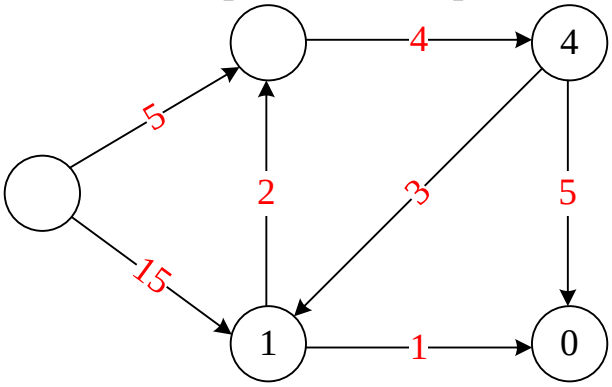


Рис.14.В. Третий этап алгоритма Минти

В третий «забор» входят обе вершины, пока оставшиеся неизвестными. Из них минимальное значение оказалось не у входной вершины.

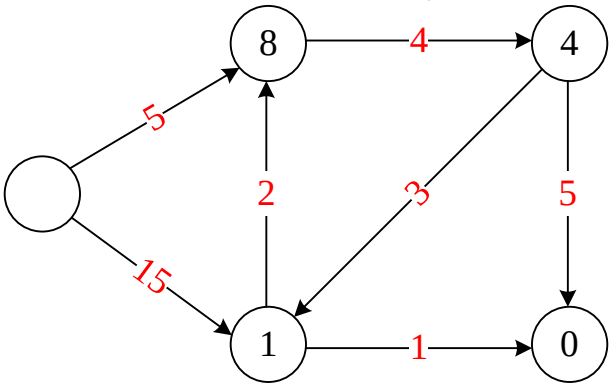


Рис.14.Г. Четвертый этап алгоритма Минти

В состав последнего забора входит первая вершина. Величина ее переменной находится по следующей формуле:

$$x = \min(5 + 8, 15 + 1) = 13.$$

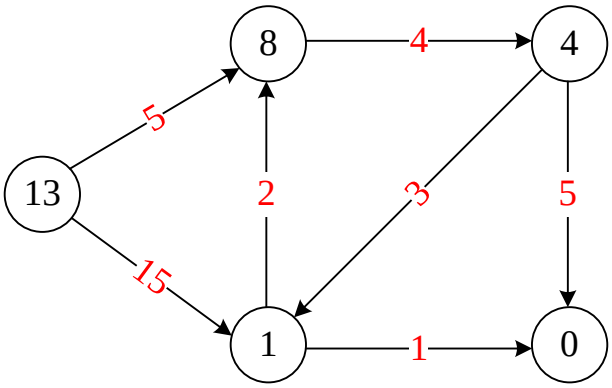


Рис.14.Д. Пятый этап алгоритма Минти

Из последнего рисунка 14.Д. можно сделать вывод о том, что времена оптимальных маршрутов, которые заканчиваются во второй вершине, а начинаются в первой, третьей, четвертой и пятой, соответственно равны 13, 1, 8 и 4.

Метод Минти работает в случаях положительных длин ветвей.

4. Методы сетевого планирования и управления проектами (PERT и МКП)

4.1 Пояснение к теме

PERT - Program (Project) Evaluation and Review Technique (сокращенно PERT) — техника оценки и анализа программ (проектов), которая используется при управлении проектами.

PERT был разработан главным образом для упрощения планирования на бумаге и составления графиков больших и сложных проектов. PERT предназначен для очень масштабных, единовременных, сложных, нерутинных проектов. Метод подразумевал наличие неопределённости, давая возможность разработать рабочий график проекта без точного знания деталей и необходимого времени для всех его составляющих. Этот метод был создан в конце 50-х годов XX века в военно-морских силах США для ускорения разработки лодочной баллистической ракеты «Поларис».

МКП – Critical part method – метод критического пути.

Метод МКП разработан практически одновременно независимо от предыдущего фирмой «Дюпон де Немур». Основное отличие методов состоит в том, что МКП не учитывает случайные колебания продолжительности работ.

С помощью этих методов руководитель проекта имеет возможность:

1. Заранее планировать работы по проекту и предвидеть возможные причины трудностей и задержек выполнения его в срок.
2. Планировать завершение каждой работы в необходимые сроки в точном соответствии с требуемой последовательностью выполнения заданий с целью оптимального осуществления проекта.
3. Координировать и контролировать выполнение работ при соблюдении календарного графика и завершения проекта точно в срок.

Сетевые методы тесно связаны с задачами распределения и использования ресурсов, сокращения сроков выполнения работ. Для этого решаются следующие задачи:

1. Устанавливаются последовательности и сроки использования ограниченных ресурсов в течение всего периода реализации проекта.
2. Проводится динамичное регулирование сроков начала и окончания каждой работы.
3. Осуществляется оптимальное распределение средств, выделенных на проект, для скорейшего его выполнения.
4. Выполняется анализ компромиссных соотношений между затратами ресурсов и сроками выполнения различных работ с учетом имеющихся резервов времени.

В первом параграфе настоящей главы была поставлена задача сетевого планирования комплекса работ. Исходными данными для нее служит перечень всех работ с указанием их длительности, а также логика следования работ. Например, при строительстве дома работа по строительству фундамента обязана предшествовать строительству крыши здания. Всю эту информацию можно представить на орграфе. На нем работы представлены ветвями, а вершинам соответствуют времена окончания работ. В отличие от задач оптимальной маршрутизации, здесь требуется находить не самые короткие пути, а самый длинный путь, ведущий из вершины – начала проекта к заданной вершине, в частности вершине – окончание всего проекта. Такой маршрут является самым напряженным. Срыв сроков проведения любой из работ этого критического маршрута приводит к затяжке осуществления всего проекта. Поэтому, при управлении производством работ руководитель вынужден, прежде всего, следить за выполнением в срок работ на критическом пути. Этого он может добиваться привлечением при необходимости дополнительных сил и средств для работ критического маршрута. Как уже было отмечено выше, задачи на максимум легко сводятся к задачам на минимум. По отношению ко всем рассмотренным выше алгоритмам оптимальной маршрутизации, это означает только то, что везде знак min следует поменять на знак max. Но, прежде чем приступать к выбору алгоритма, следует уяснить специфику орграфа в сетевой задаче планирования. Во-первых, длительность любой работы является положительным числом. Во-вторых, если первая работа предшествует второй, а вторая работа предшествует третьей, то первая работа предшествует третьей. Это значит, что в орграфе отсутствуют циклы. Таким образом, орграф задачи сетевого планирования является ациклическим с положительными длинами ветвей, соответствующих временам проведения работ. Оказывается, что для расчета переменных вершин может быть использован весьма простой алгоритм последовательного нахождения переменных вершин.

4.2 Теоретические основы алгоритма расчета самого длинного маршрута между двумя вершинами ациклического орграфа

Пусть указаны две вершины орграфа, одна из которых является исходной для маршрута, а вторая его конечной вершиной. Требуется найти самый длинный маршрут на ациклическом орграфе, между первой и второй вершинами. Сначала осуществим инвариантное погружение данной задачи в семейство задач, где ищутся самые длинные маршруты от первой вершины во все остальные вершины орграфа. Докажем, что среди вершин ациклического графа найдется хотя бы одна, в которую не ведет ни одна ветвь из остальных вершин. Допустив обратное предположение, выберем произвольную вершину и найдем другую, из которой идет ветвь в выбранную вершину. Далее найдем и третью вершину, из которой идет ветвь во вторую и т.д. В силу конечности числа вершин этот процесс должен завершиться вторичным его попаданием в уже пройденную вершину. Это означает, что найден цикл. В ациклическом графе это невозможно. Поэтому, существует вершина, в которую не ведет ни одна ветвь.

4.3. Алгоритм расчета самого длинного пути от заданной вершины А к другой вершине орграфа

1-ый этап. Будем полагать, что маршрут от первой вершины к ней же имеет длину ноль. Оцифруем нулем первую вершину.

2-ой этап. Из оставшихся вершин перебором находим ту, в которую не ведет ни одна ветвь из неозначенных вершин. То, что такая вершина существует, доказано в предыдущем пункте 4.2. Найдем максимальную величину сумм, ведущих в найденную вершину от означенных вершин и значений этих вершин.

3-ий этап. Проверим, означены ли все вершины. Если да, то алгоритм заканчивает свою работу. Если нет, то переходим к пункту 2.

Пример 11. На рис.15.А представлен ациклический орграф. Требуется найти длину самого протяженного маршрута из самой левой вершины на этом рисунке до каждой из оставшихся вершин орграфа.

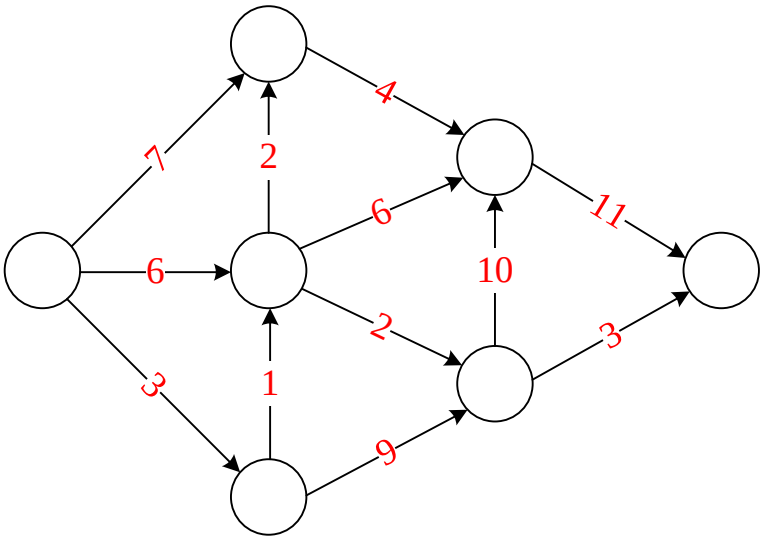


Рис.15 А. Орграф примера 11

Решение

Означим нулем самую левую вершину на рис.15 А.

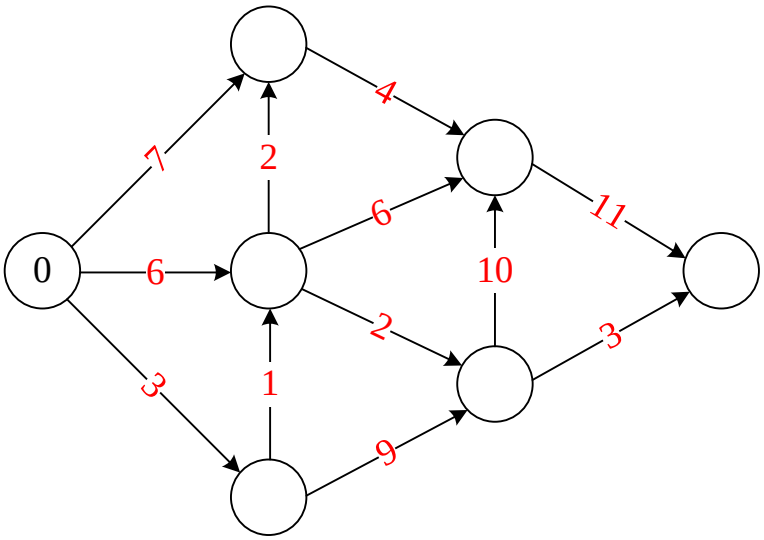


Рис.15 Б. Орграф примера 11 после первого шага алгоритма

К самой нижней вершине на рис.15 Б не подходят ветви из других неозначенных пока вершин. Ее означим длиной подходящей к ней ветви: 3.

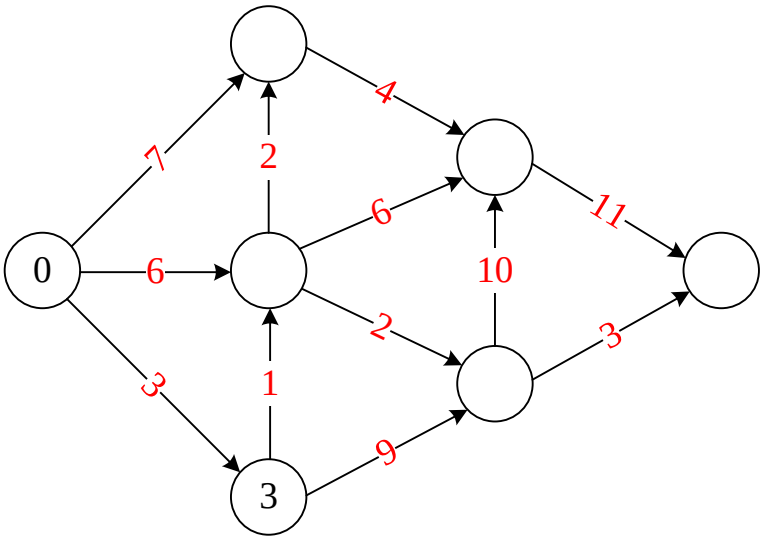


Рис.15 В. Орграф примера 11 после второго шага алгоритма

Снова находим неозначенную вершину на рис. 15.Б. Эта та, к которой идет ветвь длиной единица от только что означенной вершины. Находим значение переменной этой вершины:

$$\max(6 + 0{,}1 + 3) = 6.$$

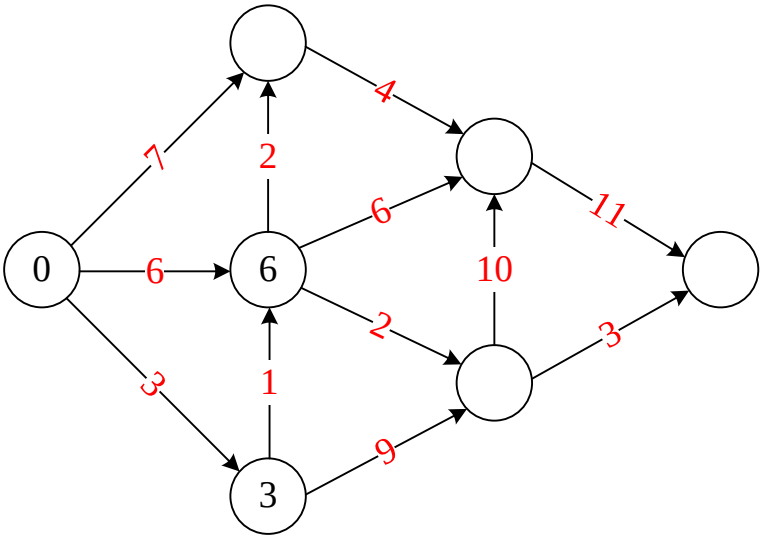


Рис.15 Г. Орграф примера 11 после третьего шага алгоритма

Снова находим неозначенную вершину на рис. 15.В.

Эта та, к которой идет ветвь длиной единица от только что означенной вершины. Находим значение переменной этой вершины:

$$\max(7 + 0{,}2 + 6) = 8.$$

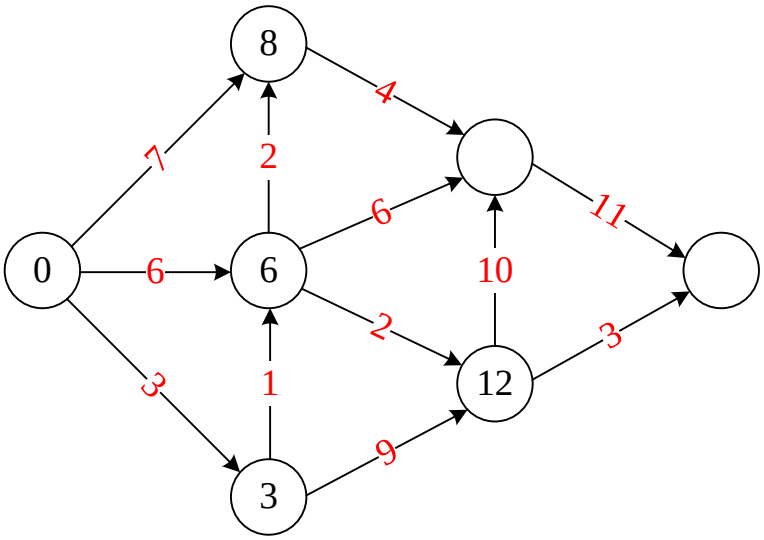


Рис.15 Г. Орграф примера 11 после четвертого шага алгоритма

Снова находим неозначенную вершину на рис. 15.Г. Это нижняя вершина из неозначенных вершин на этом рисунке. Находим значение переменной этой вершины:

$$\max(9+3, 2+6) = 12.$$

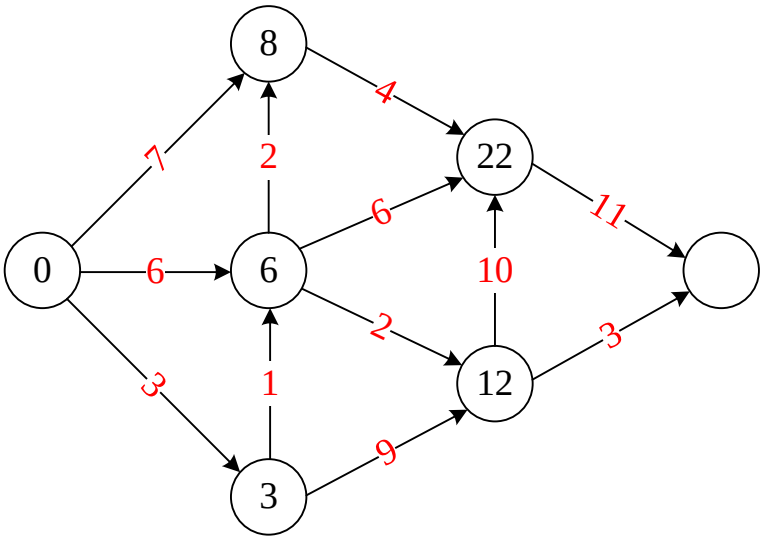


Рис.15 Д. Орграф примера 11 после пятого шага алгоритма

Снова находим неозначенную вершину на рис. 15.Д. Это верхняя из неозначенных вершин на этом рисунке. Находим значение переменной этой вершины:

$$\max(10+12, 6+6, 4+8) = 22.$$

Находим неозначенную вершину на рис. 15.Д. Это последняя из неозначенных вершин на этом рисунке. Находим значение переменной этой вершины:

$$\max(11+22, 6+6, 3+12) = 33.$$

Окончательная оцифровка вершин исходного орграфа изображена на рис.15 Е.

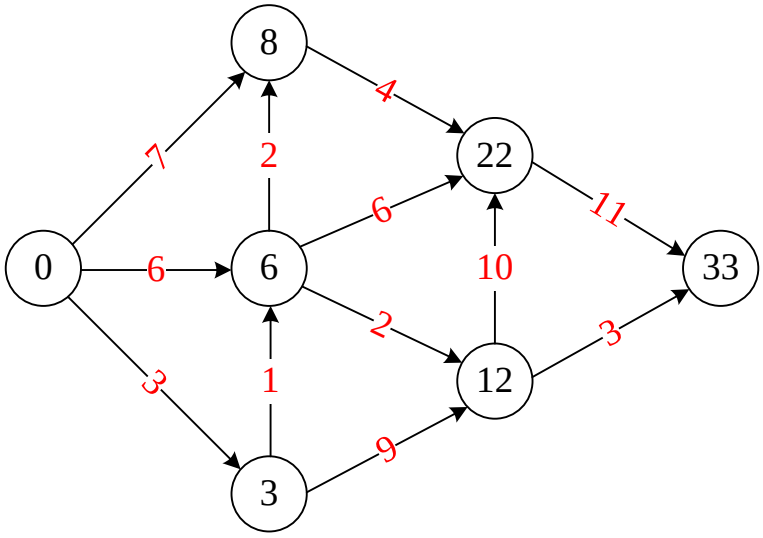


Рис.15 Е. Орграф примера 11. Результаты расчета

Расчет закончен.

4.3 Стандартная методика расчетов по алгоритму PERT

На практике для получения возможности отслеживать проведение работ при реализации сложных проектов применяется метод сетевого планирования и управления (PERT). Самой популярной частью PERT является Метод критического пути, опирающийся на построение сетевого графика (сетевой диаграммы PERT). Такой подход был разработан в 1958 году консалтинговой фирмой «Буз, Ален и Гамильтон» совместно с корпорацией «Локхид» по заказу Подразделения специальных проектов ВМС США в составе Министерства Обороны США для проекта создания ракетной системы «Поларис» (Polaris). Проект «Поларис» был ответом на кризис, наступивший после запуска Советским Союзом первого космического спутника.

Модель узел-событие

В сетевой диаграмме принято под ветвями понимать работы, а вершинам соответствуют события, представляющие собой начало ряда работ и окончание некоторых других работ. События нумеруются, начиная от 1 – начального и заканчивая конечным - завершением работ. Характеристикой работы является время t_{ij} – где i – номер вершины, от которой начинается данная работа, а j – номер вершины, в которой она заканчивается.

Для полной характеристики проведения проекта во времени календарных дат используются четыре показателя:

$T_e(k)$ –самая ранняя дата начала k-го события;

$T_l(k)$- самая поздняя дата окончания k-го события;

S(k) – резерв времени k-го события;

Критический путь от начала до окончания проекта.

Часто при проведении таких расчетов за единицу времени принимается 1 неделя.

Пример 12. Рассмотрим пример сетевого представления проекта по разработке информационной системы, орграф которого приведен на рис.16.

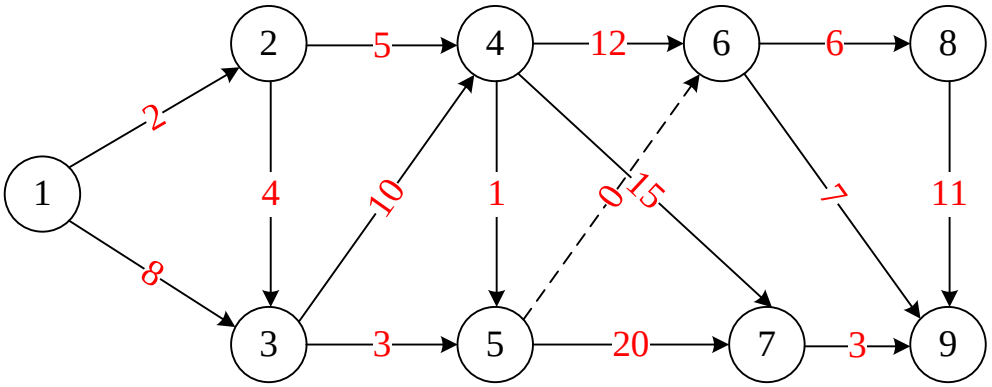


Рис.16. Пример сетевого плана

Порядок работ и соответствующие им операции приведены в табл.3.

Таблица 3

Перечень работ сетевого плана разработки информационной системы

Номера работ	Выполняемые операции
1-2	Постановка задач, решаемых данным программным продуктом
1-3	Разработка технического задания проекта
2-3	Технико-экономическое обоснование проекта
2-4	Согласование основных принципов функционирования системы
3-4	Создание необходимых входных форм
3-5	Создание необходимых выходных форм
4-5	Выбор инструмента для реализации
4-6	Программирование основных модулей
4-7	Программирование дополнительных модулей
5-6	Условная связь, отмечающая, что работы, предшествующие событию 5 не должны закончиться ранее работы 5-6
5-7	Разработка необходимой пояснительной документации
6-8	Связь всех компонентов системы воедино
6-9	Отладка системы
7-9	Тестирование системы
8-9	Создание защиты от несанкционированного доступа к системе

1 этап – расчет самой ранней даты начала каждого события.

2 этап - расчет самой поздней даты окончания каждого события.

3 этап – расчет резервов времени.

4 этап – определение критического пути.

1 этап. По определению самая ранняя дата наступления начального события равна нулю. Самая ранняя дата соответствует максимальному времени из всех полученных, т.к. для того, чтобы наступило соответствующее событие, должны закончиться все работы ему предшествующие. Расчет самой ранней даты начала каждого $k-ого$ события ведется от начального события с помощью алгоритма, описанного в пункте 4.3. По определению самая поздняя дата окончания завещающего события равна самой ранней дате его начала. Самая поздняя дата окончания k-го события соответствует разности времени самой поздней даты окончания конечного события и максимальному времени окончания всех событий, последующих k-му событию, которое можно определить тем же самым алгоритмом.

В связи с тем, что для реальных сетевых планов характерно наличие сотен, а иногда и тысяч вершин и ветвей, используют табличную форму расчета. Для этого в таблице строкам и столбцам соответствуют номера событий (такая таблица является матрицей смежностей орграфа). На пересечении строки и столбца проставляется соответствующее время работы, соединяющей эти два события. После занесения всех времен производят проверку отсутствия циклов в сети. Для этого все занесенные в таблицу значения времени должны располагаться выше главной диагонали.

Проведем расчет примера рис.16 табличным способом табл.4. Заполним таблицу 3. Первое событие соединено со вторым работой t_{12} со временем продолжительностью 2 единицы, следовательно, на пересечении строки с номером 1 и столбца с номером 2 ставится это время (2). Аналогично на пересечении первой строки и третьего столбца заносится время t_{13} равное 8 единицам. И так далее. При проверке все занесенные времена оказались выше главной диагонали таблицы, следовательно, рассматриваемая сеть не содержит циклов. На первом этапе рассчитывается самая ранняя дата начала каждого события. Для этого необходимо сложить время $\mathbf{T_e}$, соответствующее рассматриваемому событию (в строке) и время (в столбце) работы, соединяющей два события. Из полученных сумм выбирают наибольшую и заносят в первый столбец таблицы. $\mathbf{T_e\ (1)}$ по определению равно нулю. Для нахождения $\mathbf{T_e\ (2)}$ необходимо сложить уже определенное значение $T_e(1)$ (в строке) с временем равным 2, находящимся во втором столбце. Следовательно, $T_e(2)=0+2=2$. Для третьего события $T_e(3)$ будем определять как сумму $T_e(1)$ и числа 8, стоящего в той же строке и третьем столбце и $T_e(2)$ и числа 4, расположенного во второй строке и третьем столбце. Поэтому $T_e(3)=0+8=8$ и $T_e(3)=2+4=6$. Большее значение 8 заносим в таблицу. Для четвертого события $\mathbf{T_e\ (4)}$ будет рассчитано как сумма $T_e(2) + 5 = 2 + 5= 7$ и $T_e(3)+10= =8+10=18$. Максимальное значение 18 записано в таблицу. После того как определены значения $\mathbf{T_e}$ для всех событий, их переписывают в соответствующую строку таблицы и начинают расчет самой поздней даты окончания каждого события. Для этого заносят значение самой поздней даты

окончания конечного события равной самой ранней дате его начала. Далее проводят расчет путем вычитания из уже определенного значения T_l (в столбце) значения времени в соответствующей строке. Из нескольких разностей выбирается наименьшая.

$T_l(8)$=T_l(9)-11=47-11=36, других разностей нет, следовательно, в таблицу заносим число 36. Аналогично для **$T_l(7)$=**T_l(9)-3=47-3=44 в таблицу заносится 44. **$T_l(6)$=**T_l(9)-7=47-7=40 и **$T_l(6)$=**T_l(8)-6=36-6=30, выбирается минимальное значение равное 30 и вносится в таблицу, и т.д.

После нахождения самой поздней даты окончания каждого события путем вычитания $T_l(k)=T_e(k)$ определяем резервы времени и также заносим в таблицу. Далее определяется критический путь.

Таблица 4

Расчет характеристик проекта методом PERT

T_e	i\j	1	2	3	4	5	6	7	8	9	
0	**1**		2	8							**1**
2	**2**			4	5						**2**
8	**3**				10	3					**3**
18	**4**					1	12	15			**4**
19	**5**						0	20			**5**
30	**6**								6	7	**6**
39	**7**									3	**7**
36	**8**									11	**8**
47	**9**										**9**
	T_l	**0**	**4**	**8**	**18**	**24**	**30**	**44**	**36**	**47**	
	T_e	**0**	**2**	**8**	**18**	**19**	**30**	**39**	**36**	**47**	
	S	**0**	**2**	**0**	**0**	**5**	**0**	**5**	**0**	**0**	

На рис.17 показан критический путь и остальные характеристики сети.

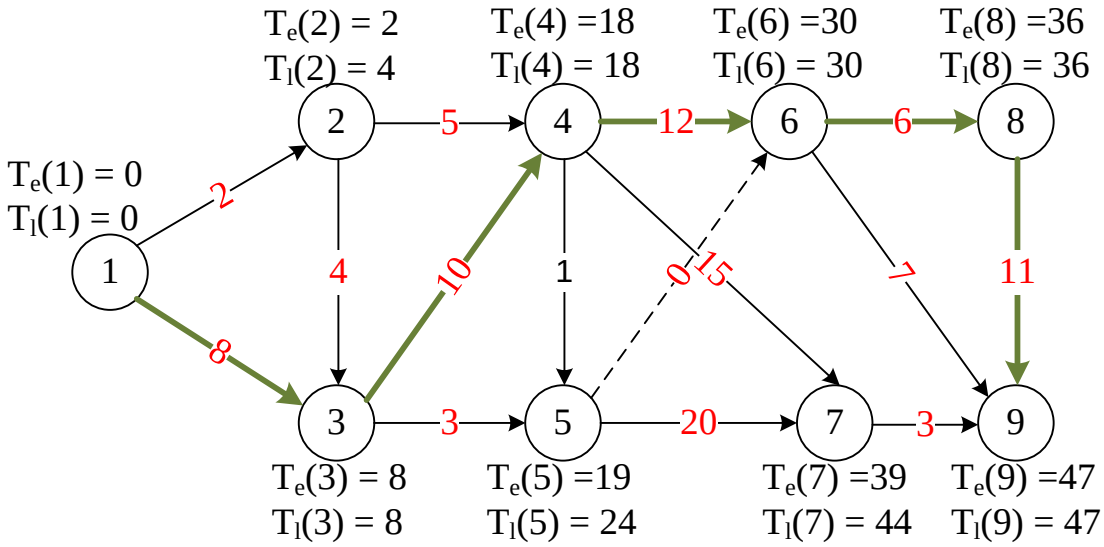


Рис.17. Результаты расчетов сетевого плана

Вычисления табличным способом удобно проводить с использованием Excel.

Таким образом, видно, что в критический путь вошли работы, требующие наибольшего ресурсного обеспечения и высокой квалификации исполнителей.

Кроме определения соответствующих характеристик для событий рассчитывают резервы времени и для работ: суммарный, свободный, независимый и гарантированный.

Суммарный резерв времени TF_{ij} для работы (i,j) представляет собой максимальную продолжительность задержки работы (i,j), не вызывающую задержки в осуществлении всего проекта. Этот показатель рассчитывается следующим образом:

$TF_{ij} = LS_{ij} - ES_{ij}$ или $TF_{ij} = LF_{ij} - EF_{ij}$,

где LS_{ij} – наиболее поздний допустимый срок начала (i,j) работы,

ES_{ij} – наиболее ранний возможный срок начала (i,j) работы,

LF_{ij} - наиболее поздний допустимый срок окончания (i,j) работы,

EF_{ij} - наиболее ранний возможный срок окончания (i,j) работы.

Следует учитывать, что $ES_{ij} = Te_i$, $LF_{ij} = Tl_j$.

Работа, имеющая нулевой суммарный резерв времени, находится на критическом пути (табл.5).

Определение суммарного резерва времени актуально для определения критического пути и при планировании срока завершения всего проекта, но не может быть использовано при планировании сроков отдельных работ.

Свободный резерв времени FF_{ij} является показателем максимальной задержки работы (i,j), не влияющей на начало последующих работ. Как и для

предыдущей оценки предполагается, что все предшествующие работы завершаются как можно раньше. Свободный резерв времени отличается от суммарного тем, что он измеряет имеющееся время, не влияющее на задержку последующих работ. Рассчитывается этот резерв времени рассчитывается следующим образом:

$$FF_{ij} = Te_j - EF_{ij},$$

где Te_j – наиболее ранний срок начала j события.

Свободный резерв времени каждой работы не может превышать суммарного резерва (табл.5).

Независимый резерв времени IF_{ij} является удобным показателем свободы планирования сроков. Независимый резерв времени работы (i,j) представляет собой максимальную продолжительность задержки работы (i,j) без задержки последующих работ, если все предшествующие работы заканчиваются в поздние сроки. Это показатель имеющегося времени, если при выполнении предшествующих работ возникнут наихудшие из возможных условия. Также он показывает возможную степень нарушения связи между работами проекта. Рассчитать этот показатель можно следующим образом:

$$IF_{ij} = max\begin{Bmatrix} 0, \\ Te_j - (Tl_i + d_{ij}) \end{Bmatrix},$$

где d_{ij} - продолжительность работы (i,j),

Tl_j – наиболее поздний срок окончания j события.

Нулевой член введен для того, чтобы приравнять нулю отрицательные результаты (табл.5).

Гарантированный резерв времени SF_{ij} – максимально возможная задержка работы, не влияющая на окончательный срок завершения проекта, если предшествующие работы выполнялись с запаздыванием. Этот показатель является наиболее удобным при планировании сроков выполнения определенной работы. Он допускает задержку только последующих работ, а не всего проекта (табл.5).

$SF_{ij} = LF_{ij} - (Tl_i + d_{ij})$ или $SF_{ij} = Tl_j - (Tl_i + d_{ij})$,

где Tl_i – наиболее поздний срок окончания i события.

Сведем все расчетные показатели для работ примера рис.16 в таблицу.

Таблица 5

Сроки начала, окончания работ и резервы времени

Работа	Продолжи-тельность работы d_{ij}	Наиболее ранний возможный срок		Наиболее поздний допустимый срок		Резервы времени			
		начала $ES_{ij}=Te_{ij}$	окончания EF_{ij}	начала LS_{ij}	окончания $LF_{ij}=Tl_j$	Суммарный TFij	Свободный FFij	Независимый IFij	Гарантиро-ванный SFij
1-2	2	0	2	0	4	2	0	0	2

1-3	8	0	8	0	8	0	0	0	0
2-3	4	2	6	4	8	2	2	0	0
2-4	5	2	7	13	18	11	11	9	9
3-4	10	8	18	8	18	0	0	0	0
3-5	3	8	11	21	24	13	8	8	13
4-5	1	18	19	23	24	5	0	0	5
4-6	12	18	30	18	30	0	0	0	0
4-7	15	18	33	29	44	11	6	6	11
5-6	0	19	19	30	30	11	11	6	6
5-7	20	19	39	24	44	2	0	0	0
6-8	6	30	36	30	36	0	0	0	0
6-9	7	30	37	40	47	7	10	10	10
7-9	3	39	42	44	47	3	5	0	0
8-9	11	36	47	36	47	0	0	0	0

Полученные данные позволяют:

1. По нулевому суммарному резерву времени определять работы, находящиеся на критическом пути, для которых необходимо точно отслеживать срок исполнения.

2. Остальные показатели резервов времени используют для более точного определения сроков выполнения работ в тех случаях, когда работники или оборудование, необходимое для выполнения какой-то работы, должны распределяться с учетом потребности в них на других работах. Различные показатели резервов времени позволяют распределять имеющиеся ресурсы для каждой работы. При наличии резерва времени появляется некоторая свобода распределения ресурсов.

4.4. Расчет времен в сетевых планах

В рассмотренном примере времена проведения работ t_{ij} были заданы. На практике получение этих значений может быть связано с определенными трудностями, особенно, если рассматривается сетевой план проекта, не имеющего аналогов. Для нахождения продолжительности проведения соответствующих работ применяют три оценки времени: пессимистическую, оптимистическую и наиболее вероятную. Получив эти оценки любым существующим методом (использующим аналогию с работами в других проектах, экспертным, методом прямого расчета времени и т.д.) можно воспользоваться следующими формулами:

$$t_{ij} = \frac{n_o t_o + n_в t_в + n_п t_п}{n_о + n_в + n_п}, \quad (25)$$

где t_o - оптимистическое время,

$t_в$ – наиболее вероятное время,

$t_п$ - пессимистическое время,

$n_{\text{о}}, n_{\text{в}}, n_{\text{п}}$ – веса, соответствующие оценкам времени: оптимистической, наиболее вероятной и пессимистической.

Часто весовые значения выбирают следующим образом:

$n_{\text{о}} = 1, n_{\text{в}} = 4, n_{\text{п}} = 1$, тогда формула (25) будет иметь следующий вид:

$$t_{ij} = \frac{t_o + 4t_{\text{в}} + t_{\text{п}}}{6}, \quad (26)$$

В случае трудности определения наиболее вероятного времени реализации работы применяют формулу:

$$t_{ij} = \frac{3t_o + 2t_{\text{п}}}{5}. \quad (27)$$

Для определения сложности сетевого плана используется «коэффициент сложности графика», который рассчитывается по формуле:

$$\text{К} = \frac{\text{Количество работ}}{\text{Количество событий}}.$$

При значении этого коэффициента от 1 до 1,5 сетевой план полагается простым, для плана средней сложности характерен коэффициент от 1,15 до 2, в случае, если этот коэффициент более 2,1 сетевой график – сложный.

5. Потоки в сетях

5.1 Пояснение к теме

Представленные в первом параграфе настоящей главы классические транспортная задача и задача о назначениях были исторически первыми из числа задач линейного программирования. Они появились еще до изобретения ЭВМ, состоявшегося в 1944 году. Примеры задач, которые были решены здесь первыми, были невысокой размерности. Поэтому, общие схемы их решения были матричными. Это означает использование специальных таблиц и составления алгоритмов преобразования таких таблиц для получения оптимального решения. Имеется аналогия с тем, как при решении систем обыкновенных линейных уравнений сначала использовались матричные методы, которые только много позже были несколько потеснены методами теории графов. Об этом написано в первой главе раздела
Теории А1 Теории систем и системного анализа. Практическая необходимость использования сетевых методов связана со значительной их вычислительной эффективностью (высокой скоростью работы программ и экономным объемом использования памяти ЭВМ), а также целым рядом теоретических результатов и обобщений. Например, с помощью сетевых методов могут быть решены многочисленные комбинаторные задачи. Центральным результатом в теории классической задачи о максимальном потоке в двухполюсной сети является алгоритм Форда и Фалкерсона численного решения этой проблемы.

5.2 Поток в двухполюсной сети

Задача о максимальном потоке в двухполюсной сети исторически возникла из потребностей железнодорожного транспорта в США. Ветки железной дороги имеют ограниченную пропускную способность (число пар поездов в сутки). Можно говорить о максимальной пропускной способности участка сложной железнодорожной сети между двумя ее пунктами. Такой участок состоит из целого ряда железнодорожных веток и промежуточных станций. При этом для каждой из веток задана ее пропускная способность.

Перейдя к математической модели, будем говорить, что задан связный орграф, каждой из ветвей которого q сопоставлено положительное число $u(q)$, которое называется пропускной способностью этой ветви. В орграфе имеются две выделенные вершины. К первой из них не подходит ни одна ветвь, но имеется хотя бы одна выходящая ветвь. Эта вершина называется истоком и обозначается как s. Из второй вершины не выходит ни одна ветвь, но существуют хотя бы одна входящая ветвь. Эта вершина называется стоком и обозначается как t. Все остальные вершины орграфа называются промежуточными (или внутренними). В таком случае говорят, что задана транспортная двухполюсная сеть. На рис. 18 изображена двухполюсная сеть.

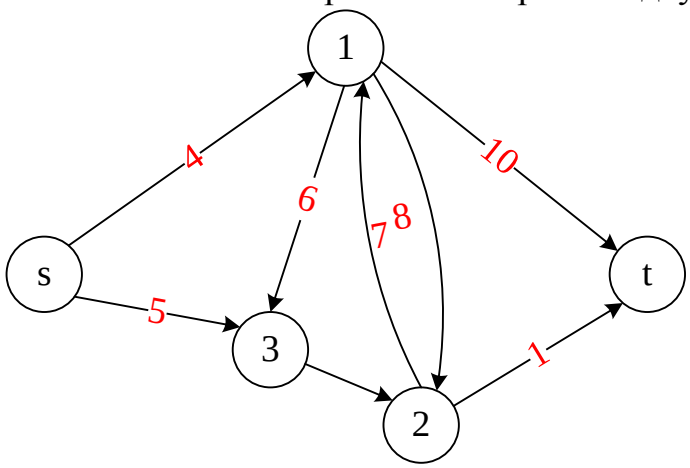


Рис. 18. Пример двухполюсной сети

В этой сети присутствует исток s, сток t и три промежуточные вершины. Пропускные способности вервей заданы числами, расположенными на рисунке рядом с ветвями.

Допустимым потоком (в дальнейшем, просто, потоком) в транспортной сети называется неотрицательная функция $f(q)$, определенная на всех ветвях орграфа и имеющая следующие два свойства:

1. $0 \le f(q) \le u(q)$;
2. для каждой промежуточной вершины сумма потоков всех подходящих ветвей равна сумме потоков всех отходящих ветвей.

Мощностью потока всей сети будем называть сумму потоков ветвей, выходящих из истока. Заметим, что сумма входящих потоков в сток равна мощности потока (см.задачу 14).

Задача Форда и Фалкерсона имеет смысл, потому что допустимые потоки существуют. Простейшим допустимым потоком является нулевой

поток, при котором потоки ветвей равны нулю. Так как сеть считается связной, то существует цепь, ведущая от источника к стоку. Выбрав значения всех потоков ветвей на этой цепи, равной минимальной пропускной способностью ветвей цепи, а для всех остальных ветвей приняв потоки равными нулю, получим еще один допустимый поток. С другой стороны, мощность потока сети не может превышать сумму потоков ветвей, отходящих от истока. По известной теореме Математического анализа о том, что непрерывная функция (в данном случае допустимый поток) достигает своего максимума при ограничениях на переменные (потоки ветвей) типа неравенство, получаем теорему существования максимального потока. Осталось только решить эту задачу – в данном случае найти алгоритм ее численного решения.

5.3 Пояснение к теме. Полный поток, не являющийся максимальным

Нахождение алгоритма решения задачи далось Форду и Фалкерсону не сразу.

Попытка прямолинейного решения не приводит к успеху. Тем не менее, рассмотрим простейший подход к поставленной проблеме. Говорят, что ветвь насыщена потоком, если дуговой поток ветви совпадает с пропускной способностью ветви. Заметим, что если существует цепь, связывающая исток со стоком и не имеющая насыщенных ветвей, то поток не является максимальным. Действительно, в этом случае можно немного увеличить на одну и ту же величину потоки всех ветвей этой цепи и, таким образом увеличить потов в сети. Будем говорить, что поток является полным, если любая цепь, соединяющая исток и сток имеет хотя бы одну насыщенную ветвь. Из приведенных рассуждений следует, что максимальный поток является полным. Казалось бы, что полный поток и является претендентом на то, чтобы быть максимальным. Но, оказывается, что не всегда полный поток максимален.

Пример 13. На рис.19А и 19 Б. приведены два потока. При этом около каждой ветви стоит число, равное ее пропускной способности, а в скобках поток ветви. Несложно проверить, что поток удовлетворяет обязательным свойствам потока о равенстве сумм потоков подходящих и отходящих от промежуточных вершин. Кроме того, потоки ветвей не превосходят пропускной способностей этих ветвей. Покажем, что первый поток является полным. Из вершины s в вершину t ведет четыре пути:

1) $s-1-t$ - насыщена ветвь 1-t,
2) $s-1-2-t$ - насыщена ветвь 1-2,

3) $s-2-t$ - насыщена ветвь $s-2$,

4) $s-2-1-t$ - насыщена ветвь $s-2$.

Потоки всех цепей, ведущих из вершины s в вершину t, имеют насыщенные ветви. Поток полон. Его мощность равна сумме потоков, отходящих от истока. Эта сумма равна 1+2=3. Но найденный поток не оптимален, потому что поток, представленный на рис. 19Б, имеет мощность, равную четырем. Его мощность больше мощности первого потока.

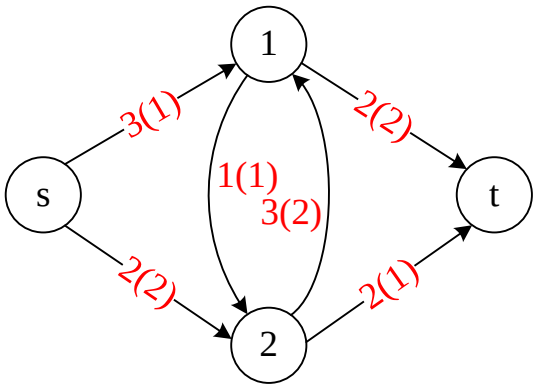


Рис. 19 А. Полный поток равен трем

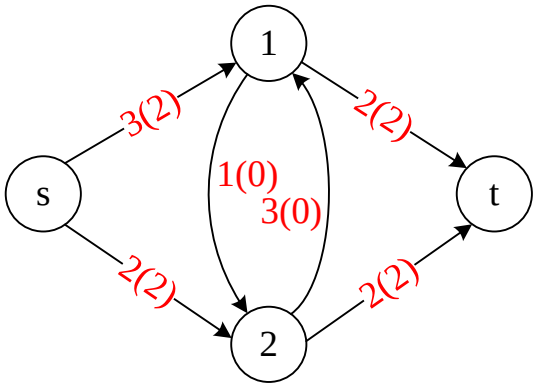


Рис.19 Б. Полный поток равен четырем

Как объясняли Форд и Фалкерсон в своей книге, проблему нахождения максимального потока удалось сдвинуть с места, только после введения понятия разреза в графе.

5.4 Способы увеличения потоков. Разрезы. Теорема Форда и Фалкерсона

Простейший способ увеличения уже имеющегося (например, нулевого) потока состоит в нахождении пути из истока в сток, составленного из ненасыщенных исходным потоком направленных ветвей. В этом случае можно увеличить исходный поток в каждой ветви цепи на величину минимальной разности между пропускной способностью каждой ветви цепи и ее потоком. Такой способ увеличения потока, примененный несколько раз, заведомо приведет к нахождению полного, но, к сожалению, не всегда максимального потока.

Замечание 4. Одна из идей Форда и Фалкерсона заключатся в расширении простейшего способа увеличения потока. Для увеличения потока следует найти цепь, ведущую из истока в сток, состоящую, в общем случае, из ветвей разного направления. При этом, ветви цепи, сонаправленные первой из них, ведущей от истока, будем называть направленными прямо. Они не должны быть насыщенными исходным потоком. Остальные ветви обратного направления должны иметь ненулевой исходный поток. Если удастся найти такую цепь, то следует вычислить минимальную разность между пропускными способностями прямых ветвей и исходным потоком на них, а также минимальный поток обратных ветвей. Далее, можно выбрать ε - наименьшее из этих двух последних чисел. Можно увеличить поток каждой прямой ветви на ε и уменьшить на ε каждый поток обратной ветви. Легко убедиться, что получится новый поток, который больше прежнего на величину ε.

Форд и Фалкерсон доказали, что с помощью этого способа можно найти максимальный поток, если все ограничения на пропускные способности ветвей являются целыми числами. Для этого они впервые ввели в рассмотрение понятие разреза на двухполюсном орграфе.

Определение 1. Разрезом на двухполюсном орграфе называется разбиение множества его вершин на два подмножества: $\mathbb{A}$, которое содержит исток, и множество $\mathbb{B}$, которое содержит сток. Мощностью разреза называется сумма пропускных способностей ветвей, ведущих из вершин множества $\mathbb{A}$ в вершины множества $\mathbb{B}$. Разрез, имеющий наименьшую мощность, называется минимальным.

Лемма 1. Мощность любого разреза не меньше мощности любого потока.

Доказательство

Рассмотрим произвольный разрез в сети с заданным потоком. Рассмотрим алгебраическую сумму потоков ветвей, которые начинаются или заканчиваются в вершинах множества разреза $\mathbb{A}$ за исключением истока. Эта сумма равна нулю, потому что она составлена из сумм потоков в промежуточных узлах. С другой стороны, она равна сумме трех слагаемых. Первое из них - сумма потоков ветвей, начало и конец которых находится в множестве $\mathbb{A}$ за исключением истока. Это слагаемое равно нулю, потому что поток каждой ветви входит в него дважды: один раз со знаком плюс как подходящий в некоторую вершину и один раз со знаком минус, как выходящий из другой вершины. Второе слагаемое равно сумме потоков ветвей с началом в истоке. Эта сумма является мощностью потока. Третье слагаемое отрицательно и равно со знаком минус сумме потоков ветвей,

имеющих начало в множестве $\mathbb{B}$ разреза и конец в его множестве $\mathbb{A}$. Из проведенных рассуждений следует, что поток равен разности мощности разреза и потока из множества $\mathbb{B}$ разреза в его множестве $\mathbb{A}$. Поэтому величина любого потока не превосходит мощности любого разреза.

Из Леммы следует вывод о том, что наименьшая мощность разреза не меньше мощности максимального потока. Фундаментальным результатом теории Форда и Фалкерсона является утверждение о равенстве этих величин.

Теорема Форда и Фалкерсона. Мощность максимального потока равна мощности минимального разреза (разреза с наименьшей мощностью).

Доказательство

Для доказательства достаточно по максимальному потоку найти **специальный** разрез, мощность которого равна мощности потока. Действительно, если такой разрез будет найден, то из Леммы 1 следует, что мощность любого разреза, не уступающая мощности любого потока, оказалась, таким образом, больше, либо равной мощности найденного разреза. Следовательно, найденный разрез минимальный, а поток через него максимальный.

Множество $\mathbb{A}$ искомого разреза строится индуктивно. Сначала в него включается исток. Затем, на каждом шаге алгоритма в множество $\mathbb{A}$ включаются те вершины, в которые можно попасть по ненасыщенным ветвям орграфа из уже найденных вершин множества $\mathbb{A}$. Также, в множество $\mathbb{A}$ включаются и те вершины, из которых ведут ненулевые потоки в вершины множества $\mathbb{A}$. Таким образом, множество $\mathbb{A}$ состоит из таких вершин, любая цепь из которых в силу Замечания 4, может привести к увеличению потока. Но поток предполагается максимальным потоком, а цепь Замечания 1, ведущая из истока в сток обязательно увеличивает поток. Следовательно, не существует цепи ведущей из истока в сток. Поэтому существует не пустое множество $\mathbb{B}$. Множества $\mathbb{A}$ и $\mathbb{B}$ определяют разрез. По построению множества $\mathbb{A}$ в него могут вести только такие ветви из множества $\mathbb{B}$, потоки которых равны нулю. Следовательно, мощность найденного разреза равна мощности максимального потока. Теорема доказана.

Замечание 5. Свойства специального разреза максимального потока.

1. Любая ветвь из множества $\mathbb{A}$ этого разреза, ведущая в множество $\mathbb{B}$, разреза, насыщена.
2. Любая ветвь из множества $\mathbb{B}$ этого разреза, ведущая в его множество $\mathbb{A}$, имеет нулевой поток.

3. Если все ограничения на пропускные способности ветвей являются натуральными числами, то максимальный поток является натуральным числом.

Замечание 6. (Понятие сети приращения потока). Пусть на сети задан некоторый поток f. Рассмотрим задачу об изменении (увеличении) этого поток. Для этого сначала на рис. 20А рассмотрим ветвь (X_i, X_j) с пропускной способностью d_{ij} и поток мощности $0 \leq f \leq d_{ij}$. Такой поток можно увеличить на величину $d_{ij} - f$ или уменьшить на величину f. Такие действия могут быть проделаны и на ветви, изображенной на рис. 20Б, с уже двумя ветвями: прямой и специально введенной, обратной.

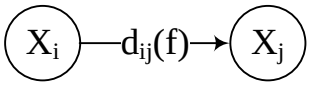


Рис. 20А. Поток f по ветви с пропускной способностью d_{ij}.

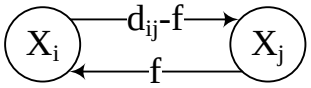


Рис. 20Б. Две ветви возможных изменений начального потока

Случай, когда между двумя вершинами уже существует и прямая и обратная ветви, представлен на рис.21А.

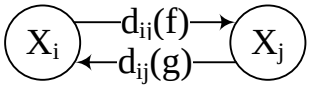


Рис.21А. Прямой и обратный потоки в ветвях между двумя узлами

Каждую из этих ветвей можно представить так, как это было сделано на рис.20Б. При этом получится четыре ветви, представленные на рис. 21Б.

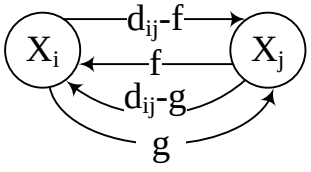


Рис.21Б Ветви возможных изменений потока. Общий случай

Параллельные, сонаправленные ветви можно объединить, суммировав их пропускные способности. Такое объединение называется склеиванием ветвей. Результат склеивания представлен на рис.21В.

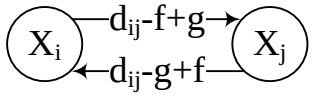


Рис.21В. Результат склеивания ветвей.

Из изложенного следует, что для получения всей сети приращения потока следует сначала дополнить каждую ветвь ей обратной с нулевой пропускной способностью, если обратной ветви изначально в сети не было. Если пропускная способность прямой ветви равнялась d_{ij}, пропускная способность обратной ветви равнялась d_{ji}, а поток в прямой ветви был равен f, а в обратной ветви g, то прямой поток f_1 в сети приращений и обратный поток g_1 в этой сети могут быть получены с помощью следующих формул:

$$f_1 = d_{ij} - f + g, \qquad g_1 = d_{ji} - g + f. \tag{28}$$

В распространенном частном случае, когда обратная ветвь отсутствует в исходной сети, формулы (28) приобретают следующий вид:

$$f_1 = d_{ij} - f, \qquad g_1 = f. \tag{29}$$

Из формул (28) и (29) следует, что если ветвь насыщена, а обратная либо отсутствует, либо ее поток равен нулю, то прямая ветвь отбрасывается. Возникающие в сети приращений потока ветви, ведущие в исток, должны быть отброшены, так как ищутся простые цепи, ведущие от истока к стоку, а не пути с циклами. Из этих же соображений отбрасываются возникающие ветви, ведущие из стока. На этом завершается алгоритм построения сети приращений потока.

Замечание 7. (Алгоритм Форда и Фалкерсона нахождения максимального потока в сети).

1-ый шаг. Выбирается начальный поток (например, нулевой, или тот, что предложен в пункте 5.2). Переходим ко второму шагу алгоритма.

2-ой шаг. Для предложенного потока строится сеть приращений потока, так как это было показано в Замечании 6. Переходим к третьему шагу алгоритма.

3-ий шаг. Если в сети приращений потока отсутствует цепь, ведущая от истока к стоку, то построен максимальный поток. Способы, позволяющие установить отсутствие цепи, или найти ее могут быть различны. Например, может быть использован метод Минти, считая что длина каждой ветви единица. Другой способ пометок Форда и Фалкерсона будет представлен ниже. Если цепи не существует, то алгоритм завершается. В противоположном случае переходим к четвертому шагу алгоритма.

4-ый шаг. Находим цепь из источника в сток и увеличиваем исходный поток на величину, равную минимальной пропускной способности из ветвей найденной цепи. Переходим ко второму шагу алгоритма.

Замечание 8. (О сходимости алгоритма Форда и Фалкерсона). Если все ограничения пропускных способностей ветвей являются натуральными числами, то алгоритм Форда и Фалкерсона сходится за конечное число шагов, потому что на каждом шаге происходит увеличение потока, по крайней мере, на единицу, а величина любого потока ограничена. Если ограничения на пропускные способности ветвей заданы в виде рациональных дробей, то за единицу можно принять число, обратное наименьшего общего кратного знаменателя этих дробей. При этом новая задача окажется целочисленной. Таким образом, алгоритм Форда и Фалкерсона может быть приспособлен для решения практических задач. Однако существуют очень простые примеры, когда этот алгоритм сходится не к максимальному потоку, для иррациональных в случае иррациональных пропускных способностей ветвей.

Пример 14. На рис. 22 А изображена сеть и начальный поток на ней.

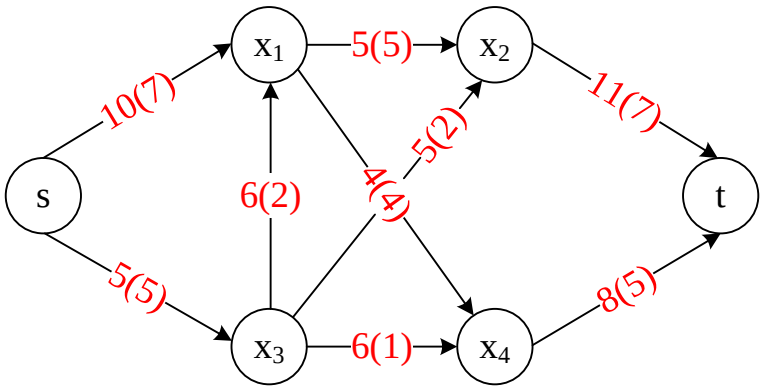


Рис.22 А. Сеть примера 14 для расчета максимального потока. Величина начального потока равна 12

Требуется найти максимальный поток методом Форда и Фалкерсона.

Решение

Не прибегая к формальным алгоритмам, на рис. 22А, найдем цепь, ведущую из истока в сток и не проходящую прямо по насыщенным начальным потоком ветвям, а также против направления ветвей с нулевым начальным потоком. Эта цепь изображена на рис. 22 Б.

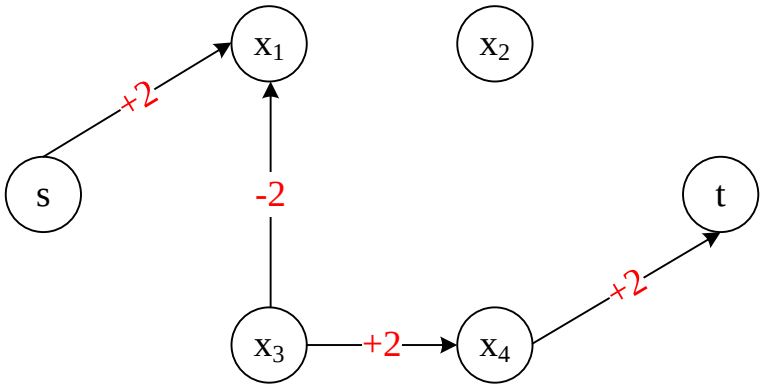


Рис. 22 Б. Цепь приращения исходного потока в примере 14.

Увеличим начальный поток на число два. При этом потоки ветвей $s \to X_1, X_3 \to X_4, X_4 \to t$ возрастут на два, а поток по ветви $X_1 \to X_3$ уменьшится на число два. На рис. 22 В изображен увеличенный поток. Он оказался максимальным.

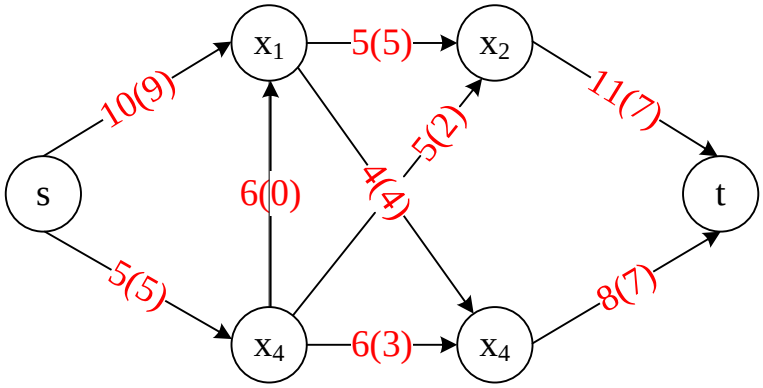


Рис. 22 В. Максимальный поток в сети величиной 14.

Для проверки того, что найден максимальный поток, найдем его критический разрез и удалим насыщенные ветви, ведущие из множества вершин $\mathbb{A}$ в вершины множества $\mathbb{B}$, а также ветви с нулевым потоком, ведущие из множества вершин $\mathbb{B}$ в множество вершин $\mathbb{A}$. Полученный оставшийся граф изображен на рис.22. Г.

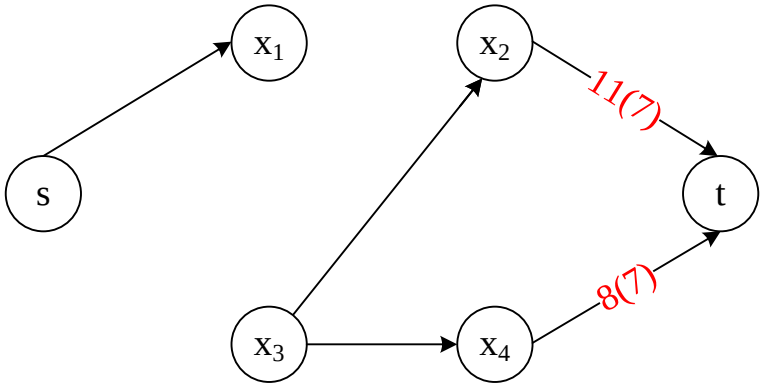


Рис. 22. Г. Орграф, после удаления ветвей разреза

После удаления ветвей разреза, орграф оказался несвязным. Не существует цепи, ведущей из истока s сток t. Следовательно, найденный на рис. 22.Б. поток является максимальным. Его величина равна 14.

5.5 Алгоритмы для нахождения цепи, по которой можно увеличить исходный поток

При реализации алгоритма Форда и Фалкерсона на ЭВМ необходимо иметь алгоритм нахождения пути из истока в сток сети, в которой удалены все насыщенные ветви, а новые пропускные способности остальных ветвей приняты разностями их исходных пропускных способностей и величины начального потока. Первый по времени возникновения стал употребляться алгоритм пометок. Этот метод был предложен Фордом и Фалкерсоном.

Алгоритм пометок вершин Форда и Фалкерсона

1) Сначала считается, что все вершины сети не просмотрены и не помечены.
2) Помечается исток.
3) На каждом этапе алгоритма произвольным выбирается не просмотренная, но помеченная вершина А и помечаются те еще не помеченные вершины В, в которые можно попасть из вершины А по их общей ветви. Эта пометка указывает, что в вершину В можно попасть из вершины А. Если таких вершин нет, то вершину А считают просмотренной и к ней в дальнейшем алгоритм не обращается. После этого переходят к следующей помеченной, но не просмотренной вершине. Если ни разу не удалось пометить ни одну вершину, то переходят к пункту 4.
4) Проверяется, включен ли сток в число помеченных вершин. Если нет, то алгоритм завершен и найденный поток является максимальным. Задача в этом случае решена. Если же сток входит в число помеченных вершин, то строится цепь от истока к стоку. Для этого на каждом шаге , начиная от стока, по пометкам находится предыдущая вершина искомой цепи.
5) Находится наименьшая пропускная способность ветви из найденной в предыдущем пункте цепи и увеличивается поток по найденной цепи на эту величину. Эта операция называется прорывом. Далее производится переход к пункту 3.

Пример 15. На рис. 23А представлена сеть из восьми вершин. Требуется найти цепь, ведущую из истока s в сток t методом пометок.

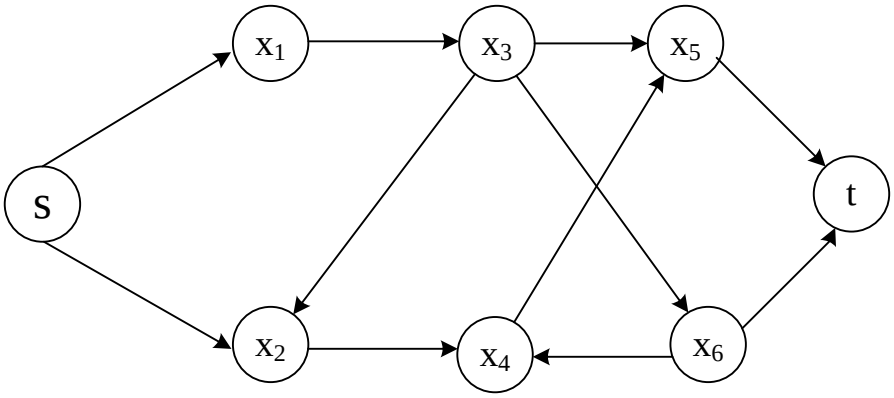


Рис. 23. А. Сеть к примеру 15

Решение

Сначала пометим вершину X_2 как доступную из истока S. Эта вершина становится помеченной, но еще не просмотренной. Поэтому в пометке участвует знак минус. Также пометим вершину X_1 как доступную из истока S. Исток помечается знаком плюс, как просмотренный. Алгоритм к нему более не обращается. Результат пометок представлен на рис. 23 Б, где не помеченные вершины имеют пометки вида $(n, -)$.

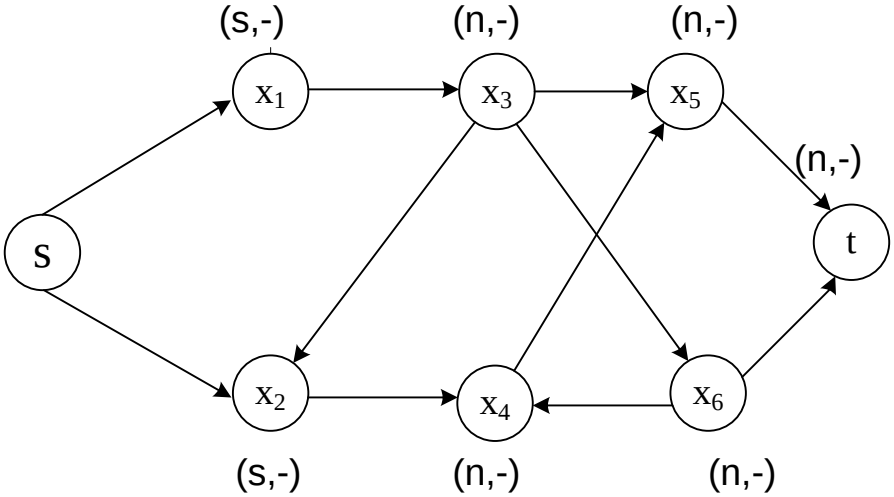


Рис. 23. Б. Сеть примера 15 после первой пометки

Всякий раз, выбирая новую, не помеченную вершину, будем исходить из последней помеченной вершины. Такой способ просмотра сети является одним из вариантов поиска по сети вглубь. На втором этапе пометим вершину X_3, как единственную доступную из вершины X_1. Вершина X_1 считается просмотренной и обозначается знаком плюс. Результат пометки представлен на рис. 23 В.

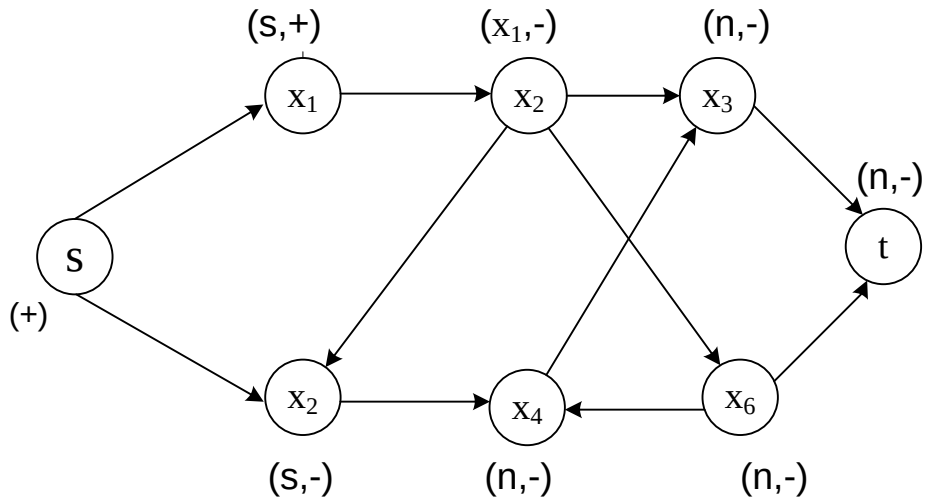


Рис. 23 В. Сеть примера 15 после второй пометки

На третьем этапе пометим вершину X_3, как доступную из вершины X_1. Доступную из вершины X_1, но уже помеченную вершину X_2 не помечаем вновь. Вершина X_1 теперь считается просмотренной и в пометку ставится знак плюс. В дальнейшем к ней алгоритм уже не обращается. Результат пометки представлен на рис. 23 Г.

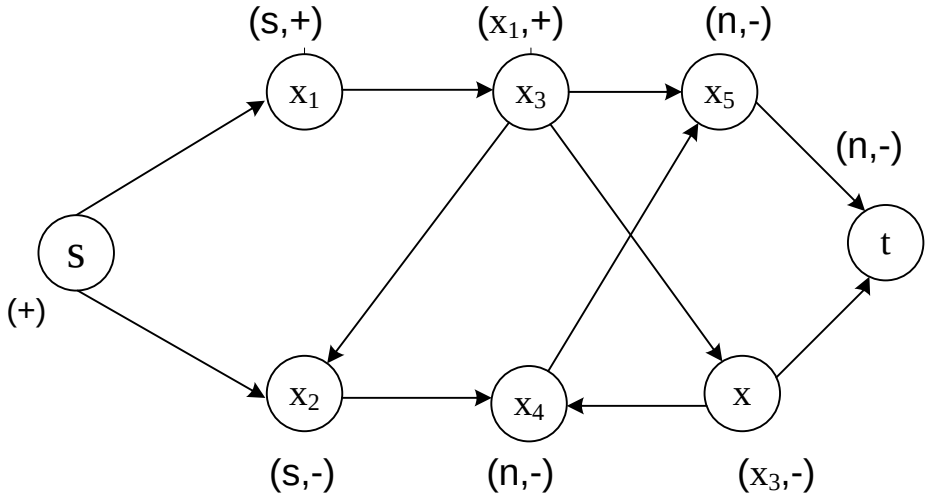


Рис. 23. Г. Сеть примера 15 после третьей пометки

На четвертом этапе помечаем вершину X_6, как доступную из вершины X_3. Доступную из вершины X_3, но уже помеченную вершину X_2 не помечаем вновь. Вершина X_3 теперь считается просмотренной и в ее пометку ставится знак плюс. В дальнейшем к ней алгоритм уже не обращается. Результат пометки представлен на рис. 23 Д.

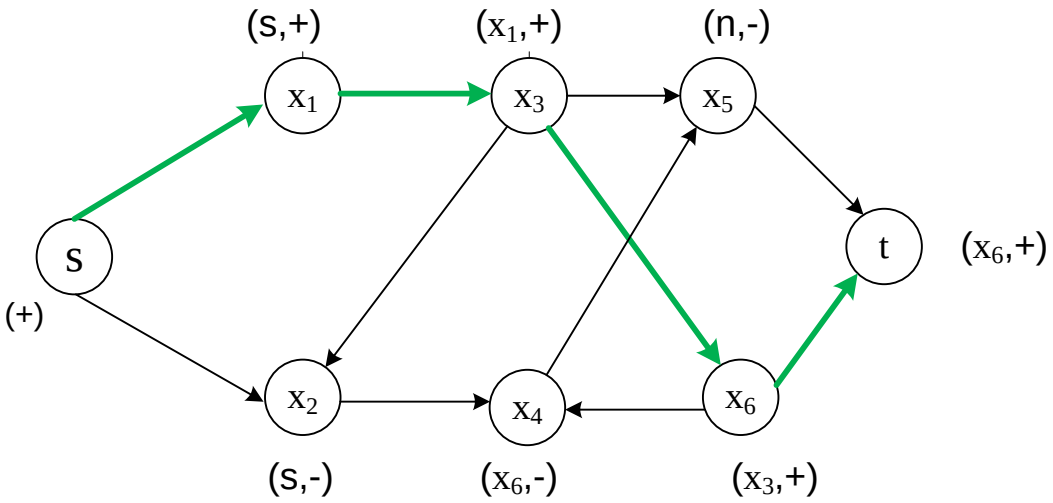


Рис. 23 Д. Сеть примера 15 после четвертой пометки

На пятом этапе помечаем вершины X_4 и t, как доступные из вершины X_6.

Произошел прорыв. Алгоритм добрался от истока до стока. Осталось найти цепь. Это мероприятие осуществляется в обратном порядке от стока к истоку по пометкам. Получим цепь: $S \rightarrow X_1 \rightarrow X_3 \rightarrow X_6 \rightarrow t$.

Пример 16. На рис. 24 А представлена сеть из семи вершин. Требуется убедиться методом пометок в отсутствии цепи, ведущей из истока s в сток t.

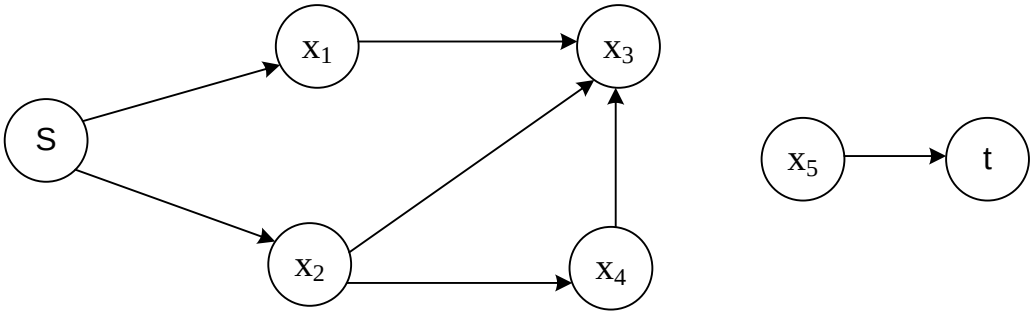


Рис. 24 А. Сеть к примеру 16

Решение

На первом этапе пометим вершины X_1 и X_2, как доступные из истока. Результаты пометок представлены на рисунках 24 Б и 24 В. Исток просмотрен.

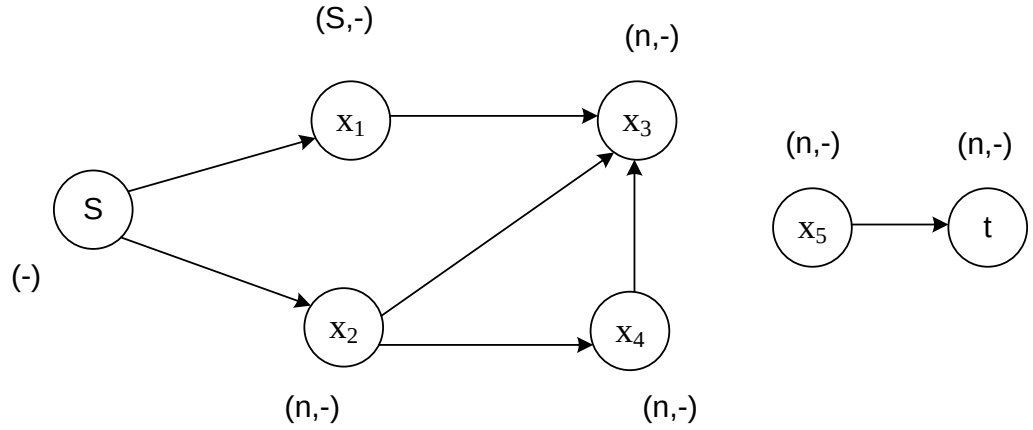


Рис. 24 Б. Первая пометка в примере 16

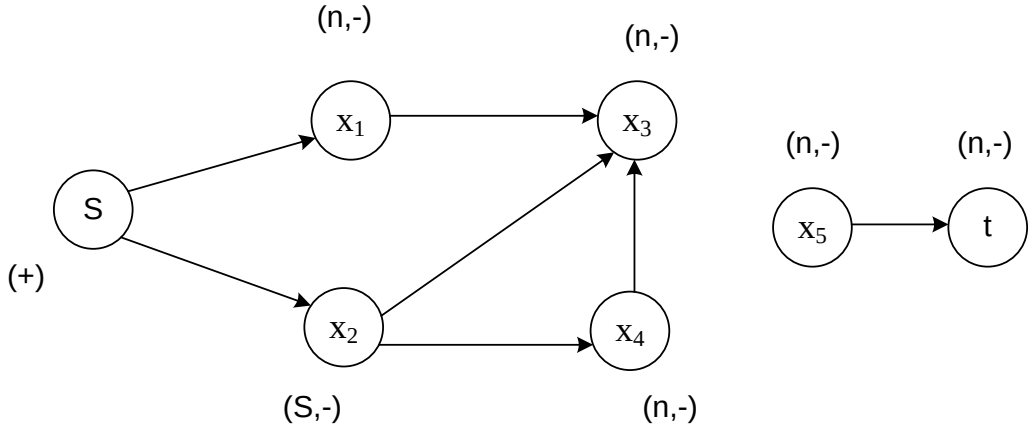


Рис. 24 В. Вторая пометка в примере 16

На втором этапе пометим вершину X_3, как доступную из вершины X_1. Вершина X_1 теперь становится просмотренной. Результат пометок представлен на рисунке 24 Г.

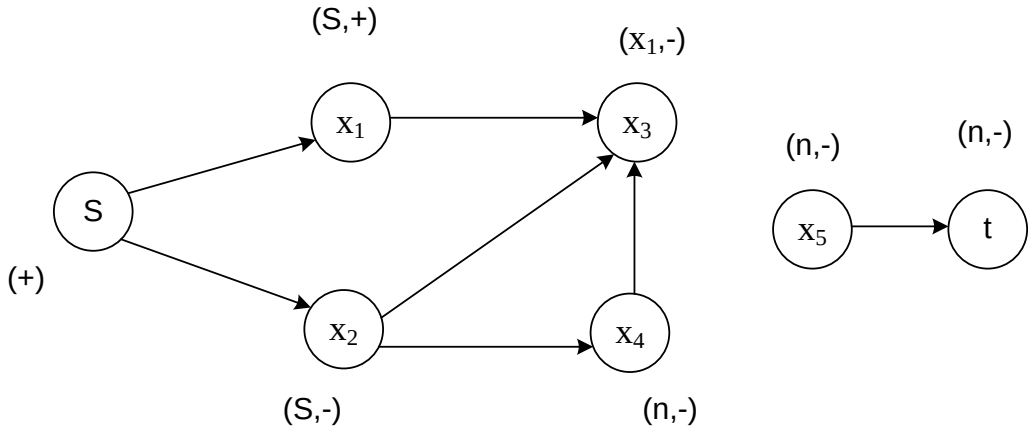


Рис. 24 Г. Третья пометка в примере 16

На третьем этапе пометим вершину X_4, как доступную из вершины X_2. Вершина X_2 теперь становится просмотренной. Результат пометок представлен на рисунке 24 Д.

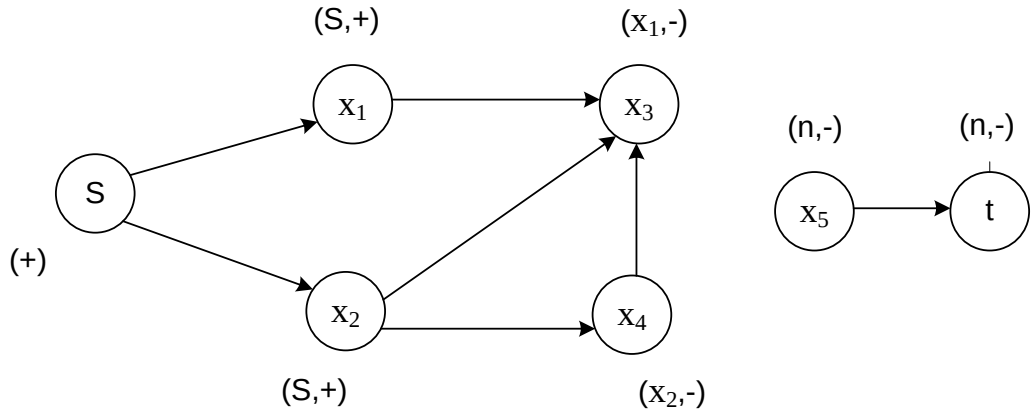


Рис. 24 Д. Четвертая пометка в примере 16

Просматриваем вершину X_3 и убеждаемся, что новых вершин с ее помощью пометить невозможно. Таким образом, эта вершина оказывается просмотренной. Результат пометки знаком плюс представлен на рисунке 24 Е.

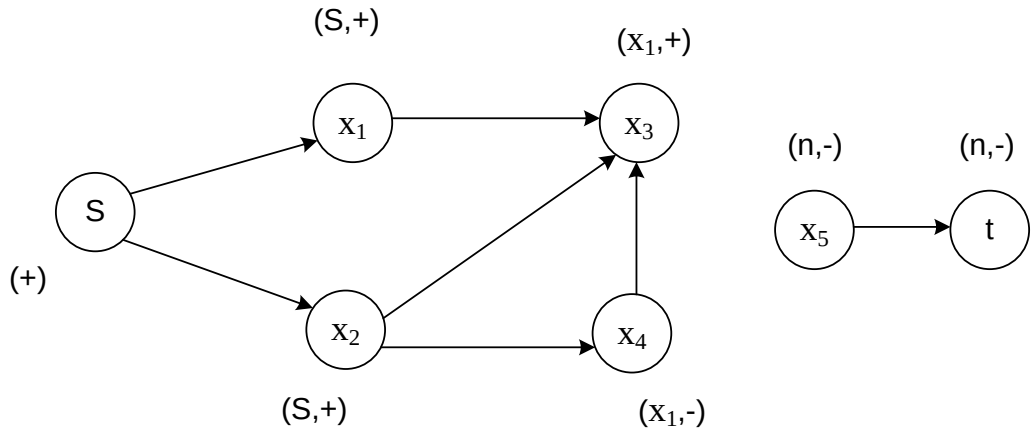


Рис. 24 Е. Пятая пометка в примере 16

Просматриваем вершину X_4 и убеждаемся, что новых вершин с ее помощью пометить невозможно. Таким образом, эта вершина оказывается просмотренной. Результат пометки знаком плюс представлен на рисунке 24 Ж.

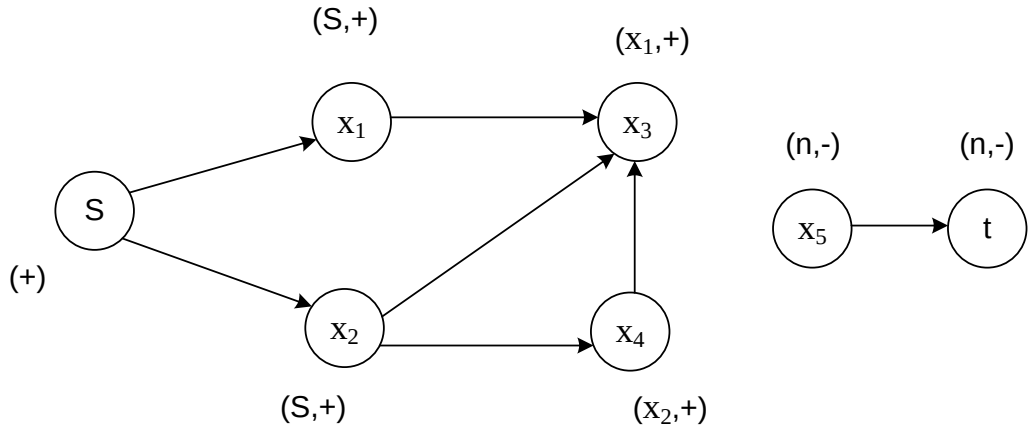


Рис. 24 Ж. Пятая пометка в примере 16

Дальнейшие действия алгоритма пометок стали невозможными. И алгоритм не пометил сток. Следовательно, прорыв невозможен. Найдем разрез. В его множество $\mathbb{A}$ входят просмотренные вершины: s, X_1, X_2, X_3, X_4 . В множество $\mathbb{B}$ входят вершины X_5 и t.

Алгоритм кратчайших путей

Этот алгоритм был предложен независимо Карпом (R.M.Karp) и Е.А. Диницем, соответственно, в 1972 и 1970 годах. Алгоритм находит не произвольную цепь между двумя вершинами, а кратчайшую по числу ветвей цепь между ними. Такую цепь можно найти с помощью алгоритма Форда-Беллмана, или с помощью алгоритма Минти. В обоих случаях длиной каждой ветви принимается равной единице.

6. Задача о максимальном потоке наименьшей стоимости

6.1 Пояснение к теме

Заметим, что обычно в сети имеется несколько различных потоков максимальной мощности. В таком случае можно выбирать из них наиболее подходящий. Если в сети для каждой ветви задана цена транспортировки единицы потока, то можно поставить задачу о нахождении максимального потока наименьшей стоимости. Это означает, что из всех потоков максимальной мощности требуется найти такой, который имеет наименьшую стоимость – сумму всех произведений потоков ветвей на цены транспортировок продукции по этим ветвям. Оказывается, что нахождение такого потока может быть организовано с помощью небольшой модификации алгоритма Форда и Фалкерсона. Такой алгоритм называется алгоритмом кратчайших путей.

Транспортная задача с ограничениями на пропускные способности ветвей легко сводится к задаче о максимальном потоке. Также к этой задаче сводится и задача об оптимальном назначении. Эти две последние задачи являются самыми распространенными среди сетевых задач. Поэтому алгоритм решения задачи о максимальном потоке наименьшей мощности имеет значительные приложения.

6.2 Алгоритм кратчайших путей

Пример 17. На рис. 25А представлена сеть с ограничениями на пропускные способности ветвей и с указаниями о ценах провоза единицы продукции по этим ветвям. В обозначениях сети на рис. 25А первая первое число является пропускной способностью ветви, а второе число представляет собой цену транспортировки единицы потока по ветви.

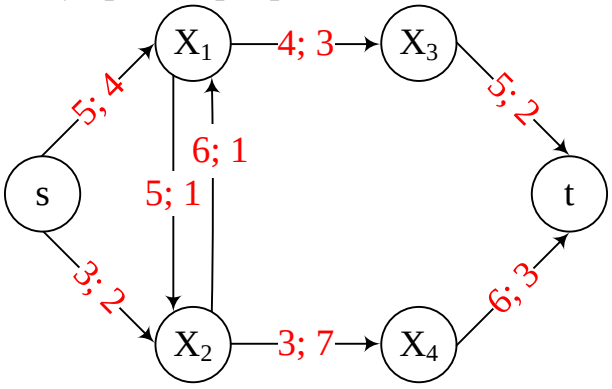


Рис. 25 А. Сеть к примеру 17.

Решение

Оставим в обозначениях только цены и опустим пропускные способности. Методом Минти найдем наиболее дешевую цепь из истока в сток. Результаты расчетов представлены на рис. 25Б.

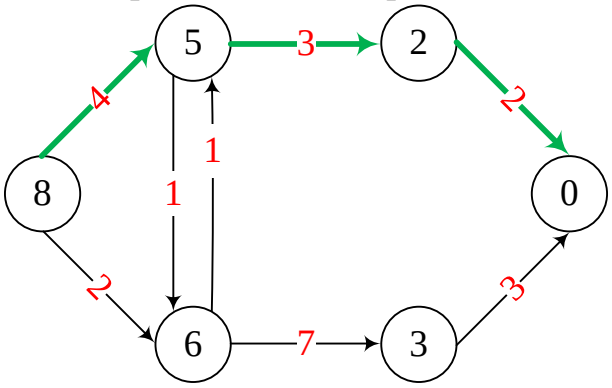


Рис. 25 Б. Самый дешевый начальный путь

Из рис. 25 А следует, что на указанной цепи минимальная пропускная способность равна 4. Следовательно, мощность первого найденного потока равна четырем. Стоимость C_1 найденного потока равна произведению его мощности на сумму стоимостей всех ветвей найденного пути:

$$C_1 = 4(4 + 3 + 2) = 36.$$

На рис. 25В изображена сеть, по которой следует строить приращение потока. Пропускные способности ветвей равны разности исходных пропускных способностей и величины найденного потока.

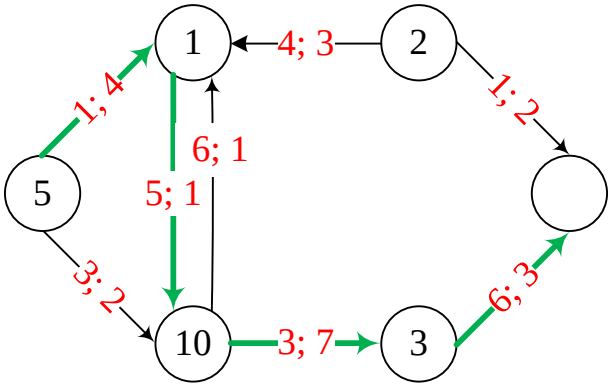


Рис. 25 В. Сеть для нахождения приращения потока после первого шага алгоритма минимальных путей 1,3

Приращение мощности потока равно $min(1,5,3,6) = 1$ Стоимость приращения потока равна $1(4 + 1 + 7 + 3) = 15$. Общая мощность потока после второй итерации равна 4+1=5, а общая его стоимость равна 36+15=51. Предпримем третью итерацию алгоритма. На рис. 23 Г представлена сеть для расчета методом Минти цепи наименьшей стоимости из истока в сток. Ветвь с нулевой пропускной способностью опущена.

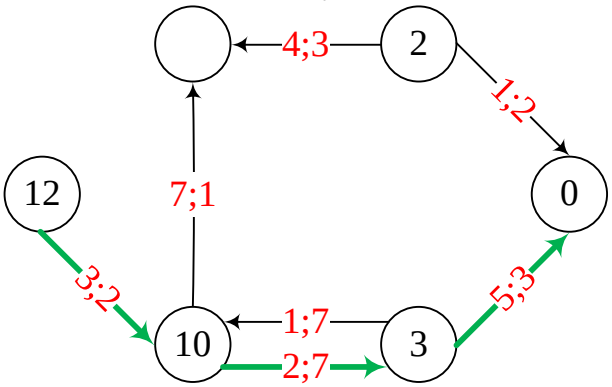


Рис. 25 Г. Результаты нахождения второй минимальной цепи методом Минти

Второй дополнительный поток равен минимальной пропускной способности найденной второй цепи. Его мощность равна 2. Мощность всего потока равна семи. Стоимость дополнительного потока $\Delta C_2 = 2 * (2 + 7 + 3) = 24$. Общая стоимость найденного после второй итерации потока $C_2 = 51 + 24 = 75$.

Сделаем четвертую итерацию алгоритма. На рис. 25 Д представлена сеть, на которой следует искать приращение потока.

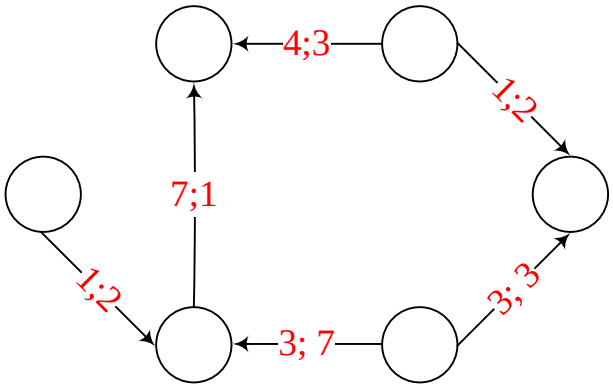


Рис. 25 Д. Сеть для третьей итерации алгоритма

Из рис. 25 Д следует, что не существует цепи из истока в сток. Следовательно, на предыдущей итерации уже найден максимальный поток минимальной стоимости. Он изображен на рис. 25 Е.

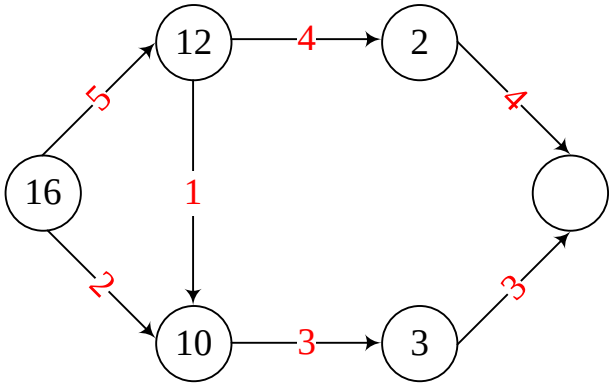


Рис. 25 Е. Максимальный поток минимальной стоимости.

Ответ: мощность максимального потока наименьшей стоимости равна 7, а его стоимость 75.

В многих случаях вместо алгоритма Минти приходится использовать алгоритм Форда-Беллмана, который работает и при отрицательных длинах (стоимостях) ветвей.

7. Транспортная задача в сетевой форме

Дадим постановку транспортной задачи линейного программирования в сетевой форме с ограничениями на пропускные способности. Задана сеть, ветви которой снабжены величинами пропускных способностей и ценами перевозки единицы груза. В сети выделено несколько истоков $X_1, \dots, X_n$ с положительными потоками $a_1, \dots, a_n$ и несколько стоков $Y_1, \dots, Y_m$ с положительными потоками $b_1, \dots, b_m$. В сбалансированной транспортной задаче выполняется условие:

$$a_1 + \cdots + a_n = b_1, \dots, b_m. \qquad (28)$$

Требуется найти план перевозок всех входных потоков, имеющий наименьшую стоимость, или убедиться в том, что все перевезти невозможно.

Стоимость потока равна сумме произведений цен ветвей на величины потоков по ветвям.

Такая задача легко сводится к задаче нахождения максимального потока наименьшей стоимости в двухполюсной сети. Для этого вводится единый исток, который соединен ветвями нулевой цены и бесконечной пропускной способности a_i с каждым X_iиз истоков транспортной сети. Кроме того вводится единый сток, с которым соединен каждый из стоков транспортной сети такой ветвью с каждым из стоков Y_j, которая имеет нулевую цену транспортировки и пропускную способность b_j.

Пример 18. На рис. 26А представлена транспортная сеть. Требуется найти оптимальный план перевозок.

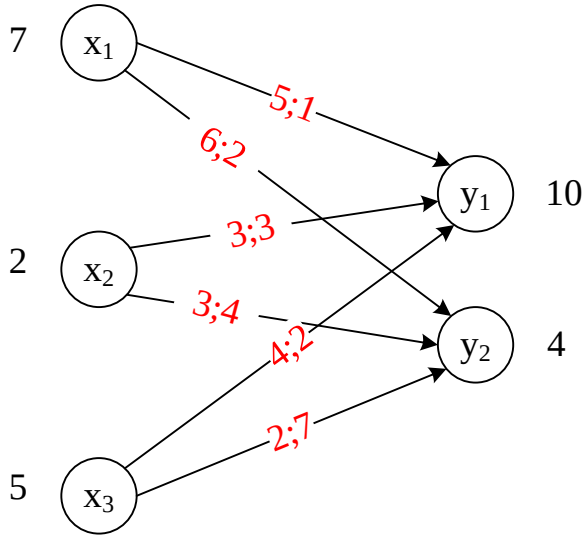


Рис. 26А. Транспортная сеть к задаче 18

Решение

Дополним сеть истоком и стоком. На рис. 26Б представлена двухполюсная сеть, эквивалентная исходной транспортной сети и пригодная для нахождения максимального потока наименьшей стоимости.

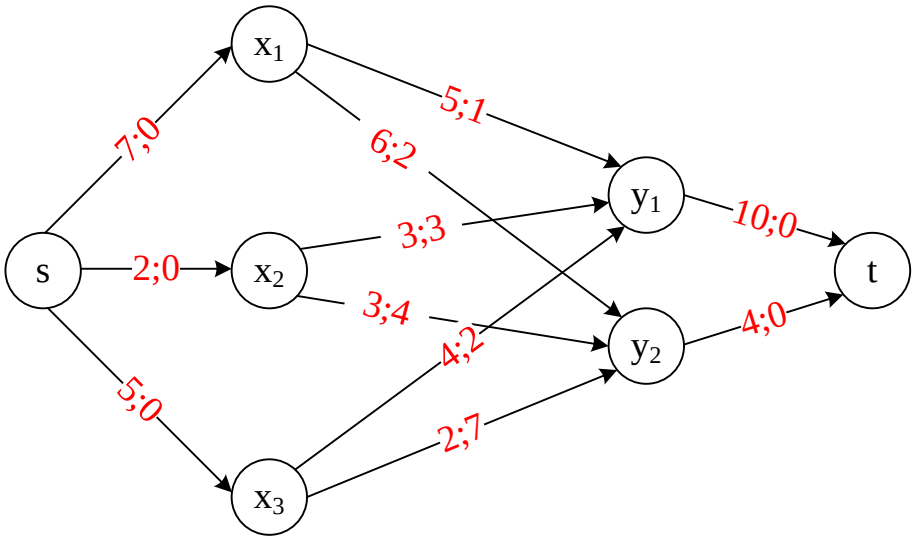


Рис. 26Б. Двухполюсная сеть для решения транспортной задачи

Последовательно найдем максимальный поток наименьшей стоимости . На первом этапе найдем начальный поток. Для этого оставляем только величины цен ветвей и применяем алгоритм Минти. На рис. 26В изображена соответствующая сеть и указана цепь, по которой направляется первое приближение максимального потока $s \to X_1 \to Y_1 \to t$. Этот начальный поток имеет мощность $min(7,5,10) = 5$ и стоимость $C_1 = 5 * (0 + 1 + 0) = 5$.

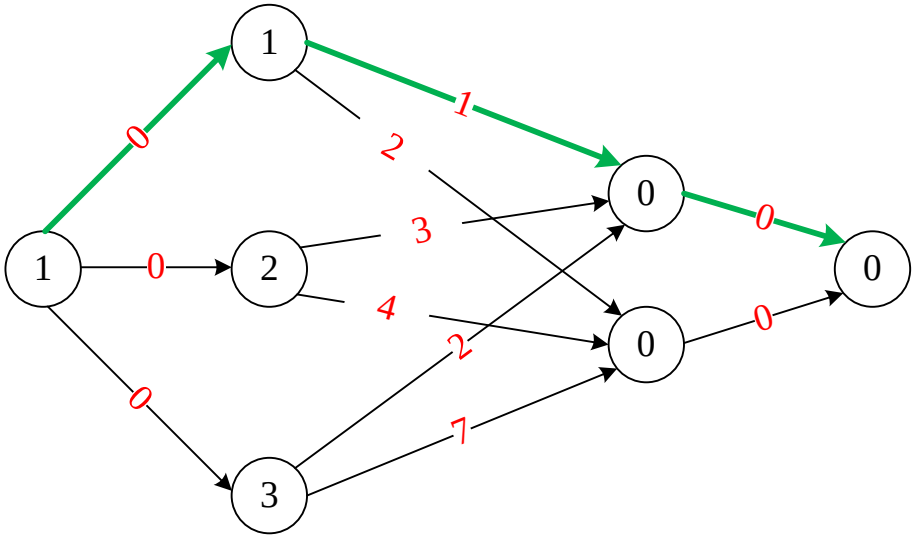
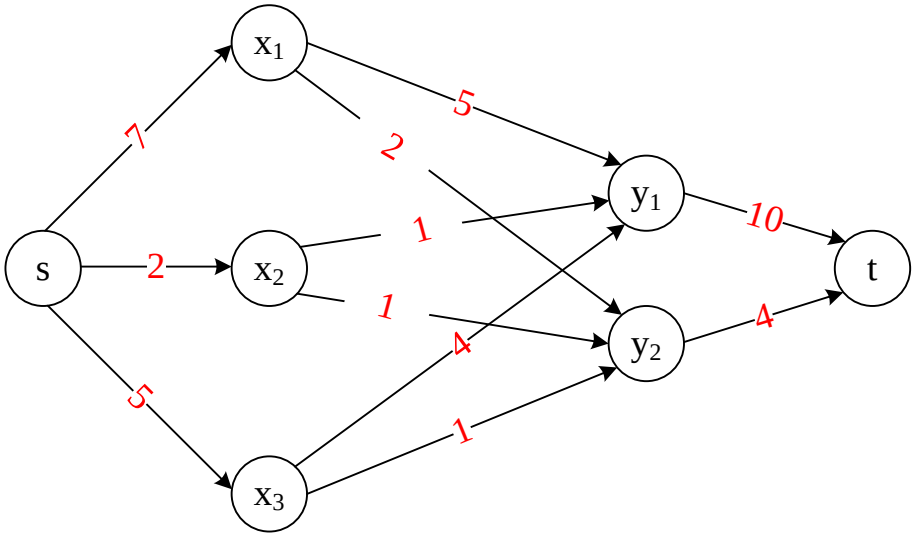


Рис.26.В. Сеть для первой итерации алгоритма для расчета методом Минти



На рис.26Г представлена сеть для нахождения приращения потока на второй итерации алгоритма кратчайших путей.

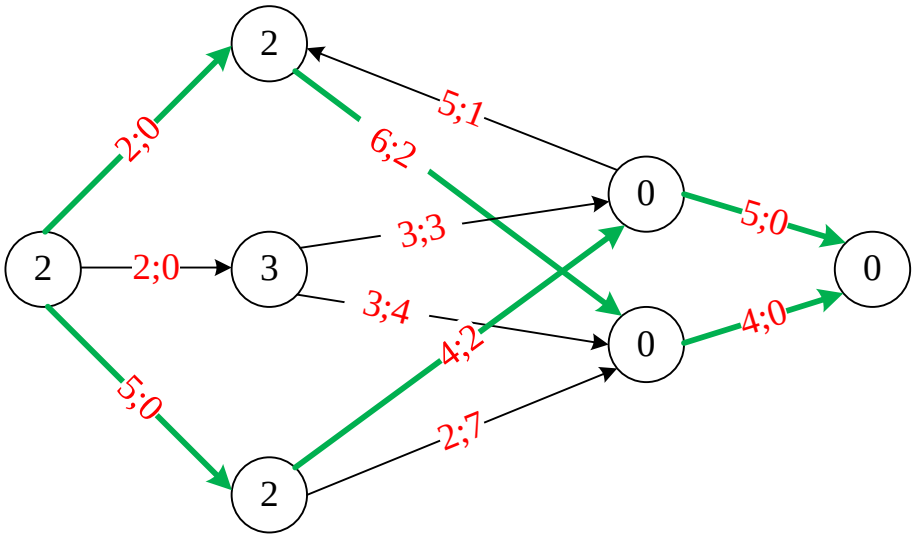


Рис.26Д. Сеть для второй итерации алгоритма

В данном случае найдены сразу две цепи с равными по стоимости потоками. Их оба можно прибавить к потоку, полученному на предыдущем шаге алгоритма. Первый поток проходит через вершины s, X_1, Y_2 и t, имеет мощность 2 и стоимость 4. Второй поток проходит через вершины s, X_1, Y_2 и t, имеет мощность 4 и стоимость 8.Общий поток после второй итерации имеет мощность 11, а его стоимость равна 5+4+8=17. На рис.26Е изображена сеть для нахождения приращения потока на третьем шаге алгоритма.

Перейдем к третьему этапу алгоритма расчета максимального потока минимальной стоимости. Сеть для нахождения цепи методом Минти показана на рис.26Д. В ней опущена одна ветвь рис. 26Г с нулевой пропускной способностью

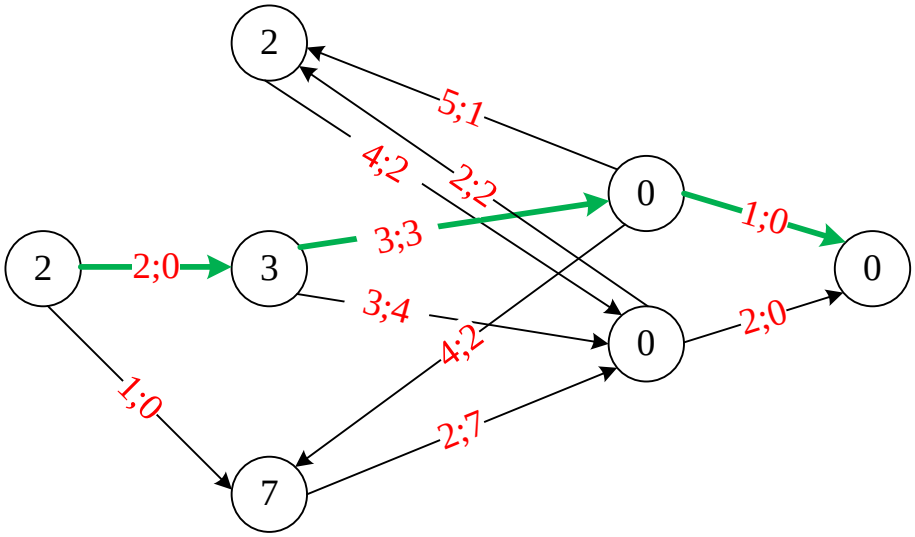


Рис. 26Е. Сеть для расчета кратчайшего пути на третьей итерации метода

Приращение потока

$$s \to X_2 \to Y_1 \to t$$

Имеет мощность единицу и стоимость 3.. Мощность всего потока после третьей итерации алгоритма равна 11+1=12, а его стоимость 17+3=20.

Перейдем к четвертой итерации алгоритма. На рис.26Ж изображена сеть для нахождения приращения потока на этой итерации.

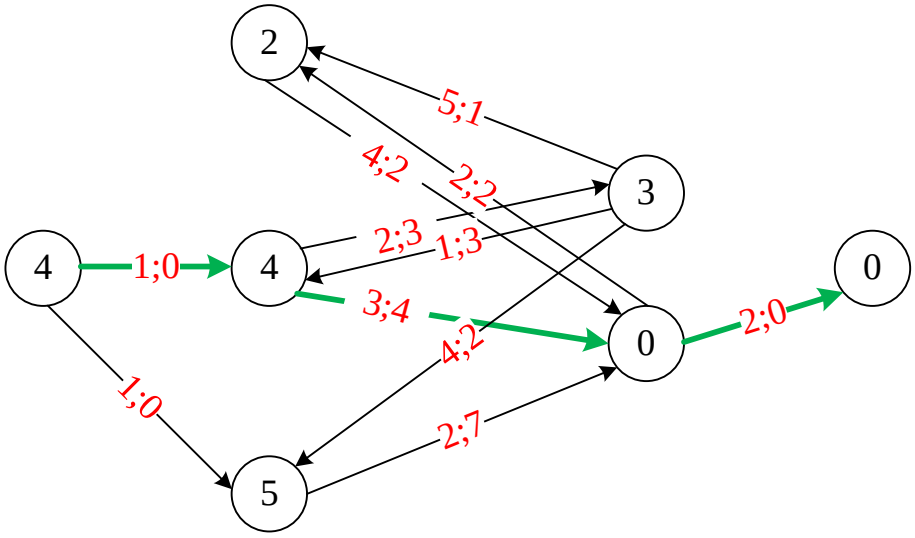


Рис. 26Ж. Сеть для третьего этапа алгоритма

Методом Минти найдена цепь из истока в сток $s \to X_2 \to Y_1 \to$t.

Приращение потока равно единице, а приращение стоимости равно 3. Мощность всего потока после четвертой итерации алгоритма равна 12+1=13, а стоимость потока 20+3=23.

Перейдем к пятой итерации алгоритма. На рис. 263 показана сеть для нахождения цепи на третьем шаге алгоритма

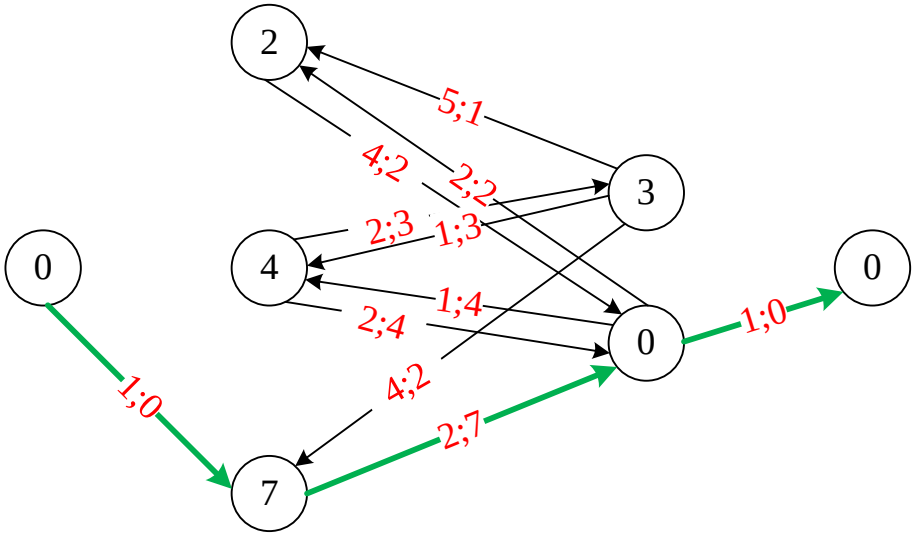


Рис.263. Цепь для приращения потока на пятом этапе работы алгоритма

Методом Минти получен маршрут, изображенный на рис.263 $s \to X_3 \to Y_2 \to$t. Приращение мощности потока равно единице, а стоимость приращения равна 7. После пятой итерации мощность потока равна 12+1=13, а его стоимость равна 23+7=30.

Шестая итерации уже не нужна, потому, что исчерпаны проводимости всех ветвей из истока.

На рис.26И представлена сеть с найденным максимальным потоком наименьшей стоимости.

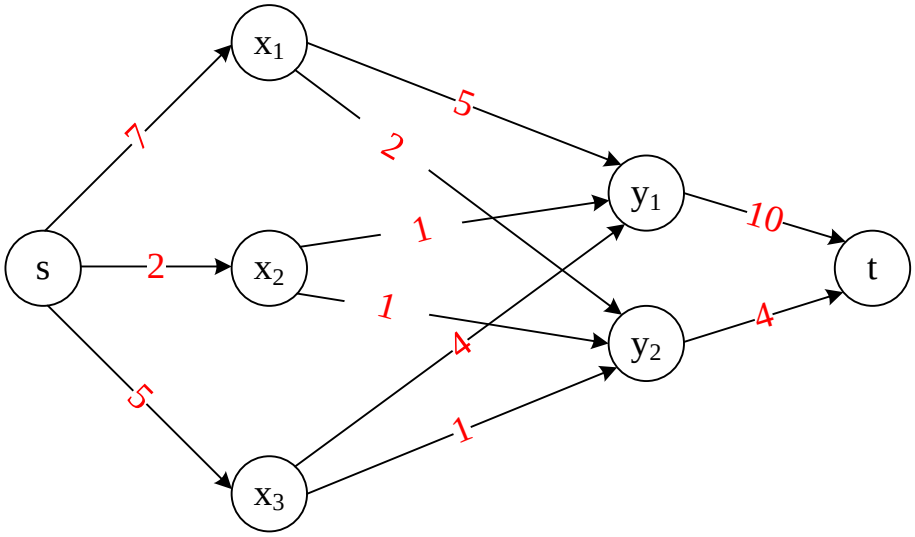


Рис.26И. Максимальный поток наименьшей стоимости.

Ответ: наилучший план транспортировки в транспортной задаче таков: из первого источника следует направить в первый сток поток мощностью 5, во второй сток – мощностью 2; из второго истока следует направить единичные потоки в первый и второй стоки; из третьего истока следует направить поток мощностью 4 в первый сток и единичный поток во второй сток. При этом суммарная стоимость найденного потока равна 30.

8. Сетевой алгоритм решения задача об оптимальном назначении

Дадим математическую постановку задачи об оптимальном назначении. Пусть дано n объектов $X_1, \dots, X_n$ и m объектов $Y_1, \dots, Y_m$, причем $n \le m$. Кроме того, считаются существующими некоторые связи, но не все, (X_i, Y_j). Для этих связей заданы их стоимости C_{ij}. Требуется из числа $Y_1, \dots, Y_m$ выбрать n элементов и сопоставить каждый них ровно одному из $X_1, \dots, X_n$.Такое сопоставление называется назначением (например, назначением на должность, или работу). При этом сумма стоимостей назначений должна быть наименьшей.

Замечание 8. 1) Если требуется решить задачу в случае $n \le m$, то она сводится к предыдущей заменой мест множеств $\{X_i\}$ и $\{Y_j\}$. 2) Если в задаче требуется максимизировать сумму стоимостей, то такая задача сводится к поставленной путем изменения знака у всех стоимостей C_{ij}.

Задача об оптимальном назначении является простым частным случаем транспортной задачи. В этом случае ограничения пропускных способностей всех ветвей равны единице. Кроме того, не может существовать промежуточных вершин. Все вершины являются либо истоками, либо

стоками. Для сетевого решения задачи об оптимальном назначении также как и в транспортной задаче добавляются две вершины – единый исток и единый сток.

Пример 19. Транспортная сеть может быть задана с помощью таблицы стоимостей назначений. В этом примере с четырьмя $\{X_i\}$ и четырьмя Y_j задается их матрица смежностей:

5	∞	1	∞
2	6	∞	∞
∞	4	7	3
3	∞	∞	5

Наличие в таблице бесконечных величин стоимостей назначений говорит о невозможности этих назначений. Требуется решить задачу оптимального назначения четырех людей на четыре должности на минимум.

Решение

Добавив единый сток и единый исток, получим сеть, изображенную на рис.27 А. В сети указаны стоимости каждой из ветвей. Ограничения пропускных способностей ветвей равны единице.

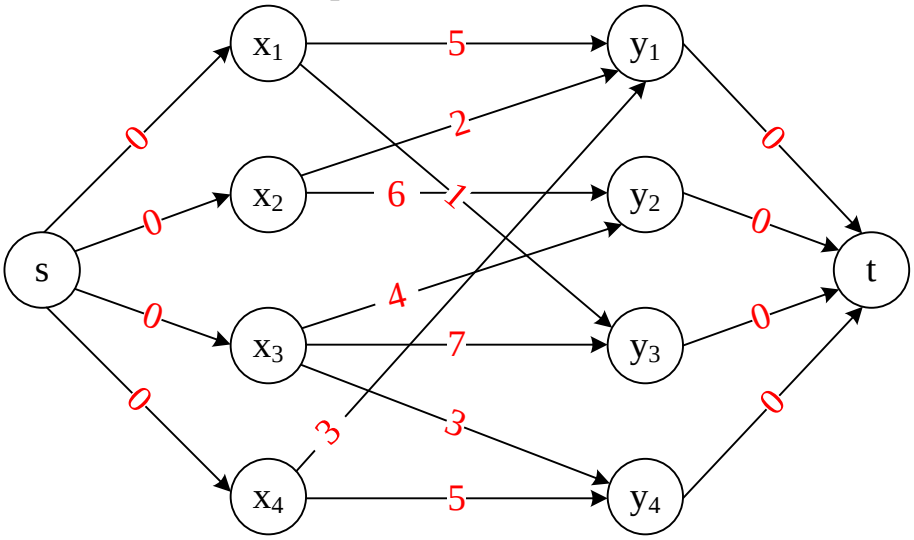


Рис.27А. Сеть для решения задачи об оптимальном назначении

Будем искать максимальный поток наименьшей стоимости. На первом этапе найдем кратчайшую цепь между стоком и истоком методом Минти. Результат расчетов показан на рис.27 Б.

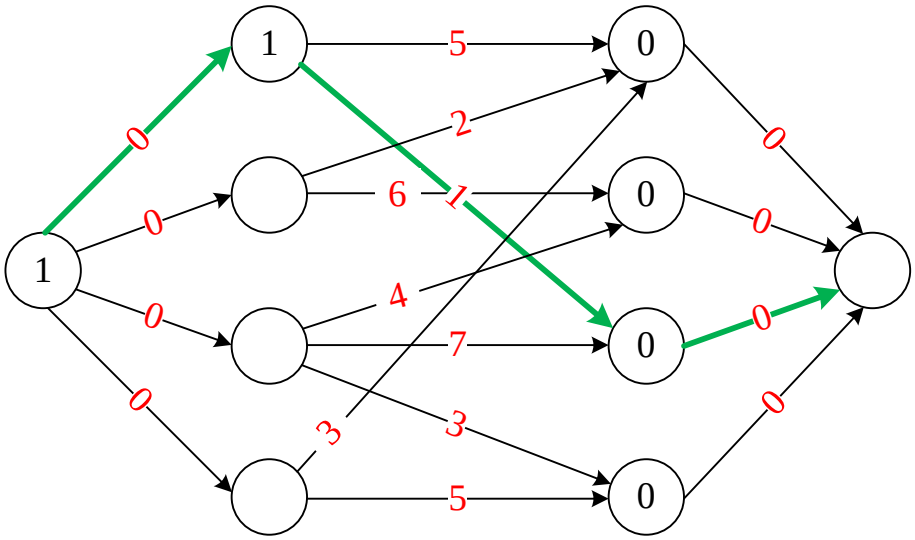


Рис.27Б. Начальный поток

Мощность начального потока равна единице, а его стоимость также равна единице. Для получения приращения потока построим сеть, изображенную на рис. 27 В.

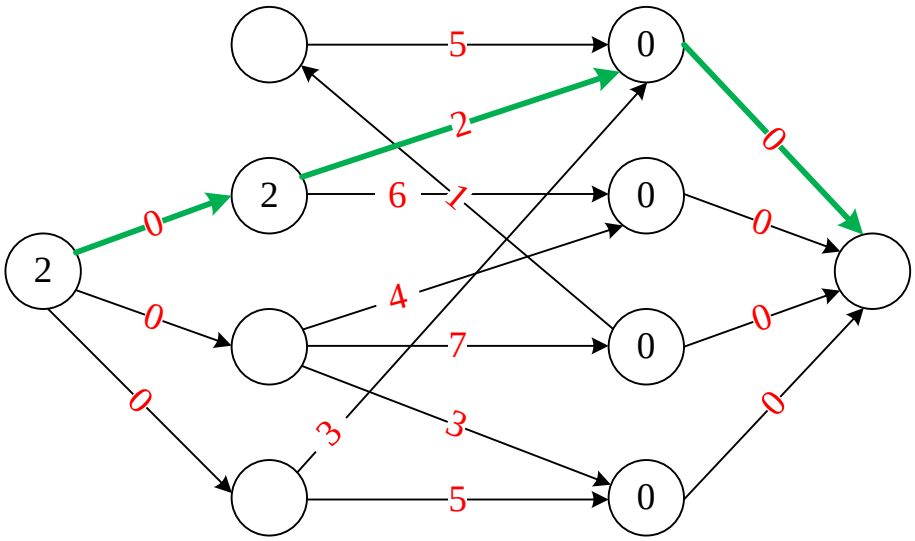


Рис.27 В. Сеть приращений для второй итерации

Приращение потока равно единице, а приращение его стоимости равно двум.

Сеть для расчета приращения потока на третьей итерации алгоритма представлена на рис. 27 Г.

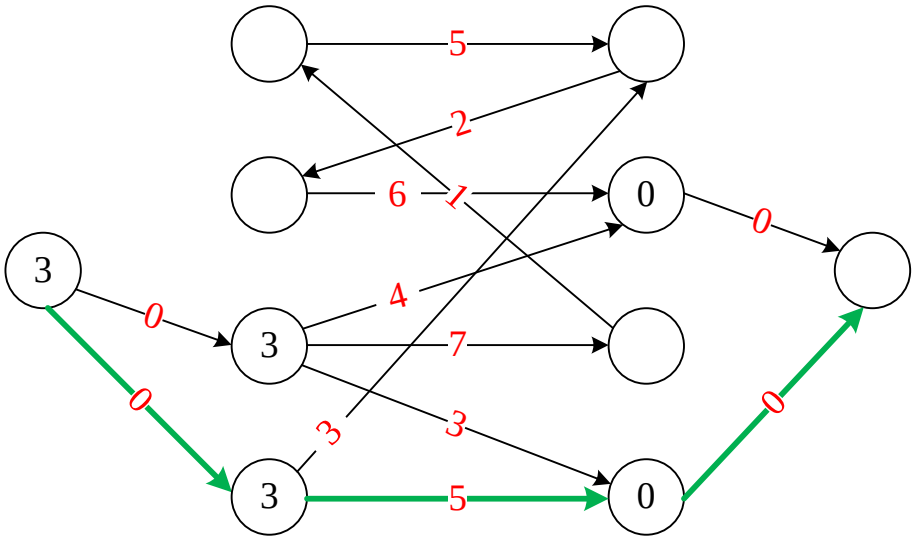


Рис.27 Г. Сеть для расчета приращения потока на третьей итерации алгоритма

Приращение потока равно единице, а приращение его стоимости равно 5.

Сеть для расчета приращения потока на четвертой итерации алгоритма представлена на рис.27 Д.

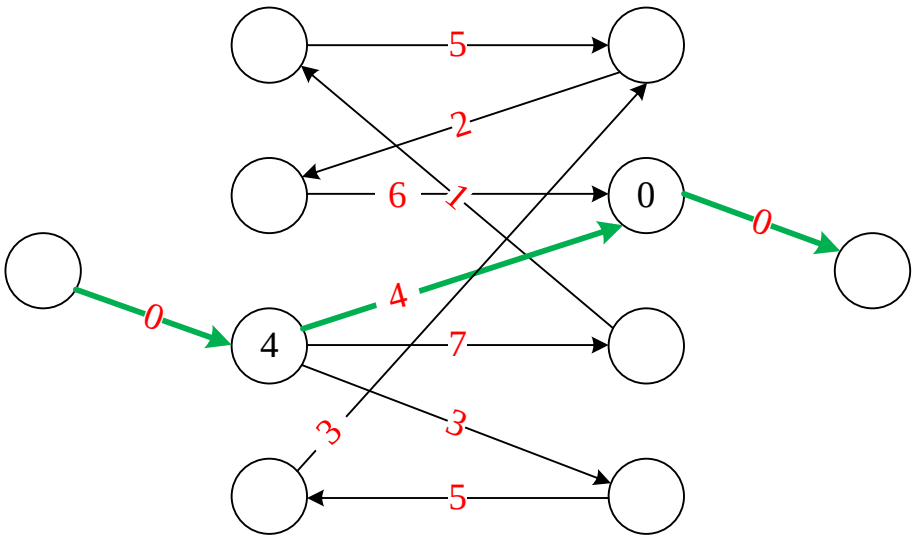


Рис.27 Д. Сеть для расчета приращения потока на четвертой итерации алгоритма

Приращение потока равно единице, а приращение его стоимости равно 4.

Дальнейшие итерации невозможны, так как исчерпались все четыре ветви, идущие из истока. Ответ в табличном виде имеет следующий вид:

		1	∞
1			
	1		
			1

Первый назначается на третью работу, второй назначается на первую работу, третий назначается на вторую работу и четвертый назначается на четвертую работу. Стоимость оптимального назначения равна 1+2+5+4=12.

9. Эвристические методы решения задач оптимизации

Известна фраза великого изобретателя Томаса Альвы Эдисона (1847 – 1931) о том, что нет ничего практичнее хорошей теории. Однако, у многих теорий имеется такой недостаток, как отсутствие достаточной широкой области применения. В задаче о назначении нет возможности назначения одного лица сразу на несколько видов работ. Такая ситуация часто возникает в практике. Например, каждый из лекторов кафедры университета читает несколько курсов. Другим примером может служить задача о ранце, в которой требуется нагрузить некоторый объем наибольшим по ценности набором различных грузов. Еще один пример – описанная выше задача коммивояжера. Задачи составления расписаний работы транспорта, людей и механизмов относятся к проблемам, для которых не разработаны общие универсальные и красивые математические теории. Эти и многие другие задачи не удается свести к классическим задачам, которые были подробно описаны выше. В то же время такие задачи часто возникают и их решение сильно востребовано практикой. Общий подход к решению этих проблем заключается в разработке эвристических алгоритмов решения, в которых на некоторых этапах могут быть использованы и точные научные алгоритмы.

Простейшим способом решения всех задач дискретной оптимизации является полный перебор всевозможных вариантов их решения с последующим выбором из них наилучшего. У такого лобового пути может возникнуть описанное выше препятствие, которое выражается в чрезвычайно большом общем числе вариантов, порожденным «бичом размерности». Поэтому перебор всех вариантов применяется только в самых простых ситуациях, когда ЭВМ способна справиться таким способом с задачей за разумное время. В остальных случаях следует каким-либо способом ограничить число перебираемых вариантов решения задачи.

Один из этих способов называется «Жадным алгоритмом». Он может иметь шанс на использование в тех случаях, когда задача разбивается на несколько этапов, на каждом из которых имеется некоторое число вариантов выбора решения. Например, в задаче оптимальной маршрутизации при движении по ветвям орграфа на каждом этапе можно выбирать кратчайшую в ближайшую ветвь и отправляться в следующую вершину орграфа по ней. Такой алгоритм может дать правильный результат, но может подарить и не оптимальное решение, а может и вообще завести в тупик и не дать никакого решения. Другой пример применения «Жадного алгоритма» при

восхождении альпинистами на самую высокую вершину некоторого горного массива в некоторых случаях приводит к достижению этой вершины (метод наискорейшего подъема). Но в других случаях может сильно обмануть участников восхождения, например, заведя их на соседнюю, не самую высокую, вершину. Часто в тех случаях, когда «Жадный алгоритм» дает правильный результат, то он его получает за небольшое число шагов. Поэтому, жадные алгоритмы высоко эффективны в смысле быстродействия, но употребим только в редких случаях. Для гарантированной работы здесь требуется доказательство их сходимости.

Другим общим способом решения задач дискретной оптимизации является Метод ветвей и границ. Этот метод совпадает со следующей идеей. В случае, когда требуется найти на графе некоторый путь, ведущий от исходной вершины к другой, заранее неизвестной, то сначала строится дерево графа (например, используется алгоритм Краскала). Далее, от дерева отсекаются заведомо неприемлемые ветви. Эти ветви могут давать слишком плохое значение оптимизируемой функции, или не входить в систему ограничений. После удаления ветвей, оставшуюся укороченную задачу решают методом перебора. В ряде случаев ветви убираются не заранее, а в ходе перебора. При этом используется уже собранная по ходу решения информация. При решении задач этим методом желательно оценить заранее объем вычислительной работы, чтобы гарантированно получит решение с помощью ЭВМ за разумное время. При таких оценках используется теория комбинаторного анализа. Если объем вычислений оказывается слишком большим, то требуются дополнительные способы отбрасывания ветвей, которые должен найти исследователь. В ряде случаев число вариантов на некотором этапе метода Ветвей и границ становится равным произведению нескольких чисел, общая сумма которых известна, в то время как сами эти числа не известны наперед. Оценка сверху числа вариантов в такой ситуации может быть получена путем решения следующей задачи, которая является частным случаем дискретной задачи оптимального распределения ресурсов.

Специальная дискретная задача оптимального распределения ресурсов. Найти наибольшее число A, являющееся произведением k натуральных чисел, сумма которых равна числу M:

$$A = \prod_{i=1}^{k} n_i \rightarrow \max, \quad \text{при условии} \sum_{i=1}^{k} n_i = M, n_i \in \mathbb{N}. \qquad (30)$$

Заметим, что с помощью логарифмирования функции цели A задача (30) сводится к классической дискретной задаче оптимального распределения ресурсов. Для решения задачи отметим, что если некоторое из чисел n_i

превосходит другое из них n_l более, чем на единицу, то функция цели A не достигает своего наибольшего значения. Действительно, в этом случае существует возможность ее увеличить, перебросив из большего n_i в меньшее n_l единицу. При этом получим, что

$$(n_i - 1)(n_l + 1) = n_i n_l + (n_i - n_l - 1) > n_i n_l.$$

Отсюда вытекает оптимальное решение задачи. При нем часть n_i должны равняться между собой, в то время как остальные обязаны превосходить их на единицу. Найдем аналитические выражения для n_i и для наибольшего значения A_{max}. Разделим целое число M на целое число k остатком r:

$$M = kq + r. \quad (31)$$

Единственное возможное оптимальное решение состоит из $k - r$ чисел q и r чисел $q + 1$.

$$A_{max} = q^{k-r}(q+1)^r = \left[\frac{M}{k}\right]^{k-r} * \left(\left[\frac{M}{k}\right] + 1\right)^r. \quad (32)$$

Приведем числовой пример: $M = 53, k = 5$. При таких исходных данных получим:

$$M = 5 * 10 + 3 \Rightarrow A_{max} = 10^2 * 11^3 = 133100.$$

Вопросы, упражнения и задачи по теме Оптимизация на сетях

1. Докажите, что, если в первом примере выбирать маршрут между пунктами A и B не из описанного там э семейства, то всегда найдется путь из описанного семейства маршрутов из пункта A в B с меньшим числом звеньев.
2. Сведите задачу поиска кратчайшего маршрута на метрополитене к задаче оптимальной маршрутизации.
3. Объясните, почему в примере 1 получилось, что $2^{100} > 10^{30}$.
4. Рассмотрите рис.9 и найдите другой оптимальный путь. Значение терминального функционала на найденном пути равно уже найденному значению. Найдите еще три оптимальных пути.
5. Предложите способ сведения задачи нелинейного программирования на максимум к другой задаче нелинейного программирования на минимум.
6. Предложите способ сведения задачи нелинейного программирования с ограничением типа равенство к другой задаче нелинейного программирования с ограничениями типа неравенство.
7. Докажите равенство

$$a + min(b, c) = min[(a + b), (a + c)].$$

8. Предложите новый способ нахождения оптимального маршрута без использования матриц переходов.

9. Какой особенностью отличается матрица смежности орграфа, одна из вершин которого не является конечной ни для одной ветви?

10. Какой особенностью отличается матрица смежности орграфа, одна из вершин которого не является начальной ни для одной ветви?

11. Модифицируйте алгоритм Минти для нахождения кратчайших маршрутов из всех вершин графа в любую из нескольких выделенных его вершин.

12. Можно ли решать задачу оптимальной маршрутизации на минимум с помощью варианта алгоритма, описанного в пункте 4.3, если орграф ациклический?

13. Как можно определить существование или отсутствие циклов в орграфе по его матрице смежностей?

14. Докажите, что в двухполюсной сети сумма входящих потоков в сток равна потоку равна сумме выходящих потоков из источника.

15. Докажите, что поток, построенный в Замечании 4, действительно является потоком, и при том, большим исходного потока.

16. Докажите, что поток в сети, где между двумя вершинами существует прямая и обратная ветви, потоки которых положительны, эквивалентен потоку, при котором поток одной из этих ветвей равен нулю.

17. Могут ли существовать в сети ветви, которые ведут в исток или выходят из стока?

18. Решите следующую задачу.

Найдите наименьшее число A, являющееся произведением k натуральных чисел, сумма которых равна числу M:

$$A = \prod_{i=1}^{k} n_i \rightarrow \min, \text{ при условии } \sum_{i=1}^{k} n_i = M, n_i \in \mathbb{N}.$$

Эта задача отличается от рассмотренной в п.8 задачи тем, что целевая функция минимизируется, а не максимизируется.

Указания к вопросам, упражнениям и задачам темы Оптимизация на сетях

1. В маршруте не из семейства, либо число движений вдоль ост абсцисс, либо вдоль оси ординат, либо каждый из этих двух видов движений содержат большее число шагов (ветвей графа).

2. $2^{100} = [(2)^{10}]^{30} = 1024^{10} > 1000^{10} = (10^3)^{10} = 10^{30}$.

3. Если движение идет по одной линии, простой путь один и продолжительность его нетрудно найти. Если движение происходит между станциями, расположенными на различных линиях, то необходимы пересадки. Для решения задачи с пересадками следует составить граф пересадочных узлов метрополитена с учетом времени пересадки Затем нужно добавить к пересадочным узлам начальный и конечный узлы искомого маршрута и решить задачу оптимальной маршрутизации для такого графа.

4. Существует пять оптимальных путей. Все они заканчиваются в одном и том же пункте С.

5. Вместо максимизации функции следует минимизировать другую функцию, равную исходной, умноженной на минус единицу.

6. Ограничение $f(x_1 \dots, x_n) = 0$ эквивалентно системе ограничений

$$\begin{cases} f(x_1 \dots, x_n) \leq 0 \\ f(x_1 \dots, x_n) \geq 0 \end{cases}.$$

7. Рассмотрите два случая: $b \geq c$ и $b < c$.

8. Для нахождения оптимального маршрута следует провести обратный ход алгоритма. Его можно осуществить с помощью подсчитанных степеней матриц. Например, для искомого маршрута в примере 7 из первой вершины во вторую найдем, как образовалась величина 13, при последнем умножении матриц. При умножении первой строки на второй столбец величина 13 получилась в виде суммы 13=5+8. Число 8, стояло в четвертой строке – следовательно, сначала на маршруте из первой вершины следует идти в четвертую. Теперь найдем элемент матрицы куба, стоящей в четвертой строке и равный 8. Посмотрим, как оно было образовано при умножении матриц. Оно было получено в результате умножения четвертой строки на второй столбец 4+4=8. Число 4 из матрицы квадрата стояло в пятой строке. Следовательно, следующая вершина оптимального маршрута – пятая. Посмотрим, как была образована величина 4 в пятой строке на втором месте при вычислении квадрата матрицы смежностей. Это число возникло при умножении пятой строки на второй столбец в виде суммы 1 и 3. Единица стоит в третьей строке матрицы смежности. Поэтому следующая вершина является третьей. Таким образом, найден оптимальный маршрут. С учетом начальной первой вершины и концом – последней вершины он выглядит следующим образом: 1-4-5-3-2. Этот результат совпадает с результатом примера 6.

9. Все элементы строки с номером этой вершины равны бесконечности, за исключением диагонального элемента, который, как и обычно, равен нулю.

10. Все элементы столбца с номером этой вершины равны бесконечности, за исключением диагонального элемента, который, как и обычно, равен нулю.

11. Измените первый этап алгоритма Минти.

12. Да.

13. Циклы в связном орграфе отсутствуют тогда и только тогда, когда матрица смежностей имеет ненулевые элементы выше своей главной диагонали.

14. Просуммируйте по всем промежуточным вершинам все входящие и исходящие потоки. С одной стороны, эта сумма равна нулю как сумма нулевых суммарных потоков в каждой промежуточной вершине. С другой стороны она равна сумме всех потоков. Каждый поток, между промежуточными вершинами, входит дважды: как отходящий – со знаком минус, и как входящий – со знаком плюс. Эти слагаемы в общей сумме потоков сократятся. Останется сумма потоков от источника за вычетом суммы потоков, ведущих в сток. Так как эта сумма ноль, то сумма потоков от источника равна сумме потоков, ведущих в сток.

15. Если к промежуточной вершине орграфа подходит одна из ветвей найденной цепи, а другая отходит, то они идут в одном направлении. При этом оба потока увеличиваются на одну и ту же величину. Это не нарушает баланса потоков в узле. Если эта ветвь к промежуточной вершине орграфа подходят, или от нее отходят обе ветвей найденной цепи, то они идут в разных направлениях. При этом поток одной из ветвей увеличивается, а другой – уменьшается на одну и ту же величину. Это также не нарушает баланса потоков в узле. Следовательно, заданные изменения в сети приводят к новому потоку. Этот поток больше исходного на величину ε, потому что увеличивает на это число одну ветвь, ведущую из истока.

16. Вычтете величину меньшего потока из обеих ветвей. Проверьте балансы потоков в обеих ветвях.

17. Такие ветви могут быть отброшены в силу результата предыдущего упражнения.

18. В оптимальном решении все $n_i = 1$, кроме одного.