1. Сетевое планирование, алгоритм правильной нумерации.

При исследовании операций часто приходится решать задачи рационального планирования сложных, комплексных проектов:

- строительство больших промышленных объектов;
- развертывание систем медицинских или профилактических мероприятий;
- выполнение комплексной научно-исследовательской темы с участием ряда организаций и т. д.

Характерной особенностью таких проектов является то, что многие из составляющих их работ взаимообусловливают друг друга: одни работы не могут начаться до завершения других.

Планирование любого комплекса такого рода работ должно осуществляться с учетом следующих факторов:

- времени на выполнение всего цикла работ и его составляющих;
- стоимости комплекса и его составляющих;
- сырьевых, энергетических и людских ресурсов.

Рациональное планирование требует ответа на вопросы:

- как распределять имеющиеся ресурсы;
- в какие моменты начинать каждую из работ;
- какие могут возникнуть проблемы из-за несвоевременного выполнения отдельных работ и как их устранить и т. д.

Методы решения таких задач предлагает ***сетевое планирование*** - рациональное планирование (включая управление) реализации комплексных проектов на основе формализации в виде так называемых сетевых графиков (СГ). В сетевом планировании решаются как *прямые задачи* - оценка последствий определенного решения, так и *обратные* - поиск наилучших решений.

СГ – это математическая модель проекта в виде орграфа, который отображает технологические связи между работами, составляющими проект.

Для построения СГ необходимы:

- перечень работ, входящих в проект;
- сведения о непосредственном предшествовании (или следовании) работ (их упорядочении во времени);
- продолжительность каждого вида работы.

Отношения непосредственного предшествования (или следования) задают частичный порядок, который можно представить в виде орграфа,

называемого сетевым графиком. При этом часто работы, составляющие проект, изображают в виде пронумерованных дуг с ориентацией от момента начала работы до момента ее окончания. Именно эти моменты будут соответствовать вершинам орграфа.

Вершины построенного СГ называют событиями. Каждое событие отражает тот факт, что все входящие в него работы завершены и только после этого может начинаться выполнение последующих работ.

Одно и то же событие является либо началом одних, либо концом других дуг-работ.

Если у него нет входящих дуг, то это событие называют началом проекта. Событие, не имеющее выходящих дуг, – концом проекта.

Последовательными называют работы, лежащие на некотором пути (или образующие путь) в графе G. Могут быть и параллельные работы (как дуги в мультиграфе).

Работа, формально заменяющая собой группу работ, расположенную между какими-то двумя событиями, называется составной. Такие работы применяются для укрупнения СГ.

Важную роль играют так называемые фиктивные работы нулевой продолжительности, вводимые в СГ для того, чтобы правильно отразить отношение порядка каждой из работ и следующими за ними работами. При необходимости вводятся фиктивные события - начала проекта и конца проекта. На СГ невозможно отобразить факт начала какой-то работы после завершения части предшествующей работы.

СГ называется правильно пронумерованным, если для каждой работы $(P_i, P_j), i < j.$

Алгоритм правильной нумерации

Э т а п I. Разбиение событий на классы. К событиям класса K0 относят все события, в которые не входит ни одной дуги. После этого удаляются дуги, выходящие из событий класса K0, и к событиям класса K1 относят все события, в которые не будет входить ни одна дуга и т. д. К очередному классу относят все события, которые не имеют входящих дуг после всех предыдущих удалений. Если на каком-то шаге окажется, что еще не все события разнесены по классам и что к очередному классу нельзя отнести ни одно событие, то это будет означать, что в сетевом графике имеется, по крайней мере, один контур, и, следовательно, СГ построен неверно, его надо исправлять.

В СГ без контуров после первого этапа все события будут разнесены по классам.

Э т а п II. Правильная нумерация. Событиям класса K0 приписываем в произвольном порядке новые номера $P'_0, P'_1, \ldots$ Начиная с очередного неиспользованного номера, в произвольном порядке нумеруются события класса K1, затем K2 и т. д. В соответствии с правилом образования классов между событиями одного класса дуг не может быть, и они направлены только из событий класса с меньшим номером в события класса с большим номером.

Это служит основанием того, что после перенумерации для каждой работы $(P'_i, P'_j), i < j.$

2. Параметры СГ: минимальный и максимальный моменты свершения событий, алгоритмы.

Минимальным моментом $\dot{T}_j^{(0)}$ свершения события Pj называют самое раннее возможное время окончания всех работ, входящих в это событие.

Условимся, что для начала проекта $P_0\ T_0^{(0)} = 0.$

Если длину пути в орграфе определить как сумму продолжительностей, входящих в этот путь работ, то минимальное время (момент) $\dot{T}_j^{(0)}$ свершения каждого события Pj равно длине максимального пути из события P0 в событие Pj. Все работы, выходящие из события Pj, не могут быть начаты ранее, чем будут выполнены работы, принадлежащие этому максимальному пути.

Минимальный момент свершения события-конца проекта называется ***критическим временем проекта***.

По определению критическое время равно длине максимального пути из P0 в Pn. Таким образом, все работы, входящие в проект, не могут быть выполнены за время меньшее, чем критическое время.

Алгоритм вычисления минимального момента свершения события. Из-за того, что СГ правильно пронумерован и не содержит контуров, вычисление длин максимальных путей из события P0 во все остальные события Pj, т. е. величин $\dot{T}_j^{(0)}$ можно осуществлять по формулам:

$$\begin{cases} T_0^{(0)} = 0, \\ T_j^{(0)} = \max\limits_i (T_i^{(0)} + t_{ij}), \qquad j = 1, 2, ..., n, \end{cases}$$

где i – индексы начал всех дуг, входящих в Pj. При применении формул можно для каждого Pj запомнить множество Ij всех тех индексов i, для которых в формуле достигается максимум. Эти множества используются при восстановлении критического пути.

Критический путь – важная характеристика сетевого графика. Работы, лежащие на критическом пути, являются наиболее важными и в этом смысле критическими. Их продолжительность нежелательно увеличивать, так как это приведет к увеличению времени выполнения всего проекта. Работы, не лежащие на критическом пути, – менее важные,

их продолжительность можно увеличить до определенного предела, и это не приведет к увеличению времени выполнения проекта.

Максимальным моментом $T_j^{(1)}$ свершения события Pj называют наиболее позднее время окончания всех работ, входящих в Pj, которое не увеличивает критическое время проекта $T_n^{(0)}$.

Величина $T_j^{(1)}$ определяется по формуле

$$T_j^{(1)} = T_n^{(0)} - d_j, \quad j = 1, 2, ..., n, \qquad \text{- (4.2)}$$

Алгоритм вычисления dj

Величина dj вычисляется по формулам:

$$\begin{cases} d_n = 0, \\ d_j = \max\limits_k (d_k + t_{jk}), \quad j = n-1, n-2, ..., 0, \end{cases} \qquad (4.3)$$

где k – индексы концов всех дуг, выходящих из Pj.

После вычисления всех значений dj, вычисляем максимальные моменты $T_j^{(1)}$ по формуле (4.2).

Теорема 4.1. Для того чтобы событие Pj принадлежало критическому пути, необходимо и достаточно, чтобы выполнялось условие

$$T_j^{(0)} = T_j^{(1)}.$$

Теорема 4.1 дает второй способ восстановления критического пути.

3. Параметры СГ: свободный и полный резервы времени, свойства.

Важными временными параметрами сетевого графика являются так называемые свободный и полный резервы времени.

Свободным резервом sij времени работы (Pi, Pj) называется величина

$$s_{ij} = T_j^{(0)} - T_i^{(0)} - t_{ij}.$$

Величина sij для (Pi, Pj) означает отрезок времени, на который можно увеличить время выполнения работы или опоздать с ее началом так, чтобы не изменить минимальные моменты совершения событий, следующих за данной работой.

Полным резервом pij времени работы (Pi, Pj) называется величина

$$p_{ij} = T_j^{(1)} - T_i^{(0)} - t_{ij}.$$

Величина pij для (Pi, Pj) означает отрезок времени, на который можно увеличить время выполнения работы или опоздать с ее началом так, чтобы не увеличилось критическое время.

Теорема 4.2. Для того чтобы работа (Pi, Pj) принадлежала критическому пути, необходимо и достаточно, чтобы выполнялось условие

$$p_{ij} = 0.$$

Доказательство. Если (Pi, Pj) принадлежит критическому пути, это означает, что

$$T_n^{(0)} = T_i^{(0)} + t_{ij} + d_j.$$

Учитывая формулу (4.2), $d_j = T_n^{(0)} - T_j^{(1)}$, и подставляя его в предыдущее выражение, получим

$$T_n^{(0)} = T_i^{(0)} + t_{ij} + T_n^{(0)} - T_j^{(1)},$$

следовательно:

$$T_j^{(1)} - T_i^{(0)} - t_{ij} = 0.$$

Проводя это же рассуждение с конца, можно показать и достаточность теоремы.

Для свободных резервов времени такое утверждение не верно, т. е. могут быть работы с нулевым свободным резервом, однако они не принадлежат критическому пути. Эта теорема сразу позволяет находить

все критические работы. Значение pij является мерой критичности работы: чем ближе оно к 0, тем более критичной являются работа (Pi, Pj).

Особенно важно знание полного резерва времени любых работ для случая, когда указан директивный срок $T_{\text{дир}}$ выполнения проекта, причем Тдир < Ткрит. Тогда возникает вопрос, какие из работ в первую очередь необходимо сократить по продолжительности, добавив им соответствующие ресурсы? Понятно, что это те, у которых pij = 0.

4. Линейная диаграмма, основные параметры.

Сетевые графики наглядно отражают технологические взаимосвязи между работами. Однако при их использовании трудно ответить, например, на вопрос, сколько и какие работы выполняются в каждый момент времени. Для ответа на такие вопросы используются представления СГ в виде ***линейных диаграмм*** (ЛД, диаграммы Ганнта или временные диаграммы).

Правило построения ЛД для правильно пронумерованных СГ: необходимо задать горизонтальную ось времени с нанесенной на ней градацией в выбранном масштабе.

Работы изображаются в виде линий-отрезков, параллельных этой оси, причем длина каждого отрезка соответствует продолжительности работы. Отрезки, соответствующие работам (P_{i_0}, P_j) т. е. с одинаковым началом Pi0, изображаются друг над другом по возрастанию индексов j. Все работы, выходящие из P0, начинаются от условной нулевой вертикали. Работы вида (Pi, Pj) откладываются от условной вертикали, которая проходит через самый правый конец работ вида (Pk, Pi), отложенных на диаграмме ранее.

//в книге идет разбор примера, выпишу основную хуйню

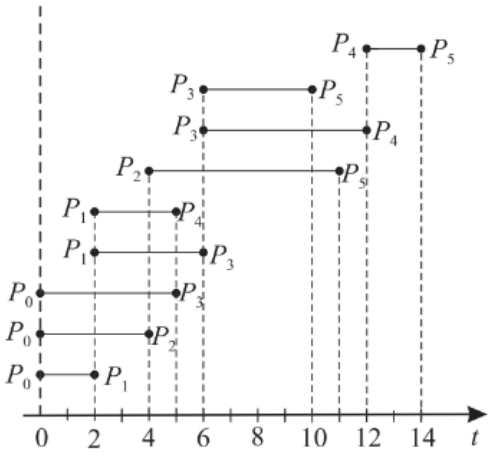


Рис. 4.9

Минимальный момент совершения каждого события Pj будет соответствовать самой правой проекции концов работ вида (Pi, Pj).

Таким образом, сразу же определяется ***критическое время*** проекта как самая правая проекция концов работ, входящих в проект.

Для построения ***критического пути*** на ЛД (см. рис. 4.9) достаточно выделить работу (горизонтальные линии на рис. 4.9), которая оканчивается в момент, соответствующий самой правой проекции события Pn. Далее,

идя к началу этой работы и спускаясь по вертикали вниз, выделяют работу, конец которой совпадает с выделенным началом и лежит на данной вертикали. Затем вновь переходят к началу такой работы и спускаются вниз по вертикали, чтобы найти конец очередной работы и т. д., пока не дойдут до нулевой вертикали. Если на какой-то из вертикалей имеется несколько работ, концы которых совпадают с выделенным началом, то появляются варианты построения нескольких критических путей.

Свободный резерв времени каждой работы на ЛД (см. рис. 4.9) определяется как величина, на которую можно сдвинуть вправо данную работу, не сдвигая все следующие за ней работы.

Для определения ***максимального момента*** $T_j^{(1)}$ свершения события отрезки-работы, входящие в событие Pj и следующие после него, сдвигают вправо, насколько это возможно без увеличения критического времени. Тогда самый левый конец Pj после такого сдвига и определит на оси времени момент $T_j^{(1)}$

На полученной после подобных сдвигов линейной диаграмме ***полный резерв времени*** pij любой работы (Pi, Pj) определяется как величина ее сдвига, т. е. длина промежутка от старого положения Pj до нового положения Pj. Естественно, что для критических работ pij = 0.

Несмотря на трудоемкость построения ЛД (особенно при большом количестве работ) они часто используются из-за наглядности и простоты при ответе на вопросы, сколько и какие работы выполняются в каждый момент. Такая возможность существенна при анализе использования разных видов ресурсов, необходимых при реализации проекта.

5. Общая задача теории расписаний.

Одним из важнейших средств эффективного выполнения любого рода деятельности является планирование во времени, т. е. использование расписаний как некоторой совокупности указаний: какие именно операции какими именно исполнителями будут выполняться в каждый момент времени. Как отмечено ранее, весьма полезным средством планирования во времени являются сетевые графики и временные диаграммы. Однако эти модели применимы только в случаях, когда заранее установлена очередность выполнения всех запланированных работ. В исследовании операций изучаются также модели, когда предварительно требуется определить очередность выполнения работ с учетом возможностей исполнителей. Ограничения на возможности изменения очередности и выбор исполнителей обусловливаются различными (в том числе и экономическими) требованиями и существенно затрудняют задачу составления наилучшего (оптимального) расписания. Такого рода проблемы являются предметом изучения в специальном разделе исследования операций – ***теории расписаний*** (ТР).

В ***теории расписаний*** изучаются системы, обслуживающие некоторое множество требований для случая, когда все требования готовы к обслуживанию, и оно, как правило, состоит из нескольких последовательных операций. Общепринято модели формулировать на языке «заявки-приборы» или даже «детали-станки».

Далее рассмотрим следующий, достаточно общий пример ***задачи теории расписаний***. Пусть имеется m приборов, на которых требуется обслужить n заявок, причем считается, что все эти заявки уже поступили в систему. Для каждой i-й заявки задана последовательность $U^{(i)} = (u_1^{(i)}, ..., u_{m_i}^{(i)}), \quad i = 1, 2, ..., n,$ которая определяет технологический маршрут обслуживания этой заявки таким образом, что $u_j^{(i)}$ означает номер прибора, обслуживающего i-ю заявку на j-м шаге ее технологического маршрута. Последовательность таких номеров будет определять порядок перехода заявки при обслуживании от прибора к прибору. В каждой последовательности $U^{(i)}, i = 1, 2, ..., n,$ в общем случае допускается повторение одинаковых номеров. Также не ограничивается значение mi, т. е. число шагов обслуживания i-й заявки. Например, может быть маршрут U (i)= (1, 2, 3, 2, 1, 4, 1). В соответствии с технологическим маршрутом U (i) также заданы значения tij (i = 1, 2, ..., n; j = 1, 2, ..., mi) –

положительные рациональные числа, означающие продолжительность обслуживания i-й заявки на каждом j-м шаге ее технологического маршрута. Не уменьшая общности, можно считать, что tij – целые числа. Будем рассматривать модели, в которых процесс обслуживания подчиняется следующим условиям:

1) ни одна заявка не может одновременно обслуживаться несколькими приборами;

2) если обслуживание заявки начато, то оно должно быть доведено до конца на каждом шаге без перерыва в обслуживании;

3) время перехода заявки от одного прибора к другому не учитывается;

4) ни один прибор не может одновременно обслуживать более чем одну заявку;

5) время настройки (наладки) приборов (т. е. подготовки к обслуживанию следующей в очереди заявки) не учитывается.

Задача состоит в том, чтобы при заданных маршрутах U (i) для i = 1, 2, ..., n и, соответственно, времен обслуживания tij в условиях 1) – 5) найти такие моменты начала обслуживания каждой заявки на любом приборе, при которых общее время обслуживания всех заявок будет минимальным.

Пусть задача ТР задана в виде следующей таблицы:

Заявки	Технологические маршруты	Время обслуживания
А	(1, 2, 4, 2)	3, 2, 1, 2
Б	(1, 2, 3)	1, 3, 3
В	(2, 3, 2, 4)	2, 3, 3, 2
Г	(4, 5, 3)	4, 3, 3
Д	(3, 1, 4, 5)	3, 4, 2, 2
Е	(4, 5, 4, 1, 2)	2, 2, 2, 2, 2
Ж	(5, 1, 5, 4)	2, 1, 2, 1
З	(3, 1)	1,3
К	(5)	3

Замечание. Сформулированная задача может быть сведена к задаче линейного программирования с логическими условиями, которая, в свою очередь, сводится к задаче линейного целочисленного программирования. Однако даже для небольших исходных моделей соответствующая задача целочисленного программирования содержит большое число ограничений и неизвестных, что существенно затрудняет ее решение.

6. Задача Беллмана – Джонсона, алгоритм для 2-х приборов.

Предположим, что к условиям сформулированной задачи ТР добавлены еще два условия:

6) технологический маршрут для всех заявок одинаков;

7) последовательность поступления заявок на обслуживание на всех приборах одинакова.

Как и в предыдущей модели, задача состоит в том, чтобы для заданного технологического маршрута и соответствующего времени обслуживания заявок в условиях 1) – 7) найти такие моменты начала обслуживания каждой заявки на любом приборе, при которых общее время обслуживания всех заявок будет минимальным.

Впервые эту задачу исследовали математики Р. Беллман и С. М. Джонсон (задача Б – Д), относя ее к задачам так называемого поточного типа (flow shop).

Не уменьшая общности, условие 6) можно сформулировать в виде 6'), технологический маршрут для всех заявок одинаков и имеет вид

$$U^{(i)} = (1, 2, ..., m), \quad i = 1, 2, ..., n.$$

Теорема 5.1. В оптимальном расписании для задачи Б – Д с двумя приборами заявка i обслуживается ранее заявки j, если выполняется неравенство $\min(t_{i1},\ t_{j2}) \le \min(t_{i2},\ t_{j1})$.

Неравенство обладает свойством транзитивности, т. е. если оно выполняется для пар i, j и j, k, то оно будет выполняться и для пары i, k. Это доказывается непосредственным анализом трех неравенств вида и возможных соотношений между входящими под знак min числами ti1, ti2, tj1, tj2, tk1, tk2. Свойство транзитивности позволяет строить хотя бы одно оптимальное расписание для задачи Б – Д в случае, когда m = 2.

Алгоритм Джонсона

Шаг 1. Составляют таблицу времен обслуживания заявок, состоящая из n строк и двух столбцов. Одновременно готовится строка из n пустых позиций.

Шаг 2. В таблице находят минимальный элемент. Если он принадлежит первому столбцу, то номер его строки записывают в первую слева свободную позицию подготовленной строки. Если этот минимум принадлежит второму столбцу, то номер его строки записывают в первую справа свободную позицию. В любом случае строка, где был минимум, вычеркивается из таблицы времен обслуживания. Шаг 2 повторяется, пока

не заполнятся все пустые позиции подготовленной строки. Этим самым определится оптимальное расписание. Если потребуется выбор из нескольких одинаковых минимумов, то можно выбирать любой из них. При выборе иного минимума может получиться иное оптимальное расписание.

Данный алгоритм, разработанный в 1954 г., является эффективным по числу операций. Для задачи Б - Д при $m > 2$ в общем случае нет эффективных алгоритмов, существенно уменьшающих полный перебор всех возможных перестановок-расписаний.

7. Задача коммивояжера.

Еще одной важной для ИСО задачей комбинаторного типа является ***задача коммивояжера***.

Пусть имеется n пунктов, пронумерованных в некотором порядке: 1, 2, ..., n. Для каждой пары (i, j) этих пунктов известно расстояние cij. Термин «расстояние» может означать физическое расстояние между этими пунктами, время перехода из пункта i в пункт j, стоимость такого перехода или количество затраченного горючего. Значения cij образуют квадратную матрицу $C = [c_{ij}]_{n\times n}$, у которой элементы на главной диагонали не определены и, например, обозначены символом ∞. В этой матрице не обязательно cij = cji, т. е. она может быть несимметричной.

Коммивояжер, начиная от некоторого начального пункта, например пункта 1, должен обойти все остальные, побывав в каждом пункте по одному разу таким образом, чтобы его маршрут был оптимален по какому-либо критерию. Чаще всего оценкой маршрута является его длина как сумма расстояний непосредственных переходов в выбранном маршруте. Коммивояжеру нужно выбрать маршрут наименьшей длины.

Построим математическую модель задачи коммивояжера. Через xij обозначим переменные, такие, что xij = 1, если коммивояжер включил в свой маршрут переход от пункта i до пункта j, и xij = 0 - в противном случае. В задаче необходимо определить значения xij, при которых функция

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

принимает минимальное значение и для которых выполняются условия:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, ..., n; \tag{5.5}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, ..., n; \tag{5.6}$$

$$x_{ij} \in \{0,1\}, \quad i, j = 1, 2, ..., n, \quad i \neq j; \tag{5.7}$$

множество x_{ij} должно определять замкнутый маршрут. (5.8)

Условие (5.5) говорит о том, что коммивояжер обязан переходить из пункта i только в один из остальных пунктов. Условие (5.6) требует, чтобы в каждый пункт j переход осуществлялся только из одного пункта.
Целевая функция (5.4) и ограничения (5.5)-(5.7) в точности совпадают с моделью классической задачи о назначениях (п. 3.6). Проанализируем детально сущность дополнительного условия (5.8).
Условие (5.8) в задаче коммивояжера приводит к тому, что не каждая подстановка определяет замкнутый маршрут коммивояжера.
Условия (5.8) можно записать в виде формальных неравенств, например, путем введения дополнительных целочисленных переменных, связанных с пунктами 1, 2, ..., n – 1. В итоге задача коммивояжера становится задачей линейного целочисленного программирования. Однако этот факт редко используется для нахождения решения из-за больших размеров получаемой задачи.
Задача коммивояжера имеет большое практическое применение не только сама по себе. К ней ***сводятся*** многие другие задачи оптимального упорядочения, в частности некоторые задачи теории расписаний.

- Одностадийная задача обслуживания заявок.

Задача состоит в том, чтобы найти такую очередность обслуживания всех заявок, при которой суммарное время обслуживания будет минимальным.

- Многостадийная задача без задержек в обслуживании заявок

Пусть, кроме условий 1) - 7), в задаче Б - Д дополнительно выполняется еще одно условие:
8) для каждой заявки момент начала ее обслуживания на очередном приборе совпадает с моментом окончания ее обслуживания на предыдущем приборе.
Требуется в условиях обслуживания 1) – 8) найти расписание-перестановку, минимизирующую общее время обслуживания всех заявок.
Замечание. Интересно, что при m = 2 (двух приборах в обслуживающей системе) в сформулированной задаче без задержек в обслуживании соответствующая задача коммивояжера решается алгоритмом, сравнимым по эффективности с алгоритмом Джонсона для решения задачи Б - Д.

8. Общая схема метода ветвей и границ.

Пусть требуется найти минимальное значение целевой функции f(X) на конечном множестве W допустимых решений X, т. е. найти

$$\min_{X\in\Omega} f(X).$$

Использование ***метода ветвей и границ*** при решении конкретной комбинаторной задачи состоит из нескольких основных ***этапов***.

1. ***Ветвление.*** Должно существовать правило разбиения множества W на подмножества типа $\Omega_1^{(1)}, \ldots, \Omega_k^{(1)}$ так, чтобы $\bigcup_{i=1}^{k}\Omega_i^{(1)} = \Omega.$

Допускается пересечение множеств, получаемых в результате разбиения, после которого необходимо, чтобы правило ветвления позволило ветвить каждое из подмножеств $\Omega_i^{(1)}$ потом $\Omega_j^{(2)}$ и т. д. Таким образом, в результате ветвления должно строиться (явно или нет) дерево ветвлений, представленное на рис. 5.5, и правило разбиения должно быть применимо к каждой висячей вершине дерева, соответствующей подмножеству с более чем одним элементом.

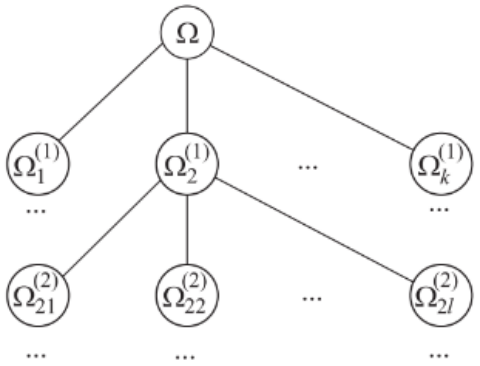


Рис. 5.5

2. ***Вычисление нижней границы целевой функции***. Для каждого подмножества Ωk, получаемого в результате ветвления, необходимо уметь вычислить нижнюю границу целевой функции на этом подмножестве, т. е. число ξ(Ωk) такое, что

$$f(X) \geq \xi(\Omega_k) \text{ для любых } X \in \Omega_k.$$

3. ***Пересчет оценок.*** Если для двух множеств Ω1 и Ω2 выполняется включение Ω1 ⊂ Ω2, то для целевой функции будет выполняться соотношение

$$\min_{X\in\Omega_1} f(X) \geq \min_{X\in\Omega_2} f(X).$$

Можно считать, что при каждом разбиении множества Ω′ на подмножества $\Omega'_1, \Omega'_2, ..., \Omega'_k$ способ вычисления нижних границ подобран так, что $\xi(\Omega'_i) \geq \xi(\Omega'),$ т. е. при ветвлении нижняя граница не убывает.

4. ***Построение допустимых решений.*** Если на дереве ветвлений найдена вершина, которую нельзя больше разветвить, то надо уметь по единственной цепи от корня дерева до этой вершины построить соответствующее допустимое решение из Ω. Конкретный способ разрабатывается в зависимости от специфики задачи.

5. ***Признак оптимальности.*** Для каждой конкретной задачи должен применяться следующий принцип: если на каком-то шаге ветвления найдено решение X и при этом

$$f(\bar{X}) = \xi(\Omega_i) = \min(\Omega_k),$$

где минимум берется по всем висячим вершинам дерева ветвлений, то это решение является оптимальным: решения, для которого значение целевой функции ниже минимальной нижней границы, не может быть.

6. ***Способы уменьшения возможных вариантов ветвления.*** Основной прием состоит в том, что некоторым способом находится хотя бы одно решение из Ω. Значение целевой функции для него объявляют рекордом и на дереве не рассматриваются (не ветвятся) все те висячие вершины, нижняя граница которых выше значения $f(X_{\text{рек}}).$

7. ***Возможность останова вычислений.*** Часто либо из-за конструкции метода, либо по необходимости вычисления останавливают. В таком случае последнее достигнутое допустимое решение $\hat{X}$ со значением целевой функции $f(\hat{X})$ и наименьшей границей $\hat{\xi}$ среди всех висячих вершин дерева позволяет оценить относительную ошибку $(f(\hat{X}) - \hat{\xi})/\hat{\xi}$ недостижения оптимального решения.

В целом метод ветвей и границ - это не один специальный алгоритм, а очень широкий класс алгоритмов. Эффективность применения этого метода зависит от разработки стратегии для конкретно рассматриваемой задачи.

9. Алгоритм Литтла для задачи коммивояжера.

Описанный ниже метод основан на идеях, предложенных Литтлом и др. в 1963 г.

Имеется n пунктов (городов). Для каждой пары городов i и j ($i \neq j$) известно расстояние $c\_(ij) \geq 0$ между ними. Коммивояжер, находясь в одном из городов, должен посетить остальные города, побывав в них по одному разу, и вернуться в начальный город так, чтобы суммарное пройденное расстояние было минимальным.

Далее будем работать с матрицей расстояний $C = [c\_(ij)]\_(n \times n)$, причем несуществующие или запрещенные переходы между городами будем обозначать символом ∞. Вначале все $c\_(ii) = \infty$.

Ветвление каждого множества возможных маршрутов коммивояжера на два подмножества состоит во включении или невключении в эти подмножества непосредственного перехода от какого-то города i до города j. Соответствующие подмножества будем обозначать на дереве ветвлений с корнем G в виде пары (i, j) или пары (i, j) в кружках - вершинах дерева (рис. 5.6). Множество G обозначает множество всех возможных маршрутов коммивояжера ($(n - 1)!$ маршрутов).

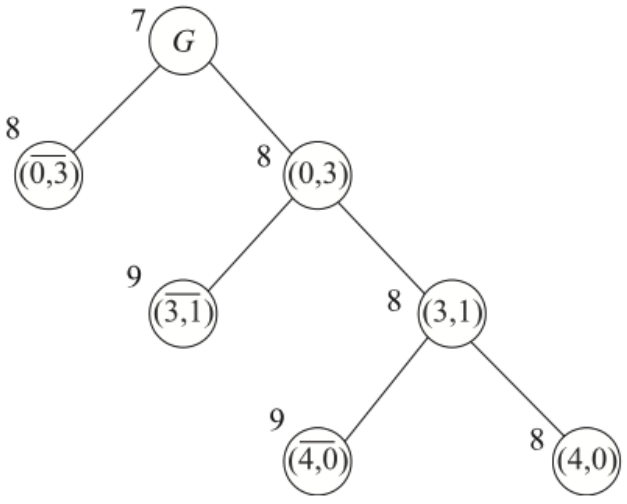


Рис. 5.6

Каждому ветвлению сопутствуют оценки снизу длины маршрутов, входящих в соответствующее множество. Такие оценки чаще всего вычисляются с помощью операции приведения квадратной матрицы

расстояний, соответствующей оцениваемому множеству маршрутов. Для любой матрицы расстояний эта операция состоит из двух шагов. На шаге первом из элементов каждой *k*-й строки вычитаем минимальный элемент *h′*_k этой строки. На втором шаге от элементов остатков (после первого шага) каждого *j*-го столбца отнимаем минимальный элемент *h″* этого столбца.

Каждой вершине дерева ветвлений, как множеству маршрутов, соответствует определенная приведенная матрица расстояний. При ветвлении такой вершины - множества маршрутов, в которых запрещается непосредственный переход от города *i* до города *j*, - в этой матрице элемент *c_ij* заменяют знаком ∞. Множеству маршрутов с разрешенным переходом от *i* до *j* будет соответствовать вышеуказанная приведенная матрица без *i*-й сроки и *j*-го столбца. В такой уменьшенной матрице, с сохранением исходных но- меров строк и столбцов, необходимо запретить непосредственный переход от города *j* до города *i*, полагая *c_ji* = ∞. Кроме того, учитывая, что разрешение непосредственно- го перехода от *i* к *j* вместе с ранее разрешенными переходами на соответствующей це- почке от корня до анализируемой вершины на дереве ветвлений может образовывать определенный подмаршрут, необходимо запретить непосредственный переход от кон- ца этого подмаршрута до его начала, снова используя символ ∞.

Для полученных в результате такого ветвления вершин и двух соответствующих матриц осуществляем операции приведения и вычисляем нижние границы длин маршрутов. При этом коэффициент приведения для каждой матрицы складываем с нижней границей ветвимой вершины и тем самым получаем нижнюю границу длин маршру- тов новых вершин. Такой способ вычисления нижних границ обеспечивает их неубы- вание по мере дальнейшего ветвления.

Процесс ветвления и оценок продолжаем до тех пор, пока не получим висячую вершину, соответствующую матрице 2×2. Единственная цепь вершин дерева от его корня до этой висячей вершины вместе с возможными переходами в итоговой матрице 2×2 определит один маршрут коммивояжера. Если при этом длина полученного маршрута будет равна нижней границе висячей вершины и эта нижняя граница будет не большей, чем нижние границы остальных висячих вершин дерева, то тем самым

выполняется критерий оптимальности и найдено решение задачи коммивояжера. Если же критерий оптимальности не выполняется, то ветвим какую-либо иную висячую вершину дерева с наименьшей нижней границей.

Алгоритм из чата:

Вот шаги алгоритма Литтла:

1. Составьте матрицу стоимостей, где каждый элемент матрицы указывает стоимость перемещения от одного города к другому.
2. Найдите минимальное значение в каждой строке матрицы, и вычтите его из всех элементов этой строки. Это позволит найти минимальную стоимость перемещения из каждого города.
3. Найдите минимальное значение в каждом столбце оставшейся матрицы (после вычитания минимальных значений строк), и вычтите его из всех элементов этого столбца. Это позволит найти минимальную стоимость перемещения в каждый город.
4. Выберите любой город в качестве начального.
5. Найдите маршрут, начиная с выбранного города, следующий следующему правилу: выберите минимальную стоимость перемещения из текущего города в доступный непосещенный город.
6. Повторяйте шаг 5, пока не будет пройден каждый город.
7. Закончите маршрут, возвращаясь в начальный город.
8. Добавьте стоимость перемещения от последнего города в маршруте к начальному городу.
9. Полученный маршрут будет приближенным решением задачи коммивояжера.

Важно отметить, что алгоритм Литтла не гарантирует нахождение оптимального решения для задачи коммивояжера, но обычно даёт достаточно хорошие результаты.

10.Общая задача ТМО.

Общая задача транспортного моделирования и оптимизации (ТМО) состоит в определении оптимального плана перемещения ресурсов (товаров, грузов, информации и т. д.) из источников до потребителей с учетом различных ограничений и целей.

ТМО включает в себя ряд подзадач, таких как задача линейного программирования (ЛП), задача сетевого потока, задача динамического программирования и другие. Некоторые из основных компонентов и задач ТМО включают:

- Задача оптимального плана производства: Определение оптимального количества и типа продукции, которую следует произвести в различных предприятиях или заводах, чтобы удовлетворить потребности рынка и минимизировать затраты.
- Задача транспортной логистики: Оптимизация маршрутов и расписания доставки товаров из одного места в другое, учитывая ограничения на доступность ресурсов, временные ограничения и минимизацию затрат на транспортировку.
- Задача инвентаризации: Определение оптимального уровня запасов товаров на складе или в магазине, учитывая потребности клиентов, стоимость хранения и заказа.
- Задача маршрутизации транспорта: Определение оптимальных маршрутов для транспортных средств, учитывая ограничения на время, вместимость и другие факторы, чтобы минимизировать расходы на топливо или время доставки.
- Задача планирования проектов: Определение оптимального расписания выполнения задач в проекте, учитывая зависимости между задачами, ограничения на ресурсы и минимизацию продолжительности проекта.

Общая задача ТМО состоит в нахождении оптимального решения, которое удовлетворяет заданным ограничениям и целям, таким как минимизация затрат, максимизация прибыли или оптимальное использование ресурсов. Для её решения применяются различные методы и алгоритмы оптимизации, такие как линейное программирование, симплекс-метод, сетевой анализ, генетические алгоритмы и другие.

Классическая ТЗ состоит в следующем. Имеется n поставщиков A_1, A_2, ..., A_n определенного однородного товара, которые предлагают соответственно a_1, a_2, ..., a_n единиц этого товара m потребителям B_1,

B_2, ..., *B_m*, создающим спрос на него в количествах *b_1*, *b_2*, ..., *b_m* единиц соответственно. Для каждой пары «*i*-й поставщик - *j*-й потребитель» известна стоимость *c_ij* (*i* = 1, 2, ..., *n*; *j* = 1, 2, ..., *m*) перевозки единицы товара. Требуется определить такой план удовлетворения спроса потребителей, при котором общие транспортные затраты минимальны. Очень часто на практике предполагается, что такие затраты являются суммой затрат на перевозку товара для каждой пары «*i*-й поставщик – *j*-й потребитель», которые, в свою очередь, прямо пропорциональны с коэффициентом *c_ij* количеству перевозимого товара. Таким образом, выполняются свойства аддитивности и прямой пропорциональности

11. Простейший поток заявок, основные свойства.

Достаточно хорошо изучены СМО (Системы массового обслуживания), в которых входной поток является случайной величиной и обладает следующими тремя свойствами.

1. Поток стационарный, т. е. вероятность попадания одного и того же числа заявок на какой-то участок времени длиной *t* зависит только от длины этого участка и не за- висит от того, в каком месте на оси времени он расположен. Стационарность потока означает его однородность во времени. В частности, в данном потоке будет постоян- ной такая важная характеристика, как интенсивность поступления заявок, – среднее число заявок, поступающих в единицу времени. Условие стационарности потока равносильно требованию постоянства интенсивности потока и неизменности во времени всех его вероятностных характеристик.

Для реальных СМО условие стационарности потока заявок не является достаточно жестким ограничением. Практически всегда можно выбрать интервалы времени, на которых поток заявок стационарный.

2. Поток заявок должен быть потоком без последействия (без памяти), т. е. для каждых непересекающихся интервалов времени число заявок, попадающих на один из них, не должно зависеть от того, сколько заявок попало на другой участок. Таким образом, заявки, подчиняющиеся этому условию, должны образовывать поток независимо друг от друга. Примером тому служит поток пассажиров на станции.

3. Поток заявок является ординарным, т. е. вероятность попадания на элементарный достаточно малый участок времени более одной заявки должна быть близкой к нулю. Таким образом, это условие предполагает, что в некоторый достаточно короткий промежуток времени, близкий по длине к нулевому, одновременно не может появляться более чем одна заявка.

Существуют реальные задачи, когда это требование не выполняется. В очередь на разгрузку, например, сразу поступает несколько вагонов.

Входные потоки заявок, удовлетворяющие условиям 1) – 3), называются **простейшими**.

Критическое изучение перечисленных свойств простейшего потока приводит к заключению, что данные потоки не очень часто могут встречаться в практических задачах. Однако показано, что существует большой круг задач, в которых можно ограничиться изучением простейших входных потоков. Например, А. Я. Хинчин доказал, что если какой-то достаточно мощный поток состоит из большого числа стационарных и ординарных потоков и каждый из них несравним по мощности с суммарным потоком, то весь суммарный поток будет простейшим.

Простейший поток заявок связан с распределением Пуассона.

При изучении простейшего потока важно выяснить распределения интервалов времени T между моментами поступления двух произвольных соседних заявок. Для этого достаточно найти закон распределения случайной величины T, т. е. функцию $F(t) = P(T < t)$ для любого t.

Условие $T < t$ означает, что на интервал от какого-то начального t_0 до t попадает хотя бы одна заявка. Используя формулу для P_m, можно вычислить, когда на интервал T не попадет ни одной заявки. Эта величина будет равна

$$P(T > t) = P_0 = e^{-\lambda t} \ (t \geq 0).$$

В случае когда на интервал T попадает хотя бы одна заявка, будем иметь

$$F(t) = P(T < t) = 1 - P_0 = 1 - e^{-a} = 1 - e^{-\lambda t}.$$

Для этого закона распределения можем найти плотность распределения

$$f(t) = [F(t)]' = \lambda e^{-\lambda t}.$$

Таким образом, промежуток времени T между моментами поступления двух заявок в простейшем потоке подчиняется закону распределения $F(t)$ с плотностью $f(t)$. Закон этого вида называется показательным (экспоненциальным).

12.Процессы гибели и размножения.

Рассмотрим специальный класс СМО, в котором в систему случайным образом может поступать заявка (процесс рождения), случайным образом она может быть обслуженной и уходить из системы (процесс гибели). В любой момент времени состояния системы могут различаться по количеству находящихся в ней заявок.

Пусть $S0$ – состояние, при котором в системе нет заявок; $S1$ – в системе одна заяв- ка; $S2$ – две заявки и т. д. $Sk(t)$ есть случайное событие, означающее, что в момент вре- мени t система находится в состоянии Sk, $k = 0, 1, 2, ...$ Если в некоторый момент $t0$ в систему прибывает новая заявка, то состояние системы изменяется на единицу в сторону увеличения, и такое изменение будем обозначать

$$S_k(t_0 - 0) \to S_{k+1}(t_0 + 0),$$

где $t_0 - 0$ и $t_0 + 1$ – один и тот же момент t_0; –0 – момент непосредственно перед t_0; +0 – момент, следующий непосредственно за t_0. Если из системы в момент t_0 заявка уходит, то такое изменение обозначим через

$$S_k(t_0 - 0) \to S_{k-1}(t_0 + 0).$$

Для решения описанной задачи можно вывести систему дифференциальных урав- нений, которым подчиняются неизвестные $Pk(t)$ для всех $k = 0, 1, ...$ Начнем с вывода дифференциального уравнения, которому должна удовлетворять вероятность $P0(t)$. Для этого надо рассмотреть всевозможные варианты, при которых система окажется в состоянии $S0$ в момент $t + \tau$, где τ – некоторый достаточно малый интервал времени.

Таких вариантов всего два:

1) может быть, что в момент t система находилась в состоянии S_0 и за время τ ни одной заявки не поступило, т. е.

$$S_0(t) \to S_0(t + \tau);$$

2) в состояние S_0 система может перейти из состояния S_1, т. е.

$$S_1(t) \to S_0(t + \tau).$$

Теорема 6.2. Если число состояний системы конечно и из любого состояния можно перейти за конечное число шагов в любое другое состояние, то предельные вероятности состояний существуют и не зависят от начального состояния системы.

13.Формулы вероятности состояний.

В учебнике ничего нету, вот только что нашлось

Модель гибели и размножения Чепмена-Колмогорова (также известная как стохастическая модель гибели и размножения) используется для моделирования дискретных процессов гибели и размножения с непрерывным временем. Она позволяет описать изменение численности популяции или другой системы с учетом вероятностей гибели и размножения.

В этой модели предполагается, что изменение численности происходит в непрерывном времени. Пусть p(n, t) - вероятность того, что численность популяции (или другой системы) будет равна n в момент времени t. Функция p(n, t) удовлетворяет следующему дифференциальному уравнению:

$$dp(n, t)/dt = -\mu(n)p(n, t) + \sum[k=1 \text{ до бесконечности}] \lambda(k)p(n-k, t)$$

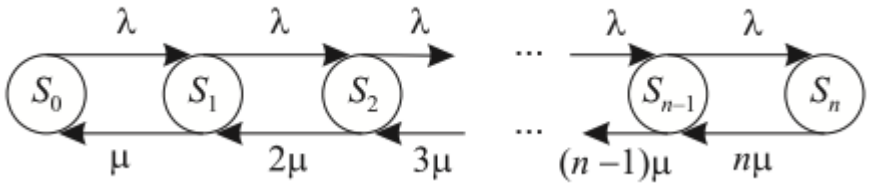
где μ(n) - интенсивность гибели (вероятность гибели единицы популяции), λ(k) - интенсивность размножения (вероятность размножения k потомков), и сумма берется по всем возможным значениям k (число потомков).

Данное уравнение описывает изменение вероятности p(n, t) с течением времени. Первый член в правой части уравнения (-μ(n)p(n, t)) представляет потери из-за гибели, а второй член (∑[k=1 до бесконечности] λ(k)p(n-k, t)) представляет прирост от размножения потомков. Обратите внимание, что сумма берется по всем значениям k, что позволяет учесть все возможные числа потомков.

Чтобы полностью описать систему, требуется задать начальные условия p(n, 0) - начальные вероятности численности популяции в момент времени t=0.

Решение этого дифференциального уравнения может быть сложной задачей, и для его получения могут применяться различные численные методы. Часто используется численное интегрирование для приближенного вычисления вероятностей состояний на разных временных отрезках.

14.СМО с потерями.



Пусть имеется *n* приборов, на которые поступает простейший поток заявок на обслуживание с интенсивностью λ. Известно, что длительности обслуживания на каждом из приборов подчинены показательному закону распределения с параметром μ, где μ – интенсивность обслуживания. Если при поступлении заявки оказывается, что все приборы заняты, она теряется, уходит из системы необслуженной. Требуется найти:

среднее число заявок, которое система может обслужить за единицу времени (так называемую абсолютную пропускную способность СМО); отношение среднего числа заявок, обслуживаемых системой в единицу времени, к среднему числу поступающих за это время заявок (так называемую относительную пропускную способность СМО).

Состояние системы будем различать по числу занятых приборов:

*S*0 – все приборы свободны;
*S*1 – занят один прибор, остальные свободны;
... ...;

Sn – заняты все *n* приборов.

Не трудно видеть, что принятое предположение описывает частный случай процесса гибели и размножения, и поэтому при определении предельных вероятностей состояния системы, т. е. величин $P_0, P_1, P_2, ..., P_n$,

Очевидно, что *P*отк = *Pn*, так как если система находится в состоянии *Sn*, то заявка не будет обслужена. Вероятность того, что заявка будет принята к обслуживанию *q* = 1 – *P*отк, – относительная пропускная способность системы. Фактически она представляет собой отношение числа обслуженных заявок к числу всех поступивших в систему заявок.

Учитывая, что поток простейший, абсолютная пропускная способность системы, как среднее число заявок, обслуживаемых в единицу времени, равна $A = \lambda q$.

Важной характеристикой СМО с потерями является среднее число занятых приборов или среднее число заявок, занятых в системе. Поскольку один занятый прибор обслуживает в среднем за единицу времени μ заявок, A – среднее число обслуживаемых заявок в единицу времени, то среднее число занятых приборов вычисляется по формуле $k = A/\mu$. Таким образом, формулы Эрланга дают возможность непосредственно μ вычислить все параметры анализируемой СМО с потерями.

15.СМО с конечной очередью.

Пусть в системе имеется *n* приборов и *m* мест для формирования очереди. Предположим, что поток заявок простейший с параметром λ и длительность обслуживания на каждом из приборов подчиняется показательному закону с параметром μ. Поступившая заявка или сразу начинает обслуживаться, или становится в очередь, если все приборы заняты, но имеются свободные места в очереди, или уходит из системы, если в очереди нет свободных мест. Таким образом, состояния системы можно различать по числу связанных с ней заявок:

*S*0 – в системе нет ни одной заявки;
*S*1 – обслуживается одна заявка (остальные приборы свободны);
... ...;
Sn – все приборы заняты;
Sn+1 – *n* заявок находится на обслуживании, одна – в очереди;
... ...;
Sn+*m* – *n* заявок находится на обслуживании, *m* – в очереди.

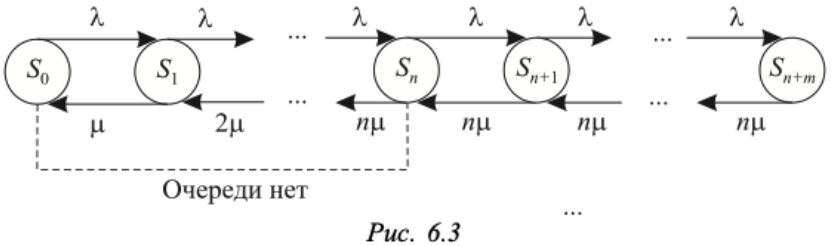


Рис. 6.3

На переходах слева направо проставлены интенсивности входного потока, которые не зависят от того, в каком из состояний находится система. На дугах справа налево проставлены интенсивности длительностей обслуживания, которые прямо пропорциональны количеству занятых приборов.

Из-за того, что по предположению система описывается таким графом, видно, что это также частный случай процесса гибели и размножения.

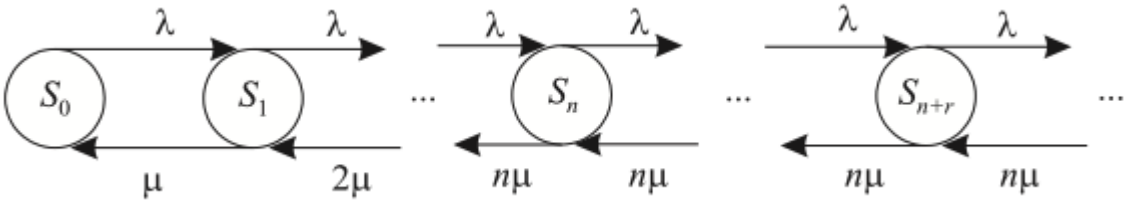
Общее число состояний $(n + m + 1)$, а $\lambda_0 = \lambda_1 = ... = \lambda_{n+m+1} = \lambda$ и

$$\mu_1 = \mu, \mu_2 = 2\mu, ..., \mu_n = n\mu, ..., \mu_{n+m} = n\mu.$$

Таким же способом можно высчитать среднее время ожидания заявки в очереди. Если система находится в состоянии *Sn*, то ей придется ждать в среднем 1/(nμ) ед. времени.

16.СМО с бесконечной очередью.

Предположим, что число мест в очереди не ограничено каким-то числом *m*, а может быть сколь угодно большим. Тогда система, описанная в п. 6.5.1, будет иметь бесконечное число состояний и граф состояний может выглядеть так, как представлено на рис.



В этом случае также имеет смысл рассматривать задачу о нахождении предельных вероятностей состояния системы. Оказывается, их не трудно получить, используя предельные переходы в формулах (6.6). Дело в том, что часть слагаемых для *P*0 в этих формулах представляет собой геометрическую прогрессию со знаменателем λ / (μ / n)

Как и выше, среднее число занятых приборов $\bar{k} = \frac{A}{\mu} = \frac{\lambda}{\mu}.$

17. Игры с природой

В модели принятия решений по управлению сложными системами весьма часто приходится включать параметры, описывающие состояние окружающей среды (например, спрос потребителей, поведение конкурентов, колебание себестоимости продукции и цены в зависимости от стоимости мировых валют, природные факторы и многое другое). Хотя такие параметры существенно влияют на качество принимаемого решения, они являются неконтролируемыми и, в общем, трудными для оценки. В таком случае в исследовании операций задачу по возможности сводят к модели, которая **называется игрой с природой**. В такой игре с одной стороны выступает активный участник-игрок, воздействующий на контролируемые факторы и принимающий решения, с другой стороны на систему влияет окружающая среда, называемая обобщенно природой, а ее действия – это просто возможные состояния, никоим образом не зависящие от поведения изучаемой системы.

Основная модель игры с природой – матричная модель. Пусть игрок имеет m различных решений $\alpha_1, \alpha_2, ..., \alpha_m$ а природа может оказаться в одном из n возможных состояний $\beta_1, \beta_2, ..., \beta_n$. Тогда для каждой пары (α_i, β_i)) должен быть известен некоторый исход. В итоге получим матрицу исходов $[a_{ij}]_{m\times n}$. Очень часто исходы a_{ij} – не числа, а какие-то качественные оценки. В теории принятия решений есть специальное направление - количественное измерение полезности или «построение функций полезности», в котором разрабатываются методы перехода от качественных к количественным оценкам для того, чтобы естественным образом сравнивать, какой из исходов лучше других.

В случае когда величины a_{ij} являются числами, в играх с природой их называют выигрышами, а матрицу исходов – матрицей выигрышей. Используя матрицу выигрышей, игроку необходимо выбрать такое решение (строку матрицы), которое было бы в некотором смысле наилучшим. Для этого можно изобрести бесчисленное множество способов сравнения решений. На практике широко используются «классические» критерии упорядочения решений: Байеса, Лапласа, Вальда, Гурвица, Севиджа.

Критерий Байеса. Предположим, что состояния природы – случайные события, причем вероятности q_j состояний природы заранее известны и $\sum_{j=1}^{n} q_j = 1.$ В этих условиях эффективность B(α_i) каждого решения α_i естественно оценивать как математическое ожидание выигрыша (средний выигрыш), т. е.

$$B(\alpha_i) = \sum_{j=1}^{n} q_j a_{ij}, \quad i = 1, 2, ..., m.$$

Критерий Байеса утверждает, что наилучшим является решение, при котором достигается $\max_i B(\alpha_i).$

Критерий Лапласа. Если ничего неизвестно о возможных вероятностях состояний природы, можно предложить, что каждое из состояний равновероятно, т. е

$q_j = \frac{1}{n}, \quad j = 1, 2, ..., n.$ Тогда математическое ожидание выигрыша для каждого решения α_i равно

$$L(\alpha_i) = \frac{1}{n}\sum_{j=1}^{n} a_{ij}, \quad i = 1, 2, ..., m.$$

Наилучшим считается решение, соответствующее $\max_i L(\alpha_i).$

Критерий Вальда. Для каждого решения α_i находится величина

$$W(\alpha_i) = \min_j a_{ij}, \quad i = 1, 2, ..., m.$$

Наилучшим считается решение, при котором достигается $\max_i W(\alpha_i).$ Кстати, эта величина (называемая максимином) является гарантированным выигрышем при любом состоянии природы. Соответствующее управляющее решение называется гарантированным

Критерий Гурвица. Для каждого решения α_i при заданной величине v, 0 ≤ v ≤ 1, вычисляется значение

$$H(\alpha_i) = v \min_j a_{ij} + (1 - v)\max_j a_{ij}, \quad i = 1, 2, ..., m.$$

Наилучшим считается решение, при котором достигается $\max_i H(\alpha_i).$
Коэффициент v – своеобразный показатель оптимизма игрока. Если v = 1, получаем максиминный критерий Вальда (критерий крайнего пессимизма). При v = 0 получаем критерий крайнего оптимизма.

Критерий Севиджа. Вычисляются величины rij риска потерь игрока при выборе им решения α_i и состояния природы βj

$$r_{ij} = \max_i a_{ij} - a_{ij}.$$

На матрице $[r_{ij}]_{m\times n}$ находятся величины

$$S(\alpha_i) = \max_j r_{ij}, \quad i = 1, 2, ..., m.$$

Наилучшим является решение, при котором достигается $\min_i S(\alpha_i).$

18. Игры с седловой точкой

Пусть $[a_{ij}]_{m\times n}$ – матрица выигрышей в конечной антагонистической игре. Последовательно перебирая стратегии A1, A2, ..., Am, найдем среди них наилучшую для стороны А. Игрок А должен при этом учитывать, что игрок В будет отвечать лучшей для себя стратегией. Поэтому при выборе конкретной стратегии Ai игрок А может заведомо рассчитывать на свой выигрыш $\alpha_i = \min_j a_{ij}$.

Действуя осторожно, игрок А может выбрать ту из стратегий Ai , где достигается выигрыш $\alpha = \max_i \min_j a_{ij}$.

Значение α - это нижняя цена игры. Стратегия, соответствующая α, называется максиминной. Если придерживаться этой стратегии, то игроку А гарантируется выигрыш, равный α. Игрок В заинтересован, чтобы обратить в минимум выигрыш игрока А. Следовательно, перебирая свои стратегии B1, …, Bn, он, учитывая разумное поведение игрока А, должен для каждой из стратегий выделять максимальное значение выигрыша игрока А и минимизировать этот выигрыш. Таким образом, сторона В может гарантировать себе проигрыш не более чем $\beta = \min_j \max_i a_{ij}$.

Значение β - верхняя цена игры. Придерживаясь этой осторожной для себя (минимаксной) стратегии, игрок В гарантирует себе проигрыш не больший чем β.

Существуют матричные игры, у которых верхняя цена игры совпадает с нижней. **Такие игры называются играми с седловой точкой**. Проверить существование седловой точки несложно. Из общих определений понятно, что соответствующая пара максиминной и минимаксной стратегий задает ситуацию равновесия, т. е. является решением игры. При этом если один из игроков придерживается своей оптимальной стратегии, то для другого не может быть выгодным отклонение от его оптимальной стратегии.

19. Игры со смешанными стратегиями

Игры с седловой точкой встречаются довольно редко, и более типичны случаи, когда нижняя и верхняя цена игры различны. Как известно, игрок А, постоянно придерживаясь одной и той же максиминной стратегии, гарантирует себе выигрыш, равный α. Игрок В, придерживаясь своей минимаксной стратегии, гарантирует себе проигрыш, не больший β. Понятно, что в случае β > α встает вопрос: может ли игрок А получить выигрыш, больший α, если при многократном повторении игры использовать несколько случайным образом чередующихся стратегий? Аналогично для игрока В ставится задача уменьшить величину β за счет случайного выбора своих стратегий. Случайный вектор, координатами которого являются вероятности применения чистых стратегий того или иного игрока при многократном повторении игры, называется его **смешанной (рандомизированной) стратегией.**

Таким образом, если задана антагонистическая игра Г и игрок А располагает m чистыми стратегиями, то, для того чтобы определить смешанную стратегию,

$$(p_1, ..., p_m) = S_A, \text{ где } \sum_{i=1}^{m} p_i = 1 \text{ и величина } p_i \geq 0$$

необходимо задать вектор . . . . . представляет собой вероятность применения i-й чистой стратегии игрока А при многократном повторении игры.

Аналогично, если игрок В располагает n чистыми стратегиями, то чтобы задать

$$(q_1, ..., q_n) = S_B, \text{ где } \sum_{j=1}^{n} q_j = 1$$

его смешанную стратегию, необходимо указать вектор и qi ≥ 0 – вероятность применения его j-й чистой стратегии при многократном повторении игры. Очевидно, что если в векторе S_A или S_B одна координата – 1, а все остальные – 0, то такая смешанная стратегия представляет собой **чистую стратегию**.

Ситуацией равновесия (по Нэшу) или решением игры называется пара (S_A^*, S_B^*) смешанных стратегий, обладающих тем свойством, что если один из игроков придерживается своей оптимальной стратегии, то другому невыгодно отступать от своей оптимальной стратегии. Выигрыш, соответствующий ситуации равновесия, называется ценой игры и обозначается γ.

Если оказывается, что для некоторых стратегий S_A^* игрока A и S_B^* игрока B выполняются неравенства $E(S_A, S_B^*) \leq E(S_A^*, S_B^*) \leq E(S_A^*, S_B)$ при любых (S_A, S_B), то пара (S_A^*, S_B^*) называется решением игры, соответствующие стратегии S_A^* и S_B^* – оптимальными стратегиями, а величина $E(S_A^*, S_B^*) = \gamma$ – ценой игры. Встает вопрос о существовании таких оптимальных стратегий для каждого игрока.

Конкретизируем вид функции $E(S_A, S_B)$, учитывая, что игра Г задается матрицей выигрышей $(a_{ij})_{m \times n}$. Если игроки придерживаются любых своих смешанных стратегий S_A и S_B, то ожидаемый накапливаемый выигрыш при многократном повторении игры можно вычислять как математическое ожидание выигрыша для игрока A. Это будет величина

$$E(S_A, S_B) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} p_i q_j. \qquad (8.2)$$

20. Основная теорема матричных игр

Каждая конечная матричная игра имеет, по крайней мере, одно решение, возможно, в области смешанных стратегий.

У этой теоремы есть и другая формулировка: если множество допустимых стратегий для игрока A обозначить $\boldsymbol{A}$, а множество стратегий для B через $\boldsymbol{B}$, то величины $\max\limits_{S_A \in \boldsymbol{A}} \min\limits_{S_B \in \boldsymbol{B}} E(S_A, S_B)$ и $\min\limits_{S_B \in \boldsymbol{B}} \max\limits_{S_A \in \boldsymbol{A}} E(S_A, S_B)$ существуют, причем

$$\max_{S_A \in \boldsymbol{A}} \min_{S_B \in \boldsymbol{B}} E(S_A, S_B) = \min_{S_B \in \boldsymbol{B}} \max_{S_A \in \boldsymbol{A}} E(S_A, S_B). \qquad (8.3)$$

Из равенства (8.3) как раз и следует, что существует пара стратегий (S_A^*, S_B^*) такая, что

$$E(S_A^*, S_B) \geq \gamma;\ E(S_A, S_B^*) \leq \gamma;\ E(S_A^*, S_B^*) = \gamma \text{ для любых } S_A,\ S_B.$$

21. Графическое решение игры

Когда в любой из игр 2×n с матрицей выигрышей $(a_{ij})_{2\times n}$ нет седловой точки, то, как уже отмечалось, смешанная стратегия игрока A представляет собой совокупность чисел p1 и p2, в сумме дающих единицу. Геометрически ее можно представить точкой P на единичном отрезке (рис. 8.1).

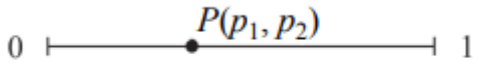


Рис. 8.1

Левая точка (точка 0) этого отрезка соответствует чистой стратегии A2, потому что в этом случае p1 = 0, p2 = 1, а правая точка - стратегии A1, потому что p1 = 1, p2 = 0. Восстановим на концах такого отрезка перпендикуляры-оси и зададим на них одинаковый масштаб.

Предположим, что игрок A использует свои чистые стратегии A1 и A2, а игрок B – свою чистую стратегию B1, тогда на правой оси можно отметить выигрыш игрока A – значение a11, а на левой – значение a21. Соединим эти точки прямой (рис. 8.3) (на основании предположения (8.2) и отмеченных в замечании 8.2 свойств линейности функции выигрыша). Очевидно, что любая промежуточная точка этой прямой определит величину выигрыша игрока A в смешанной стратегии (p1, p2). Аналогичные прямые можно построить для остальных стратегий B2, B3, ..., Bn и найти нижнюю границу множества n прямых над отрезком (0,1). Точка отрезка, при которой эта нижняя граница достигает максимума, определяет искомую оптимальную смешанную стратегию игрока A. Высота точки максимума равна цене игры.

Замечание 8.3. Может случиться, что максимум нижней границы достигается не в одной точке, а на бесконечном множестве, тогда любая точка этого множества является решением игры для игрока A.

Замечание 8.4. Максимум нижней границы не может оказаться на одной из осей, потому что по предположению анализируется игра без седловой точки. Точка максимума нижней границы является пересечением двух (или более) прямых, соответствующих чистым стратегиям игрока B. На основании теоремы об активных стратегиях неактивные стратегии для игрока B можно исключить и он может искать свою оптимальную смешанную стратегию, решая для себя игру 2×2. При этом концы единичного отрезка (рис. 8.2) будут соответствовать чистым активным стратегиям, например, B2 и B1, а стратегия Q будет смешанной стратегией (q1, q2). Опять на концах отрезка восстанавливаются перпендикуляры-оси с заданными, как для игрока A, масштабами и строятся две прямые, соответствующие чистым стратегиям

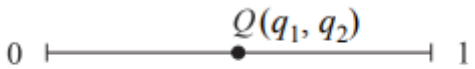


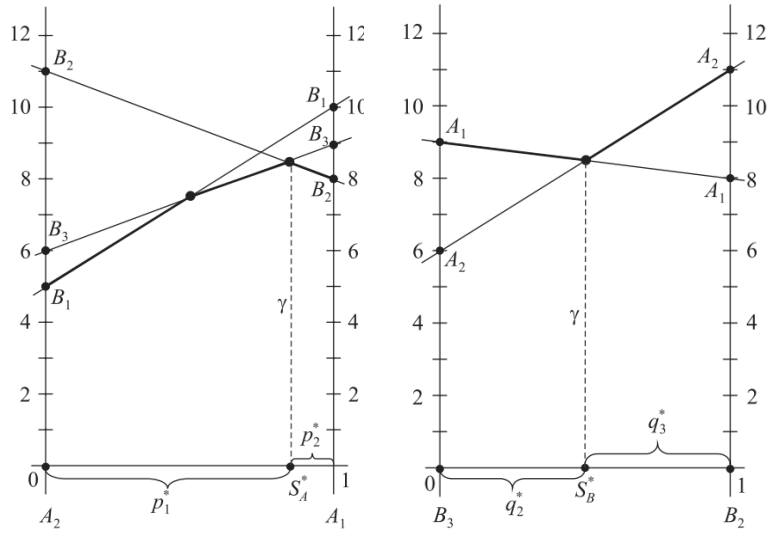
Рис. 8.2

A1, A2 игрока A. Теперь необходимо искать верхнюю границу этих двух прямых над отрезком (0,1). Точка отрезка, при которой эта верхняя граница достигает минимума, определяет искомую оптимальную смешанную стратегию игрока B. Высота точки минимума будет равна цене игры, которая, впрочем, уже определена игроком A.

Подобным образом решаются антагонистические игры m×2. Естественно, что сначала игра рассматривается с точки зрения игрока B с построением верхней границы множества m прямых, нахождением ее минимума и соответствующего

решения игры для игрока B, а также определением активных стратегий для игрока A. Далее для игрока A решается игра 2×2 путем построения нижней границы множества двух прямых и нахождения точки максимума этой нижней границы.

$$\begin{array}{c} \\ A_1 \\ A_2 \end{array}\begin{array}{c} \begin{array}{ccc} B_1 & B_2 & B_3 \end{array} \\ \begin{bmatrix} 10 & 8 & 9 \\ 5 & 11 & 6 \end{bmatrix} \end{array}.$$



22. Позиционные игры

В матричных играх фактически анализируются одноходовые партии, в которых игроки одновременно выбирают свои стратегии, оценивают выигрыши и на этом партия заканчивается. Для реальных многоходовых игр, когда игроки делают ходы поочередно, более **естественна развернутая или позиционная форма представления игры**. При этом можно учесть, что в игре участвует более двух (конечное число) игроков, что некоторые ходы могут быть случайными, а информация о состоянии партии (позиции) в каждый момент для каждого игрока может быть различна.

Определение 8.1. **Позиционной будем называть конечную игру n сторон, задаваемую следующими шестью объектами:**

1) корневым ориентированным деревом T, вершины которого расположены по уровням, причем на самом нижнем уровне расположена одна вершина – корень. Дуги ориентированы в направлении от вершины меньшего уровня к вершинам большего уровня;

2) n-мерным вектором $F_1, ..., F_n$, определяемым для каждой из висячих вершин дерева T;

3) пометками из множества $\{0, 1, ..., n\}$ у всех ветвящихся вершин дерева T;

4) пометками из множества $v_j = \{1, 2, ..., m_j\}$ для каждой дуги, выходящей из вершины j;

5) сопоставлением каждой i-й дуги с вершиной дерева T с пометкой 0 i-й координаты неотрицательного вектора $(p_1, p_2, ..., p_k)$, где $\sum_{i=1}^{k} p_i = 1$;

6) разбиением ветвящихся вершин на непересекающиеся (так называемые информационные) множества со следующими свойствами:

а) все вершины, принадлежащие одному множеству, имеют одну и ту же пометку и одинаковое количество выходящих дуг;

б) каждое множество содержит не более одной вершины, принадлежащей любому пути от корня до висячей вершины;

в) множество, в котором имеется вершина с пометкой 0, состоит из одного элемента.

Таким образом, игра в позиционной форме описывается деревом T, вершины которого соответствуют состояниям игры (позициям), а дуги, выходящие из вершин, отображают варианты выбора решений в этих позициях. Каждый путь от корня до висячей вершины дерева соответствует одной партии в игре, выигрыш каждого i-го игрока в партии равен значению элемента Fi в векторе (F1, F2, ..., Fn). Понятно, что в игре двух игроков с нулевой суммой вместо вектора (F1, F2, ..., Fn) указывается выигрыш первого игрока, который равен проигрышу второго.

Пометка у каждой ветвящейся вершины указывает номер игрока, делающего ход. Номер 0 соответствует ходу, осуществляемому некоторым случайным механизмом. Пометки vj для каждой дуги, выходящей из вершины j, обозначают варианты выбора решения игроком j. Дугам, исходящим из вершины с пометкой 0, кроме того, приписываются соответствующие вероятности из вектора (p1, p2, ..., pk). Наконец, для каждого из игроков определяется совокупность информационных множеств. Каждое информационное множество содержит необходимые сведения о сделанных выборах при определенных ходах игроков. Игрок, находящийся в любой из

вершин одного информационного множества, обладает одной и той же информацией о состоянии игры, т. е. вершины одного информационного множества неразличимы для соответствующего игрока.

Имеются игры (например, шашки, шахматы и т. п.), в которых каждый из игроков **достоверно знает**, в каком состоянии (позиции) находится игра в каждый момент времени. Это **игры с полной информацией**. В таких играх каждое информационное множество содержит в точности одну вершину.

Существуют также игры (например, карточные), в которых игрок **не знает точно**, в какой позиции находится игра. Именно подобная неполнота информации изображается на дереве игры с помощью информационных множеств. Обычно их границы обозначают на дереве пунктиром.

Стратегией игрока j в позиционной игре называется правило, ставящее в соответствие каждому информационному множеству этого игрока определенный выбор хода в этом информационном множестве. Таким образом, стратегия однозначно определяет линию поведения игрока и указывает ему, что нужно делать при любой возможной информации.

Любую конечную позиционную игру n игроков можно свести к бескоалиционной матричной игре с n матрицами выигрышей, соответствующих игрокам. **Процесс сведения позиционной игры к матричной называется нормализацией.**

Для позиционной игры двух игроков с нулевой суммой выигрышей после нормализации получается антагонистическая матричная игра, которую можно решить известными методами.

23. Коалиционные игры

При конфликтной ситуации, в которой участвует более двух игроков, анализ игры может усложниться, если некоторые игроки договариваются о совместных действиях и правилах дележа полученного выигрыша, **т. е. образуют коалицию (кооперацию)**. В таких моделях имеется явный обмен информацией между участниками (игроками).

Можно выделить 3 уровня взаимодействия (сотрудничества):

1) обмен информацией о ходе игры и складывающейся обстановке;

2) совместный выбор стратегии на основе общей договоренности;

3) объединение активных средств (ресурсов) с соответствующей координацией предпринимаемых действий.

Далее, в соответствии с целями настоящего учебного пособия, обсудим только некоторые результаты и подходы к решению коалиционных игр, учитывая лишь первый уровень взаимодействия игроков.

Пусть N – множество всех игроков $\{1, 2, ..., n\}$ и K – любое его непустое подмножество. Игроки из K договариваются о совместном действии и дележе получаемых выигрышей, т. е. образуют одну коалицию.

Число всевозможных коалиций равно $2^n - 1$ (по числу всевозможных непустых подмножеств).

Образовав коалицию, множество игроков K действуют как один игрок против остальных, и выигрыш этой коалиции зависит от применяемых стратегий каждым из n игроков.

Функция υ, ставящая в соответствие каждой коалиции K наибольший получаемый ею выигрыш $\upsilon(K)$, называется *характеристической функцией игры*.

Пусть x_i – выигрыш i-го игрока и $\upsilon(i)$ обозначает выигрыш $\upsilon(\{i\})$.

После реализации игры в коалициях происходит перераспределение выигрышей – *дележ*. Это перераспределение должно удовлетворять следующим условиям:

1) $x_i \geq \upsilon(i)$ для $i \in N$ – условие индивидуальной рациональности;

2) $\sum_{i \in N} x_i = \upsilon(N)$ – условие коллективной рациональности.

Вектор $x = (x_1, ..., x_n)$, удовлетворяющий этим условиям, называется дележом для характеристической функции υ.

Пара (N, υ), определяющая множество игроков, характеристическую функцию игры и дележ, называется *классической коалиционной игрой*.

Утверждение 8.1. Чтобы вектор $x = (x_1, ..., x_n)$ был дележом в классической коалиционной игре (N, υ), необходимо и достаточно, чтобы для всех $i \in N$:

$$x_i \geq \upsilon(i) + c_i, \text{ где } c_i \geq 0,$$

$$\sum_{i \in N} c_i = \upsilon(N) - \sum_{i \in N} \upsilon(i).$$

Коалиционная игра называется *существенной*, если выполняется неравенство $\upsilon(K) + \upsilon(L) < \upsilon(K \cup L)$ для любых $K, L \subset N$ (общий выигрыш коалиции заведомо больше суммы выигрыша всех участников коалиции).

Коалиционная игра называется *несущественной*, если

$$\upsilon(K) + \upsilon(L) < \upsilon(K \cup L) \text{ для любых } K, L \subset N.$$

Справедливы следующие свойства:

1) для того чтобы коалиционная игра была несущественной, необходимо и достаточно, чтобы

$$\sum_{i \in N} \upsilon(i) = \upsilon(N);$$

2) в несущественной игре имеется только один дележ:

$$x = (\upsilon(1);\ \upsilon(2);\ ...;\ \upsilon(n));$$

3) в существенной игре множество дележей бесконечно и имеет следующий вид

$$\upsilon(1) + \alpha_1;\ \upsilon(2) + \alpha_2;\ ...;\ \upsilon(n) + \alpha_n,$$

где $\alpha_i \geq 0$ для любого $i \in N$, $\upsilon(N) - \sum \upsilon(i) > 0$.

Теорема 8.3. Каждая существенная коалиционная игра (N, υ) стратегически эквивалентна одной и только одной игре в (0,1)-редуцированной форме.

Множество дележей, над которыми в любых коалициях не доминируют никакие другие дележи, называют ***C-ядром*** этой игры.

Теорема 8.4. Для того чтобы дележ принадлежал ***C-ядру*** коалиционной игры с характеристической функцией υ, необходимо и достаточно, чтобы выполнялось неравенство

$$\upsilon(K) \leq \sum_{i \in K} x_i \text{ для всех непустых } K \subset N.$$

Теорема 8.4 утверждает, что дележ входит в ***C-ядро*** тогда и только тогда, когда он удовлетворяет минимальным требованиям каждой коалиции. В игре с пустым ***C***-ядром всегда найдется одна неполностью удовлетворенная коалиция.

Теорема 8.5. Всякая существенная игра с постоянной суммой не имеет ядра.

24. Статистическая модель с одним плановым периодом.

Пусть u – объем запасов ресурса перед моментом принятия решения по его накоплению, x – заказываемый объем ($x \geq 0$) и $y = x + u$ – суммарный объем ресурса для выполнения будущего спроса, q – фактический уровень спроса (положим, что это неотрицательная случайная переменная).

Пусть p(q) – вероятность того, что уровень спроса на какой-то момент в точности равен q. Будем предполагать, что все введенные переменные принимают целочисленные значения. Чтобы не вводить различных коэффициентов для пересчета, примем также, что вводимые переменные приведены к одной шкале измерения. При решении задачи об управлении запасами искомые переменные – x и y, и можно считать, что они являются функциями от переменной u, т. е. $x = x(u)$, $y = y(u)$.

В качестве целевой функции примем выражение общих затрат на весь плановый период. Эти затраты складываются из следующих слагаемых:

1) при пополнении запаса необходимы затраты на покупку ресурса;

2) при наличии какого-то запаса необходимы также затраты на хранение;

3) если спрос продукции не будет удовлетворен, то предприятие понесет какие-то потери в виде штрафа за неудовлетворение спроса.

Два последних вида затрат представим в виде математического ожидания их величин в зависимости от заданных вероятностей.

Распишем подробнее общий вид целевой функции. Будем считать, что затраты на приобретение ресурса в объеме x выражается формулой

$$C(x) = \begin{cases} 0, & \text{если } x = 0, \\ K + cx, & \text{если } x > 0, \end{cases} \qquad (7.1)$$

где K – некоторые накладные расходы, не зависящие от объема заказанного ресурса; c – стоимость единицы ресурса. Таким образом, здесь принято, что стоимости приобретенных ресурсов близки (в силу целочисленности значений) к линейной функции от x.

Два последних вида ожидаемых потерь, при хранении ресурсов и при неудовлетворенном спросе, можно представить в виде функции

$$L(y) = \sum_{q=0}^{y} h(y-q)p(q) + \sum_{q>y} d(q-y)p(q), \qquad (7.2)$$

где h – затраты на хранение единицы ресурса, остающегося в запасе до конца планового периода; d – штрафные потери на единицу ресурса из-за неудовлетворенного спроса по причине отсутствия ресурса до конца планового периода. Очевидно, что рассматриваем частный случай функции L(y), когда все ожидаемые затраты также близки к линейным функциям от y. **Общая целевая функция будет иметь вид**

$$F = C(x) + L(y). \qquad (7.3)$$

Рассмотрим стратегию пополнения запасов, подчиняющуюся следующему правилу:

$$\begin{cases} y(u) = u, \quad x(u) = 0 \quad \text{при} \quad u \geq s, \\ y(u) = S, \quad x(u) = S - u \quad \text{при} \quad u < s. \end{cases} \qquad (7.4)$$

Здесь s – критический уровень запаса; S – уровень запаса, достигаемый в результате пополнения.

По этой стратегии видно, что в зависимости от значения текущего объема запасов происходит пополнение в том случае, если он становится ниже так называемого критического уровня s. Это пополнение происходит до тех пор, пока суммарный уровень запаса ресурсов не станет равным какой-то заданной величине S. Правило (7.4) в теории управления запасами (УЗ) называется (s, S)-стратегией. Для того чтобы задать (s, S)-стратегию, необходимо определить параметры s и S.

В теории управления запасами доказано, что в случае, когда целевая функция имеет вид (7.3) и ожидаемые затраты являются линейными (или более общими, выпуклыми) функциями от y, то (s, S)-стратегия - наилучшая форма пополнения запасов.

Рассмотрим способы нахождения s и S. Введем еще один параметр $R = \frac{d-c}{d+h}$. В рассматриваемой модели разумными предположениями являются неравенства $d - c > 0;\ \ d + h > 0$. Тогда $0 < R < 1$.

В теории УЗ величину R называют определяющим параметром. Именно с этим параметром связан способ нахождения значения S.

Оптимально такое значение S, при котором полный спрос на весь плановый период удовлетворяется, по крайней мере, с вероятностью R. Оказывается, что вероятность полного удовлетворения спроса можно представить в виде так называемой статистической (кумулятивной) функции распределения

$$P(S) = \sum_{q=0}^{S} p(q).$$

Поэтому для получения оптимального S требуется найти наименьшие из чисел q, при котором выполняется неравенство

$$P(S) = \sum_{q=0}^{S} p(q) \geq R. \qquad (7.5)$$

Таким образом, достаточно при $q = 0,1, ...$, накапливать сумму $\sum p(q)$ до тех пор, пока не выполнится неравенство (7.5). Наименьшее значение q, при котором это произойдет, и даст S.

Приведем способ нахождения s.

Из (s, S)-стратегии видно, что если начальный объем запасов равен s, то заказ на дополнительную поставку не производится. Следовательно, сумма ожидаемых затрат и ожидаемых штрафных потерь при $y = s$ $(x = 0)$ не должна превышать значения целевой функции при $y = S$ $\quad (x = S - s)$. Если бы этого не было, то имело бы смысл пополнить запас и тем самым уменьшить значение целевой функции. Из этого следует, что в качестве s можно выбирать наименьшее из чисел, при котором выполняется неравенство

$$L(s) \leq K + c(S - s) + L(S).$$

Таким образом, начиная с $y = S$ (это значение уже определено выше) необходимо перебирать последовательно все целочисленные значения, меньшие S, и сравнивать значения выражения $L(y) + cy$ со значением $K + cS + L(S)$. Наименьшее значение y, при котором $L(y) + cy \leq K + cS + L(S)$, определит величину $y = s$.

Таким образом, в предположении линейности (или выпуклости) целевой функции эффективно находятся величины s и S, и тем самым определяется оптимальная (s, S)-стратегия.