трудным вычислительным задачам. При их решении часто приходится прибегать к приближенным, так называемым «эвристическим» методам оптимизации, так как точные методы по своей трудоемкости оказываются неподъемными даже для современной вычислительной техники. Бывает так, что весь выигрыш, который был бы получен от точной оптимизации решения, «съедается» затратами на получение этого решения, так что «игра не стоит свеч». Здесь опять мы встречаемся с необходимостью «системного» подхода к задачам исследования операций, учета не только непосредственного выигрыша в данной операции, но и затрат на ее оптимизацию.

Правда, возможности вычислительной техники (быстродействие, объем памяти) с течением времени растут, и то, что невыгодно экономически сегодня, может сделаться выгодным завтра. С другой стороны, параллельно с ростом возможностей вычислительной техники растет и сложность оптимизационных задач, которые приходится решать. Не надо забывать, что принятие того или иного способа оптимизации решения есть тоже «решение», и иной раз весьма ответственное.

ГЛАВА 4
ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

§ 12. Метод динамического программирования

Динамическое программирование (иначе «динамическое планирование») есть особый метод оптимизации решений, специально приспособленный к так называемым «многошаговым» (или «многоэтапным») операциям.

Представим себе некоторую операцию *О,* распадающуюся на ряд последовательных «шагов» или «этапов»,— например, деятельность отрасли промышленности в течение ряда хозяйственных лет; или же преодоление группой самолетов нескольких полос противовоздушной обороны; или же последовательность тестов, применяемых при контроле аппаратуры. Некоторые операции (подобно вышеприведенным) расчленяются на шаги естественно; в некоторых членение приходится вводить искусственно — скажем, процесс наведения ра-

кеты на цель можно условно разбить на этапы, каждый из которых занимает какое-то время Δt.

Итак, рассмотрим операцию О, состоящую из m шагов (этапов). Пусть эффективность операции характеризуется каким-то показателем W, который мы для краткости будем в этой главе называть «выигрышем». Предположим, что выигрыш W за всю операцию складывается из выигрышей на отдельных шагах:

$$W = \sum_{i=1}^{m} w_i, \qquad (12.1)$$

где w_i — выигрыш на i-м шаге.

Если W обладает таким свойством, то его называют «аддитивным критерием».

Операция O, о которой идет речь, представляет собой управляемый процесс, т. е. мы можем выбирать какие-то параметры, влияющие на его ход и исход, причем на каждом шаге выбирается какое-то решение, от которого зависит выигрыш на данном шаге и выигрыш за операцию в целом. Будем называть это решение «шаговым управлением». Совокупность всех шаговых управлений представляет собой управление операцией в целом. Обозначим его буквой x, а шаговые управления — буквами $x_1, x_2, \ldots, x_m$:

$$x = (x_1, x_2, \ldots, x_m). \qquad (12.2)$$

Следует иметь в виду, что $x_1, x_2, \ldots, x_m$ в общем случае — не числа, а, может быть, векторы, функции и т. д.

Требуется найти такое управление x, при котором выигрыш W обращается в максимум:

$$W = \sum_{i=1}^{m} w_i \Rightarrow \max. \qquad (12.3)$$

То управление x^*, при котором этот максимум достигается, будем называть оптимальным управлением. Оно состоит из совокупности оптимальных шаговых управлений:

$$x^* = (x_1^*, x_2^*, \ldots, x_m^*). \qquad (12.4)$$

Тот максимальный выигрыш, который достигается при этом управлении, мы будем обозначать W^*:

$$W^* = \max_x \{W(x)\}. \qquad (12.5)$$

Формула (12.5) читается так: величина W^* есть максимум из всех $W(x)$ при разных управлениях x (максимум берется по всем управлениям x, возможным в данных условиях). Иногда это последнее оговаривается в формуле и пишут:

$$W^* = \max_{x \in X} \{W(x)\}. \qquad (12.5')$$

Рассмотрим несколько примеров многошаговых операций и для каждого из них поясним, что понимается под «управлением» и каков «выигрыш» (показатель эффективности) W.

1. Планируется деятельность группы промышленных предприятий $П_1$, $П_2$, ..., $П_k$ на период m хозяйственных лет m-летку). В начале периода на развитие группы выделены какие-то средства M, которые должны быть как-то распределены между предприятиями. В процессе работы предприятия вложенные в него средства частично расходуются (амортизируются), а частично сохраняются и снова могут быть перераспределены. Каждое предприятие за год приносит доход, зависящий от того, сколько средств в него вложено. В начале каждого хозяйственного года имеющиеся в наличии средства перераспределяются между предприятиями. Ставится вопрос: какое количество средств в начале каждого года нужно выделять каждому предприятию, чтобы, суммарный доход за m лет был максимальным?

Выигрыш W (суммарный доход) представляет собой сумму доходов на отдельных шагах (годах):

$$W = \sum_{i=1}^{m} w_i, \qquad (12.6)$$

и, значит, обладает свойством аддитивности.

Управление x_i на i-м шаге состоит в том, что в начале i-го года предприятиям выделяются какие-то средства $x_1, x_2, \ldots, x_{ik}$ (первый индекс — номер шага, второй — номер предприятия). Таким образом, шаговое управление есть вектор с k составляющими:

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{ik}). \qquad (12.7)$$

Разумеется, величины w_t в формуле (12.6) зависят от количества вложенных в предприятия средств.

Управление x всей операцией состоит из совокупности всех шаговых управлений:

$$x = (x_1, x_2, \ldots, x_m). \qquad (12.8)$$

Требуется найти такое распределение средств по предприятиям и по годам (оптимальное управление x^*), при котором величина W обращается в максимум.

В этом примере шаговые управления были векторами; в последующих примерах они будут проще и выражаться просто числами.

2. Космическая ракета состоит из m ступеней, а процесс ее вывода на орбиту — из m этапов, в конце каждого из которых очередная ступень сбрасывается. На все ступени (без учета «полезного» веса кабины) выделен какой-то общий вес:

$$G = G_1 + G_2 + \ldots + G_m,$$

где G_i — вес i-й ступени.

В результате i-го этапа (сгорания и сбрасывания i-й ступени) ракета получает приращение скорости Δ_i, зависящее от веса данной ступени и суммарного веса всех оставшихся плюс вес кабины. Спрашивается, как нужно распределить вес G между ступенями, чтобы скорость ракеты V при ее выводе на орбиту была максимальна?

В данном случае показатель эффективности (выигрыш) будет

$$V = \sum_{i=1}^{m} \Delta_i, \qquad (12.9)$$

где Δ_i — выигрыш (приращение скорости) на i-м шаге. Управление x представляет собой совокупность весов всех ступеней G_i:

$$x = (G_1, G_2, \ldots, G_m).$$

Оптимальным управлением x^* будет то распределение весов по ступеням, при котором скорость V максимальна. В этом примере шаговое управление — одно число, а именно, вес данной ступени.

3. Владелец автомашины эксплуатирует ее в течение m лет. В начале каждого года он может принять одно из трех решений:

1) продать машину и заменить ее новой;
2) ремонтировать ее и продолжать эксплуатацию;
3) продолжать эксплуатацию без ремонта.

Шаговое управление — выбор одного из этих трех решений. Непосредственно числами они не выражаются, но можно приписать первому численное значение 1, второму 2, третьему 3. Какие нужно принять решения по годам (т. е. как чередовать управления 1, 2,3), чтобы суммарные расходы на эксплуатацию, ремонт и приобретение новых машин были минимальны?

Показатель эффективности (в данном случае это не «выигрыш», а «проигрыш», но это неважно) равен

$$W = \sum_{i=1}^{m} w_i, \qquad (12.10)$$

где w_i — расходы в i-м году. Величину W требуется обратить в минимум.

Управление операцией в целом представляет собой какую-то комбинацию чисел 1, 2, 3, например:

$$x = (3, 3, 2, 2, 2, 1, 3, \ldots),$$

что означает: первые два года эксплуатировать машину без ремонта, последующие три года ее ремонтировать, в начале шестого года продать, купить новую, затем снова эксплуатировать без ремонта и т. д. Любое управление представляет собой вектор (совокупность чисел):

$$x = (j_1, j_2, \ldots, j_m), \qquad (12.11)$$

где каждое из чисел $j_1, j_2, \ldots, j_m$ имеет одно из трех значений: 1, 2 или 3. Нужно выбрать совокупность чисел (12.11), при которой величина (12.10) минимальна.

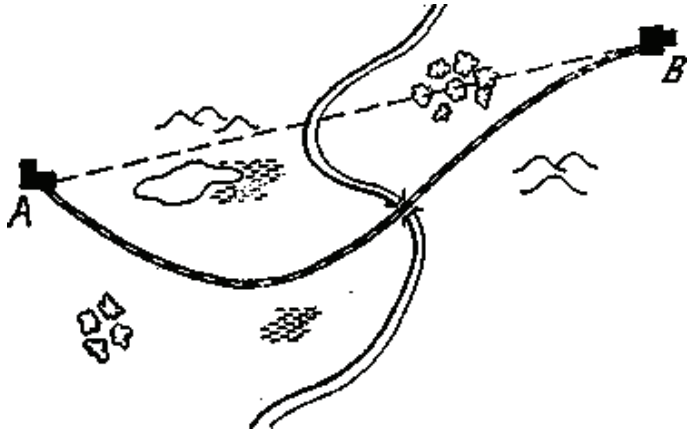

Рис. 12.1.

4. Прокладывается участок железнодорожного пути между пунктами A и B (рис. 12.1). Местность пересе-

ченная, включает лесистые зоны, холмы, болота, реку, через которую надо строить мост. Требуется так провести дорогу из A в B, чтобы суммарные затраты на сооружение участка были минимальны.

В этой задаче, в отличие от трех предыдущих, нет естественного членения на шаги: его приходится вводить искусственно, для чего, например, можно отрезок AB разделить на m частей, провести через точки деления прямые, перпендикулярные AB, и считать за «шаг» переход с одной такой прямой на другую. Если провести их достаточно близко друг от друга, то можно считать на каждом шаге участок пути прямолинейным. Шаговое управление на i-м шаге представляет собой угол φ_i, который составляет участок пути с прямой AB. Управление всей операцией состоит из совокупности шаговых управлений:

$$x = (\varphi_1, \varphi_2, \ldots, \varphi_m).$$

Требуется выбрать такое (оптимальное) управление x^*, при котором суммарные затраты на сооружение всех участков минимальны:

$$W = \sum_{i=1}^{m} w_i \Rightarrow \min. \qquad (12.12)$$

Итак, мы рассмотрели несколько примеров многошаговых задач исследования операций. А теперь поговорим о том, как можно решать подобного рода задачи?

Любую многошаговую задачу можно решать по-разному: либо искать сразу все элементы решения на всех m шагах, либо же строить оптимальное управление шаг за шагом, па каждом этапе расчета оптимизируя только один шаг. Обычно второй способ оптимизации оказывается проще, чем первый, особенно при большом числе шагов.

Такая идея постепенной, пошаговой оптимизации и лежит в основе метода динамического программирования. Оптимизация одного шага, как правило, проще оптимизации всего процесса: лучше, оказывается, много раз решить сравнительно простую задачу, чем один раз — сложную.

С первого взгляда идея может показаться довольно тривиальной. В самом деле, чего казалось бы, проще: если трудно оптимизировать операцию в целом, раз-

бить ее па ряд шагов. Каждый такой шаг будет отдельной, маленькой операцией, оптимизировать которую уже нетрудно. Надо выбрать на этом шаге такое управление, чтобы эффективность этого шага была максимальна. Не так ли?

Нет, вовсе не так! Принцип динамического программирования отнюдь не предполагает, что каждый шаг оптимизируется отдельно, независимо от других. Напротив, шаговое управление должно выбираться дальновидно, с учетом всех его последствий в будущем. Что толку, если мы выберем на данном шаге управление, при котором эффективность э т о г о шага максимальна, если этот шаг лишит нас возможности хорошо выиграть на последующих шагах?

Пусть, например, планируется работа группы промышленных предприятий, из которых часть занята выпуском предметов потребления, а остальные производят для них машины. Задача операции — получить за *m* лет максимальный объем выпуска предметов потребления. Допустим, планируются капиталовложения на первый год. Исходя из узких интересов этого шага (года), мы должны были бы все наличные средства вложить в производство предметов потребления. Но правильно ли будет такое решение с точки зрения эффективности операции в целом? Очевидно, нет. Это решение — расточительное, недальновидное. Имея в виду будущее, надо выделить какую-то долю средств и па производство машин. От этого объем продукции за первый год, конечно, снизится, зато будут созданы условия для его увеличения в последующие годы.

Еще пример. Допустим, что в задаче 4 (прокладка железнодорожного пути из *А* в *В)* мы прельстимся идеей сразу же устремиться по самому легкому (дешевому) направлению. Что толку от экономии на первом шаге, если в дальнейшем он заведет нас (буквально или фигурально) в «болото»?

Значит, *планируя многошаговую операцию, надо выбирать управление на каждом шаге с учетом всех его будущих последствий на еще предстоящих шагах.* Управление на і-м шаге выбирается не так, чтобы выигрыш именно на д а н н о м шаге был максимален, а так, чтобы была максимальна сумма выигрышей я а всех о с т а в ш и х с я до к о н ц а ш а г а х плюс данный.

Однако из этого правила есть исключение. Среди всех шагов есть один, который может планироваться попросту, без оглядки на будущее. Какой это шаг? Очевидно, последний! Этот шаг, единственный из всех, можно планировать так, чтобы он сам, как таковой, принес наибольшую выгоду.

Поэтому процесс динамического программирования обычно разворачивается от конца к началу: прежде всего планируется последний, m-й шаг. А как его спланировать, если мы не знаем, чем кончился предпоследний? Т. е. не знаем условий, в которых мы приступаем к последнему шагу?

Вот тут-то и начинается самое главное. Планируя последний шаг, нужно сделать разные предположения о том, чем кончился предпоследний, (*m*— 1)-й шаг, и для каждого из этих предположений найти условное оптимальное управление на *m*-м шаге («условное» потому, что оно выбирается исходя из условия, что предпоследний шаг кончился так-то и так-то).

Предположим, что мы это сделали, и для каждого из возможных исходов предпоследнего шага знаем условное оптимальное управление и соответствующий ему условный оптимальный выигрыш на m-м шаге. Отлично! Теперь мы можем оптимизировать управление на предпоследнем, *(m* — 1)-м шаге. Снова сделаем все возможные предположения о том, чем кончился предыдущий, *(m* — 2)-й шаг, и для каждого из этих предположений найдем такое управление на *(m* — 1)-м шаге, при котором выигрыш за последние два шага (из которых *m-й* уже оптимизирован!) максимален. Так мы найдем для каждого исхода (*m* — 2)-го шага условное оптимальное управление на (*m*—1)-м шаге и условный оптимальный выигрыш на двух последних шагах. Далее, «пятясь назад», оптимизируем управление на *(m* — 2)-м шаге и т. д., пока не дойдем до первого.

Предположим, что все условные оптимальные управления и условные оптимальные выигрыши за весь «хвост» процесса (на всех шагах, начиная от данного и до конца) нам известны. Это значит: мы знаем, что надо делать, как управлять на данном шаге и что мы за это получим на «хвосте», в каком бы состоянии ни был процесс к началу шага. Теперь мы можем построить

уже не условно оптимальное, а просто О п т и м а л ь ное у п р а в л е н и е x^* и найти не условно оптимальный, а просто оптимальный выигрыш W^*. В самом деле, пусть мы знаем, в каком состоянии S_0 была управляемая система (объект управления S) в начале первого шага. Тогда мы можем выбрать оптимальное управление x_1^* на первом шаге. Применив его, мы изменим состояние системы на некоторое новое S_1^*; в этом состоянии мы подошли ко второму шагу.

Тогда нам тоже известно условное оптимальное управление x_2^*, которое к концу второго шага переводит систему в состояние S_2^*, и т. д. Что касается оптимального выигрыша W^* за всю операцию, то он нам уже известен: ведь именно на основе его максимальности мы выбирали управление на первом шаге.

Таким образом, в процессе оптимизации управления методом динамического программирования многошаговый процесс «проходится» дважды: первый раз — от конца к началу, в результате чего находятся условные оптимальные управления и условные оптимальные выигрыши за оставшийся «хвост» процесса; второй раз — от начала к концу, когда нам остается только «прочитать» уже готовые рекомендации и найти безусловное оптимальное управление x^*, состоящее из оптимальных шаговых управлений $x_1^*, x_2^*, \ldots, x_m^*$.

Первый этап — условной оптимизации — несравненно сложнее и длительнее второго. Второй этап почти не требует дополнительных вычислений.

Автор не льстит себя надеждой, что из такого описания метода динамического программирования читатель, не встречавшийся с ним до сих пор, поймет по-настоящему его идею. Истинное понимание возникает при рассмотрении конкретных примеров, к которым мы и перейдем.

§ 13. Примеры решения задач динамического программирования

В этом параграфе мы рассмотрим (и даже решим до конца) несколько простых (до крайности упрощенных) примеров задач динамического программирования.

1. Прокладка наивыгоднейшего пути между двумя пунктами. Вспомним задачу 4 предыдущего параграфа и решим ее до конца в крайне (и намеренно) упрощенных условиях. Нам нужно соорудить путь, соединяющий

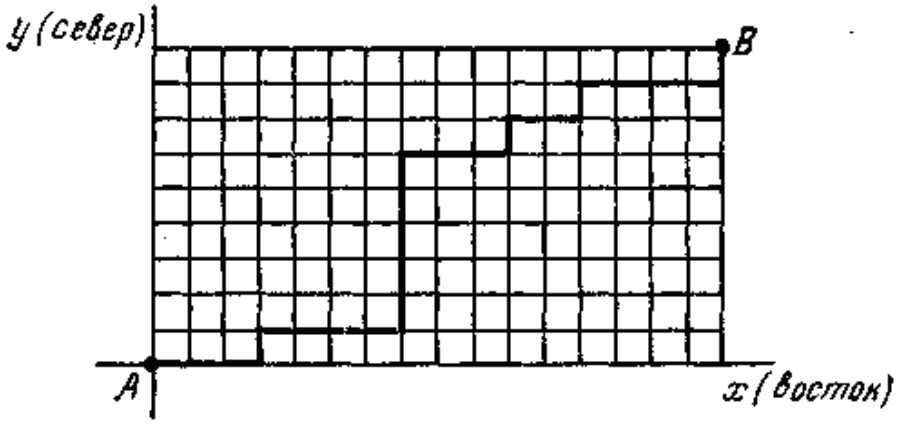

Рис. 13.1.

два пункта *A* и *B,* из которых второй лежит к северо-востоку от первого. Для простоты допустим, что прокладка пути состоит из ряда шагов, и на каждом шаге мы можем двигаться либо строго на восток, либо строго на север; любой путь из *A* в *B* представляет собой ступенчатую ломаную линию, отрезки которой параллельны одной из координатных осей (рис. 13.1). Затраты на сооружение каждого из таких отрезков известны. Требуется проложить такой путь из *A* в *B,* при котором суммарные затраты минимальны.

Как это сделать? Можно поступить одним из двух способов: либо перебрать все возможные варианты пути, и выбрать тот, на котором затраты минимальны (а при большом числе отрезков это очень и очень трудно!); либо разделить процесс перехода из *A* в *B* на отдельные шаги (один шаг — один отрезок) и оптимизировать управление по шагам. Оказывается, второй способ несравненно удобнее! Тут, как и везде **в** исследовании операций, сказываются преимущества целенаправленного, организованного поиска решения перед наивным «слепым» перебором.

Продемонстрируем, как это делается, на конкретном примере. Разделим расстояние от *A* до *B* в восточном направлении, скажем, на 7 частей, а в северном — на 5 частей (в принципе дробление может быть сколь угодно мелким). Тогда любой путь из *A* в *B* со-

стоит из $m = 7 + 5 = 12$ отрезков, направленных на восток или на север (рис. 13.2). Проставим на каждом из отрезков число, выражающее (в каких-то условных единицах) стоимость прокладки пути по этому отрезку. Требуется выбрать такой путь из *A* в *B,* для которого сумма чисел, стоящих на отрезках, минимальна.

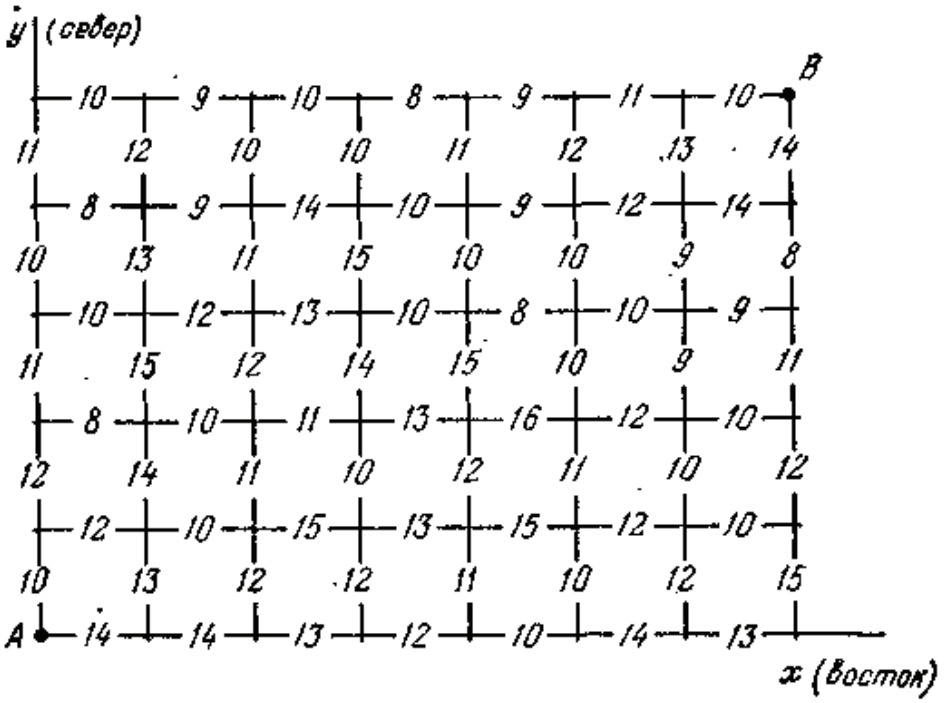

Рис. 13.2.

Будем рассматривать сооружаемый путь как управляемую систему *S,* перемещающуюся под влиянием управления из начального состояния *A* в конечное *B.* Состояние этой системы перед началом каждого шага будет характеризоваться двумя координатами: восточной (*x)* и северной *(y),* обе — целочисленные ($0 \leq x \leq \leq 7$, $0 \leq y \leq 5$). Для каждого из состояний системы (узловой точки прямоугольной сетки на рис. 13.2) мы должны найти условное оптимальное управление: идти нам из этой точки на север (управление «с») или на восток (управление «в»). Выбирается это управление так, чтобы стоимость всех оставшихся до конца шагов (включая данный) была минимальна. Эту стоимость мы по-прежнему будем называть «условным оптимальным выигрышем» (хотя в данном случае это не «выигрыш», а «проигрыш») для данного состояния системы *S* перед началом очередного шага.

Процедуру условной оптимизации будем разворачивать в обратном направлении — от конца к началу. Прежде всего произведем условную оптимизацию последнего, 12-го шага. Рассмотрим отдельно правый верхний угол нашей прямоугольной сетки (рис. 13.3). Где мы можем находиться после 11-го шага? Только

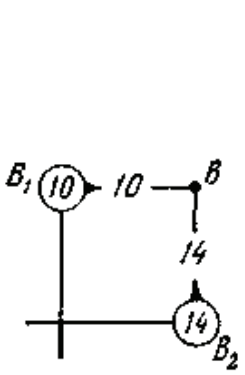

Рис. 13.3.

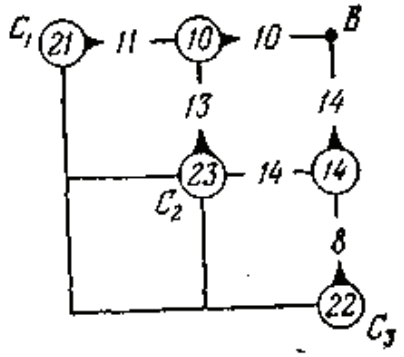

Рис. 13.4.

там, откуда за один (последний) шаг можно попасть в *B,* т. е. в одной из точек B_1 или B_2. Если мы находимся в точке *B\,* у нас нет выбора (управление вынужденное): надо идти на восток, и это обойдется нам в 10 единиц. Запишем это число 10 в кружке у точки *B\,* а оптимальное управление покажем короткой стрелкой, исходящей из B_1 и направленной на восток. Для точки B_2 управление тоже вынужденное (север), расход до конца равен 14, мы его запишем в кружке у точки B_2. Таким образом, условная оптимизация последнего шага сделана, и условный оптимальный выигрыш для каждой из точек B_1, B_2 найден и записан в соответствующем кружке.

Теперь давайте оптимизировать предпоследний (11-й) шаг. После пред-предпоследнего (10-го) шага мы могли оказаться в одной из точек C_1, C_2, C_3 (рис. 13.4). Найдем для каждой из них условное оптимальное управление и условный оптимальный выигрыш. Для точки C_1 управление вынужденное: идти на восток; обойдется это нам до конца в 21 единицу (11 на данном шаге, плюс 10, записанных в кружке при B_i). Число 21 записываем в кружке при точке C_1. Для точки C_2 управление уже не вынужденное: мы можем идти как на восток, так и на север. В первом случае мы затратим на данном шаге 14 единиц и от B_2 до конца — еще 14, всего 28 единиц. Если пойдем на се-

вер, затратим 13 + 10, всего 23 единицы. Значит, условное оптимальное управление в точке C_2 — идти на север (отмечаем это стрелкой, а число 23 записываем в кружке у C_2). Для точки C_3 управление снова вынужденное («с»), обойдется это до конца в 22 единицы (ставим стрелку на север, число 22 записываем в кружке при C_3).

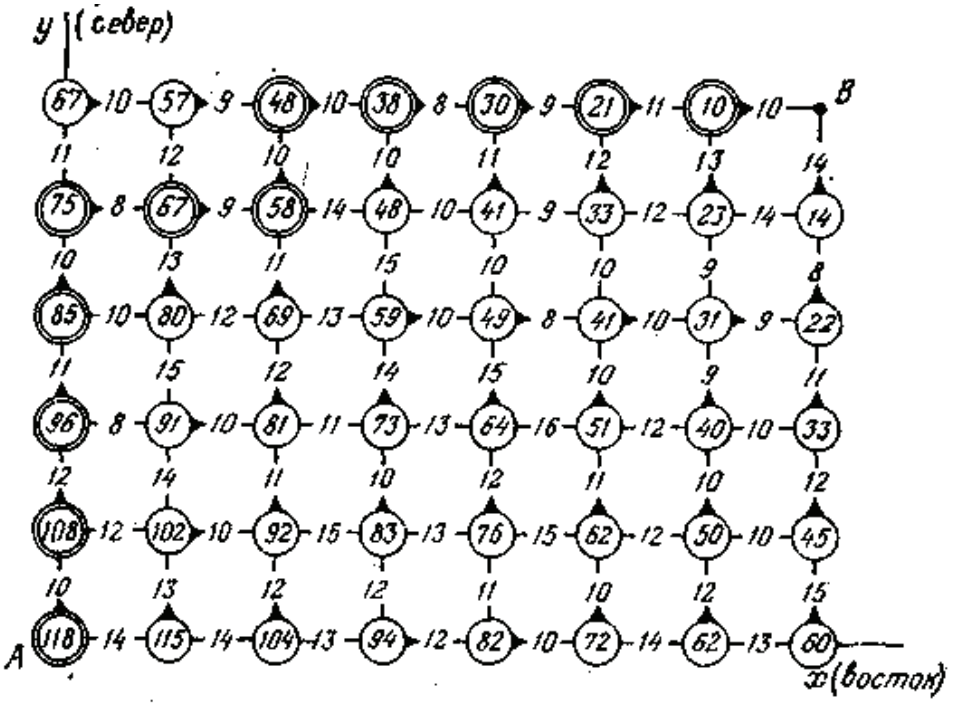

Рис. 13.5.

Аналогично, «пятясь» от предпоследнего шага назад, найдем для каждой точки с целочисленными координатами условное оптимальное управление («с» или «в»), которое обозначим стрелкой, и условный оптимальный выигрыш (расход до конца пути), который запишем в кружке. Вычисляется он так: расход на данном шаге складывается с уже оптимизированным расходом, записанным в кружке, куда ведет стрелка. Таким образом, на каждом шаге мы оптимизируем только этот шаг, а следующие за ним — уже оптимизированы. Конечный результат процедуры оптимизации показан на рис. 13.5.

Таким образом, условная оптимизация уже выполнена: в какой бы из узловых точек мы ни находились, мы уже знаем, куда идти (стрелка) и во что нам обойдется путь до конца (число в кружке).. В кружке при

точке A записан оптимальный выигрыш на все сооружение пути из A в B: $W^* = 118$.

Теперь остается построить безусловное оптимальное управление — траекторию, ведущую из A и B самым дешевым способом. Для этого нужно только «слушаться стрелок», т. е. прочитать, что они предписывают делать на каждом шаге. Такая оптимальная траектория отмечена на рис. 13.5 дважды обведенными кружками. Соответствующее безусловное оптимальное управление будет:

$$x^* = (\text{с}, \text{с}, \text{с}, \text{с}, \text{в}, \text{в}, \text{с}, \text{в}, \text{в}, \text{в}, \text{в}, \text{в}),$$

т. е. первые четыре шага мы должны делать на север, следующие два — на восток, затем опять один на север и остальные пять — на восток. Задача решена.

Заметим, что в ходе условной оптимизации мы можем столкнуться со случаем, когда оба управления для какой-то точки на плоскости являются оптимальными, т. е. приводят к одинаковому расходу средств от этой точки до конца, Например, в точке с координатами (5; 1) оба управления «с» и «в» являются оптимальными и дают расход до конца равным 62. Из них мы произвольно выбираем любое (в нашем случае мы выбрали «с»; с тем же успехом мы могли бы выбрать «в»). Такие случаи неоднозначного выбора оптимального управления постоянно встречаются в динамическом программировании; в дальнейшем мы специально отмечать их не будем, а попросту выберем произвольно любой из равноценных вариантов. От этого произвола, разумеется, может зависеть оптимальное у п р а в л е ние всем процессом, но не оптимальный выигрыш. Вообще, в задачах динамического программирования (как и в задачах линейного) решение далеко не всегда единственное.

А теперь вернемся к началу и попробуем решить задачу «наивным» способом, выбирая на каждом шаге, начиная с первого, самое выгодное (для этого шага) направление (если таких два, выбираем любое). Таким способом мы получим управление

$$x = (\text{с}, \text{с}, \text{в}, \text{в}, \text{в}, \text{в}, \text{с}, \text{в}, \text{в}, \text{в}, \text{с}, \text{с}).$$

Подсчитаем расходы для этой траектории. Они будут равны $W = 10 + 12 + 8 + 10 + 11 + 13 + 15 + 8 + + 10 + 9 + 8 + 14 = 128$, что безусловно больше, чем

$W^* = 118$. В данном случае разница не очень велика, но в других она может быть существенной.

В решенной выше задаче условия были намеренно до крайности упрощены. Разумеется, никто не будет вести железнодорожный путь «по ступенькам», перемещаясь только строго на север или строго на восток. Такое упрощение мы сделали для того, чтобы в каждой точке выбирать только из двух управлений: «с» или «в». Можно было бы вместо двух возможных направлений ввести их несколько и, кроме того, взять шаги помельче; принципиального значения это не имеет, по, разумеется, усложняет и удлиняет расчеты.

Заметим, что задачи, сходные с рассмотренной выше, очень часто встречаются на практике: например, при выборе наискорейшего пути между двумя точками или наиболее экономного (в смысле расхода горючего) набора скорости и высоты летательным аппаратом.

Сделаем одно попутное замечание. Внимательный читатель, вероятно, заметил, что в нашей задаче точки *A* и *B* (начало и конец) в принципе ничем друг от друга не отличаются: можно было бы строить условные оптимальные управления не с конца к началу, а с начала к концу, а безусловные — в обратном направлении. Действительно, это так: в любой задаче динамического программирования «начало» и «конец» можно поменять местами. Это совершенно равносильно описанной ранее методике в расчетном отношении, но несколько менее удобно при словесном объяснении идеи метода: легче аргументировать, ссылаясь на «уже сложившиеся» условия к началу данного шага, чем на те, которые еще «предстоят» после этого шага. По существу же оба подхода совершенно равносильны.

2. Задача о распределении ресурсов. Метод динамического программирования позволяет с успехом решать многие экономические задачи (см., например, [6, 10]). Рассмотрим одну из простейших таких задач. В нашем распоряжении имеется какой-то запас средств (ресурсов) *K,* который должен быть распределен между *m* предприятиями $П_1$, $П_2$, ..., $П_m$. Каждое из предприятий $П_i$ при вложении в него каких-то средств *x* приносит доход, зависящий от *x,* т. е. представляющий собой какую-то функцию $\varphi_i(x)$. Все функции $\varphi_i(x)$ ($i = 1, 2, ..., m$) заданы (разумеется, эти функции — не-

убывающие). Спрашивается, как нужно распределить средства K между предприятиями, чтобы в сумме они дали максимальный доход?

Эта задача легко решается методом динамического программирования. Хотя в своей постановке она не содержит упоминания о времени, можно все же операцию распределения средств мысленно развернуть в какой-то последовательности, считая за первый шаг вложение средств в предприятие $П_1$, за второй — в $П_2$ и т. д.

Управляемая система S в данном случае — средства или ресурсы, которые распределяются. Состояние системы S перед каждым шагом характеризуется одним числом S — наличным запасом еще не вложенных средств. В этой задаче «шаговыми управлениями» являются средства $x_1, x_2, \ldots, x_m$, выделяемые предприятиям. Требуется найти оптимальное управление, т. е. такую совокупность чисел $x_1, x_2, \ldots, x_m$, при которой суммарный доход максимален:

$$W = \sum_{i=1}^{m} \varphi_i(x_i) \Rightarrow \max. \qquad (13.1)$$

Решим эту задачу сначала в общем, формульном виде, а потом — для конкретных числовых данных. Найдем для каждого i-го шага условный оптимальный выигрыш (от этого шага и до конца), если мы подошли к данному шагу с запасом средств S. Обозначим условный оптимальный выигрыш $W_i(S)$, а соответствующее ему условное оптимальное управление — средства, вкладываемые в i-е предприятие, — $x_i(S)$.

Начнем оптимизацию с последнего, m-го шага. Пусть мы подошли к этому шагу с остатком средств S. Что нам делать? Очевидно, вложить всю сумму S целиком в предприятие $П_m$. Поэтому условное оптимальное управление на m-м шаге: отдать последнему предприятию все имеющиеся средства S, т. е.

$$x_m(S) = S,$$

а условный оптимальный выигрыш

$$W_m(S) = \varphi_m(S).$$

Задаваясь целой гаммой значений S (располагая их достаточно тесно), мы для каждого значения S будем знать $x_m(S)$ и $W_m(S)$. Последний шаг оптимизирован.

Перейдем к предпоследнему, *(m* — 1)-му шагу. Пусть мы подошли к нему с запасом средств *S.* Обозначим $W_{m-1}(S)$ условный оптимальный выигрыш на двух последних шагах: *(m* — 1)-м и *m-м* (который уже оптимизирован). Если мы выделим на *(m* — 1)-м шаге *(m*— 1)-му предприятию средства *x,* то на последний шаг останется *S* — *x.* Наш выигрыш на двух последних шагах будет равен

$$\varphi_{m-1}(x) + W_m(S - x),$$

и нужно найти такое *x,* при котором этот выигрыш максимален:

$$W_{m-1}(S) = \max_{x \leqslant S} \{\varphi_{m-1}(x) + W_m(S - x)\}. \qquad (13.2)$$

Знак $\max\limits_{x \leqslant S}$ означает, что берется максимальное значение по всем *x,* какие только возможны (вложить больше, чем *S,* мы не можем), от выражения, стоящего в фигурных скобках. Этот максимум и есть условный оптимальный выигрыш за два последних шага, а то значение *x,* при котором этот максимум достигается,— условное оптимальное управление на *(m* — 1)-м шаге. Далее оптимизируем *(m* — 2)-й, *(m* — 3)-й и т. д. шаги. Вообще, для любого *i-го* шага будем находить условный оптимальный выигрыш за все шаги с этого и до конца по формуле

$$W_i(S) = \max_{x \leqslant S} \{\varphi_i(x) + W_{i+1}(S - x)\} \qquad (13.3)$$

и соответствующее ему условное оптимальное управление $x_i(S)$ — то значение *x,* при котором этот максимум достигается.

Продолжая таким образом, дойдем, наконец, до 1-го предприятия $П_1$. Здесь нам не нужно будет варьировать значения *S;* мы точно знаем, что запас средств перед первым шагом равен *К:*

$$W^* = W_1(K) = \max_{x \leqslant K} \{\varphi_1(x) + W_2(K - x)\}. \quad (13.4)$$

Итак, максимальный выигрыш (доход) от всех предприятий найден. Теперь остается только «прочесть рекомендации». То значение *x,* при котором достигается максимум (13.4), и есть оптимальное управление x_1^*

па 1-м шаге. После того как мы вложим эти средства в 1-е предприятие, у нас их останется $K—x_1^*$. «Читая» рекомендацию для этого значения S, выделяем второму предприятию оптимальное количество средств: $x_2^* = x_2(K — x_1^*)$, и т. д. до конца.

А теперь решим численный пример. Исходный запас средств $K = 10$ (условных единиц), и требуется его оптимальным образом распределить между пятью предприятиями $(m = 5)$. Для простоты предположим, что вкладываются только целые количества средств. Функции дохода $\varphi_i(x)$ заданы в таблице 13.1.

Таблица 13.1

x	$\varphi_1(x)$	$\varphi_2(x)$	$\varphi_3(x)$	$\varphi_4(x)$	$\varphi_5(x)$
1	0,5	0,1	0,6	0,3	1,0
2	1,0	0,5	1,1	0,6	1,2
3	1,4	1,2	1,2	1,3	1,3
4	2,0	1,8	1,4	1,4	1,3
5	2,5	2,5	1,6	1,5	1,3
6	2,8	2,9	1,7	1,5	1,3
7	3,0	3,5	1,8	1,5	1,3
8	3,0	3,5	1,8	1,5	1,3

В каждом столбце, начиная с какой-то суммы вложений, доходы перестают возрастать (реально это соответствует тому, что каждое предприятие способно «освоить» лишь ограниченное количество средств).

Произведем условную оптимизацию так, как это было описано выше, начиная с последнего, 5-го шага. Каждый раз, когда мы подходим к очередному шагу, имея запас средств S, мы пробуем выделить на этот шаг то или другое количество средств, берем выигрыш на данном шаге по таблице 13.1, складываем с уже оптимизированным выигрышем на всех последующих шагах до конца (учитывая, что средств у нас осталось уже меньше, как раз на такое количество средств, которое мы выделили) и находим то вложение, на котором эта сумма достигает максимума. Это вложение и есть условное оптимальное управление на данном шаге, а сам максимум — условный оптимальный выигрыш.

В таблице 13.2 даны результаты условной оптимизации по всем шагам. Таблица построена так: в первом столбце даются значения запаса средств *S,* с которым мы подходим к данному шагу. Далее таблица разделена на пять пар столбцов, соответственно номеру шага. В первом столбце каждой пары приводится значение

Таблица 13.2

S	$i=5$		$i=4$		$i=3$		$i=2$		$i=1$	
	$x_5(S)$	$W_5(S)$	$x_4(S)$	$W_4(S)$	$x_3(S)$	$W_3(S)$	$x_2(S)$	$W_2(S)$	$x_1(S)$	$W_1(S)$
1	**1**	1,0	**0**	1,0	0	1,0	0	1,0		
2	2	1,2	1	1,3	1	1,6	0	1,6		
3	3	1,3	2	1,6	**2**	2,1	0	2,1		
4	4	1,3	3	2,3	2	2,4	0	2,4		
5	5	1,3	3	2,5	1	2,9	0	2,9		
6	6	1,3	4	2,6	2	3,4	5	3,5		
7	7	1,3	5	2,7	2	3,6	5	4,1		
8	8	1,3	5	2,8	4	3,7	**5**	4,6		
9	9	1,3	6	2,8	5	3,9	7	5,1		
10	10	1,3	7	2,8	5	4,1	7	5,6	**2**	**5,6**

условного оптимального управления, во втором — условного оптимального выигрыша. Таблица заполняется слева направо, сверху вниз. Решение на пятом — последнем — шаге вынужденное: выделяются все средства; на всех остальных шагах решение приходится оптимизировать. В результате последовательной оптимизации 5-го, 4-го, 3-го, 2-го и 1-го шагов мы получим полный список всех рекомендаций по оптимальному управлению и безусловный оптимальный выигрыш W^* за всю операцию — в данном случае он равен 5,6. В последних двух столбцах таблицы 13.2 заполнена только одна строка, так как состояние системы перед началом первого шага нам в точности известно: $S_0 = = K = 10$. Оптимальные управления на всех шагах выделены рамкой. Таким образом, мы получили окончательный вывод: надо выделить первому предприятию две единицы из десяти, второму — пять единиц, третьему — две, четвертому — ни одной, пятому — одну единицу. При этом распределении доход будет максимален и равен 5,6.

Чтобы читателю было понятно, как заполняется таблица 13.2, продемонстрируем это на одном образце расчета. Пусть, например, нам нужно оптимизировать решение $x_3(7)$— как поступать на третьем шаге, если мы подошли к нему с запасом средств $S = 7$, и сколько максимум мы можем выиграть на всех оставшихся

Таблица 13.3

x	$7 - x$	$\varphi_3(x)$	$W_4(7-x)$	$\varphi_3(x) + W_4(7-x)$
7	0	1,8	0	1,8
6	1	1,7	1,0	2,7
5	2	1,6	1,3	2,9
4	3	1,4	1,6	3,0
3	4	1,2	2,3	3,5
2	5	1,1	2,5	3,6
1	6	0,6	2,6	3,2
0	7	0	2,7	2,7

шагах, включая третий? Предположим, что все шаги после третьего (4-й и 5-й) уже оптимизированы, т. е. заполнены две первые пары столбцов таблицы 13.2. Найдем $x_3(7)$ и $W_3(7)$. Для этого составим вспомогательную табличку (см. таблицу 13.3). В первом ее столбце перечислены все возможные вложения x на третьем шаге, не превосходящие $5 = 7$. Во втором столбце — то, что останется после такого вложения от запаса средств $S = 7$. В третьем столбце — выигрыш на третьем шаге от вложения средств x в третье предприятие (заполняется по столбцу $\varphi_3(x)$ таблицы 13.1). В четвертом столбце — оптимальный выигрыш на всех оставшихся шагах (четвертом и пятом) при условии, что мы подошли к четвертому шагу с оставшимися средствами (заполняется по столбцу $i = 4$ таблицы 13.2). В пятом столбце — сумма двух выигрышей: шагового и оптимизированного дальнейшего при данном вложении x в третий шаг.

Из всех выигрышей последнего столбца выбирается максимальный (в таблице 13.3 он равен $W_3(7) = 3{,}6$, а соответствующее управление $x(7) = 2$).

Возникает вопрос: а что если во вспомогательной таблице типа 13.3 максимум достигается не при одном

x, а при двух или больше? Отвечаем: совершенно все равно, какое из них выбрать; от этого выигрыш не зависит. Вообще, в задачах динамического программирования решение вовсе не должно быть единственным (мы об этом уже упоминали).

3. Задача о загрузке машины. Пользуясь методом динамического программирования, можно с успехом решать ряд задач оптимизации, описанных в главе 3, в частности, некоторые задачи целочисленного программирования. Заметим, что целочисленность решений, так затрудняющая задачи линейного программирования, в данном случае не усложняет, а наоборот, упрощает процедуру (как нам ее упростила целочисленность вложений в предыдущей задаче).

В качестве примера рассмотрим задачу о загрузке машины (мы уже упоминали о ней в предыдущей главе): имеется определенный набор предметов $П_1$, $П_2$, ..., $П_n$ (каждый в единственном экземпляре); известны их веса q_1, q_2, ..., q_n и стоимости c_1, c_2, ..., c_n. Грузоподъемность машины равна Q. Спрашивается, какие из предметов нужно взять в машину, чтобы их суммарная стоимость (при суммарном весе $\leq Q$) была максимальна?

Нетрудно заметить, что эта задача, в сущности, ничем не отличается от предыдущей (распределение ресурсов между n предприятиями), но несколько проще ее. В самом деле, процесс загрузки машины можно представлять себе как состоящий из n шагов; на каждом шаге мы отвечаем на вопрос: б р а т ь данный предмет в машину или не б р а т ь? Управление на i-м шаге равно единице, если мы данный (i-й) предмет берем, и нулю — если не берем. Значит, на каждом шаге у нас всего два управления, а это очень приятно.

А чем мы будем характеризовать состояние системы S перед очередным шагом? Очевидно, весом S, который еще остался в нашем распоряжении до конца (полной загрузки машины) после того, как предыдущие шаги выполнены (какие-то предметы погружены в машину). Для каждого из значений S мы должны . найти $W_i(S)$ — суммарную максимальную стоимость предметов, которыми можно «догрузить» машину при данном значении S, и положить $x_i(S) = 1$, если мы данный (i-й) предмет б е р е м в машину, и $x_i(S) = 0$,

если не берем. Затем эти условные рекомендации должны быть прочтены, и дело с концом!

Решим до конца конкретный числовой пример: имеется шесть предметов, веса и стоимости которых указаны в таблице 13.4.

Таблица 13.4

Предмет $П_i$	$П_1$	$П_2$	$П_3$	$П_4$	$П_5$	$П_6$
Вес q_i	4	7	11	12	16	20
Стоимость c_i	7	10	15	20	27	34

Суммарная грузоподъемность машины $Q = 35$ единиц веса. Требуется указать номера предметов, которые нужно включить в груз, чтобы их суммарная стоимость была максимальна[1].

Как и ранее, будем придавать S только целые значения. Условная оптимизация решения показана в таблице 13.5, где в каждой строке для соответствующего номера шага (номера предмета) приведены: условное оптимальное управление x_i (0 или 1) и условный оптимальный выигрыш Wi (стоимость всех оставшихся до конца предметов при оптимальном управлении на всех шагах). Как эта таблица составляется, мы уже объяснять не будем — тут полная аналогия с предыдущей задачей, с той разницей, что возможные управления только 0 или 1.

В таблице 13.5 выделены: оптимальный выигрыш $W^* = 57$ и оптимальные шаговые управления, при которых этот выигрыш достигается: $x_1^* = 0$, $x_2^* = 1$, $x_3^* = 0$, $x_4^* = 1$, $x_5^* = 1$, $x_6^* = 0$, т. е. загрузить машину надо предметами 2, 4 и 5, суммарный вес которых равен в точности 35 (вообще это необязательно; при оптимальном выборе грузов может быть и некоторый общий «недогруз»).

Заметим, что в нашем элементарном примере, возможно, было бы проще искать решение «простым перебором»), пробуя все возможные комбинации предме-

[1] Предметы в таблице 13.4 перенумерованы в порядке возрастания веса; стоимости их также возрастают, что не обязательно, но естественно. Заранее ясно, что загружать машину предметами большого веса и малой стоимости нецелесообразно.

тов, проверяя на каждой из них, «влезают» ли они в заданный вес, и выбирая ту, для которой стоимость максимальна. Но при большом числе предметов это было бы затруднительно: число комбинаций неумеренно растет при увеличении числа предметов. Для метода же динамического программирования увеличение числа шагов не страшно: оно только приводит к пропорциональному возрастанию объема расчетов.

Таблица 13.5

S	i=6		i=5		i=4		i=3		i=2		i=1	
	x_i	W_i	x_i	W_i	x_i	W_i	x_i	W_i	x_i	W_i	x_i	W_i
0	**0**	0	0	0	0	0	0	0	0	0		
1	$\overline{0}$	0	**0**	0	0	0	0	0	0	0		
2	0	0	0	0	0	0	0	0	0	0		
3	0	0	0	0	0	0	0	0	0	0		
4	0	0	0	0	0	0	0	0	0	0		
5	0	0	0	0	0	0	0	0	0	0		
6	0	0	0	0	0	0	0	0	0	0		
7	0	0	0	0	0	0	0	0	1	10		
8	0	0	0	0	0	0	0	0	1	10		
9	0	0	0	0	0	0	0	0	1	10		
10	0	0	0	0	0	0	0	0	1	10		
11	0	0	0	0	0	0	1	15	0	15		
12	0	0	0	0	1	20	0	20	0	20		
13	0	0	0	0	1	20	0	20	0	20		
14	0	0	0	0	1	20	0	20	0	20		
15	0	0	0	0	1	20	0	20	0	20		
16	0	0	**1**	27	0	27	0	27	0	27		
17	0	0	$\overline{1}$	27	0	27	0	27	0	27		
18	0	0	1	27	0	27	0	27	0	27		
19	0	0	1	27	0	27	0	27	1	30		
20	1	34	0	34	0	34	0	34	0	34		
21	1	34	0	34	0	34	0	34	0	34		
22	1	34	0	34	0	34	0	34	0	34		
23	1	34	0	34	0	34	1	35	1	37		
24	1	34	0	34	0	34	1	35	1	37		
25	1	34	0	34	0	34	1	35	1	37		
26	1	34	0	34	0	34	1	35	1	37		
27	1	34	0	34	0	34	1	42	1	44		
28	1	34	0	34	**1**	47	**0**	47	0	47		
29	1	34	0	34	$\overline{1}$	47	$\overline{0}$	47	0	47		
30	1	34	0	34	1	47	0	47	0	47		
31	1	34	0	34	1	47	1	49	0	49		
32	1	34	0	34	1	54	0	54	0	54		
33	1	34	0	34	1	54	0	54	0	54		
34	1	34	0	34	1	54	0	54	0	54		
35	1	34	0	34	1	54	0	54	$\underline{1}$	57	$\underline{0}$	$\underline{57}$

§14. Задача динамического программирования в общем виде. Принцип оптимальности

Рассмотренные выше простейшие задачи динамического программирования дают понятие об общей идее метода: пошаговая оптимизация, проходимая в одном направлении «условно», в другом — «безусловно». Метод динамического программирования является очень мощным и плодотворным методом оптимизации управления; ему не страшны ни целочисленность решения, ни нелинейность целевой функции, ни вид ограничений, накладываемых на решение. Но в отличие от линейного программирования динамическое программирование не сводится к какой-либо стандартной вычислительной процедуре; оно может быть передано па машину только после того, как записаны соответствующие формулы, а это часто бывает не так-то легко.

В этом параграфе мы дадим нечто вроде «сводки советов начинающим» — как ставить задачи динамического программирования, в каком порядке их решать, как записывать и т. д.

Первый вопрос, на который нужно ответить ставящему задачу: какими параметрами характеризуется состояние управляемой системы S перед каждым шагом? От удачного выбора набора этих параметров часто зависит возможность успешно решить задачу оптимизации. В трех конкретных примерах, которые мы решали в предыдущем параграфе, состояние системы характеризовалось очень небольшим числом параметров: двумя координатами — в первом примере, одним числом — во втором и третьем. Но такие ультрапростые задачи не так уже часто встречаются на практике. Если, как это обычно и бывает, состояние системы описывается многими параметрами (так называемыми «фазовыми координатами»), то становится трудно перед каждым шагом перебрать все их варианты и для каждого найти оптимальное условное управление. Последнее еще больше затрудняется в случае, когда число возможных вариантов управления велико. В этих случаях над нами повисает, по меткому выражению Р. Беллмана, «проклятие многомерности»—бич не только метода динамического программирования, но и всех других методов оптимизации. Обычно задачи динамического программирования решаются не вручную, а на ЭВМ, од-

нако многие такие задачи не под силу даже современным машинам. Поэтому очень важно уметь правильно и «скромно» поставить задачу, не переобременяя ее лишними подробностями, упрощая елико возможно описание управляемой системы и вариантов управления. Так что в методе динамического программирования очень многое зависит от искусства и опыта исследователя.

Вторая задача после описания системы и перечня управлений — это членение на шаги (этапы). Иногда (на счастье) оно бывает задано в самой постановке задачи (например, хозяйственные годы в экономических задачах), но часто членение на шаги приходится вводить искусственно как мы сделали, например, в задаче 1 § 13. Если бы мы в этой задаче не ограничились самым примитивным случаем всего двух управлений («с» и «в»), то было бы удобнее членение на шаги произвести иначе, например, считая за «шаг» переход с одной прямой, параллельной оси ординат, на другую. Можно было бы вместо прямых рассмотреть окружности с центром в точке A или же другие кривые. Все такие способы выбора наивыгоднейшего пути неизбежно ограничивают выбор возможных направлений. Если за «шаги» считать переходы с одной прямой, параллельной оси ординат, на другую, то здесь множество возможных управлений не предусматривает «пути назад», т. е. с более восточной прямой на более западную. В большинстве задач практики такие ограничения естественны (например, трудно себе представить, чтобы наивыгоднейшая траектория космической ракеты, пущенной с Земли, на каких-то участках включала движение «назад», ближе к Земле). Но бывает и другая обстановка. Например, путь по сильно пересеченной местности («серпантинная» дорога в горах) часто «петляет» и возвращается ближе к исходному пункту. При постановке задачи динамического программирования, в частности при выборе системы координат и способа членения на шаги, должны быть учтены все разумные ограничения, накладываемые на управление.

Как быть с числом шагов m? С первого взгляда может показаться, что чем больше m, тем лучше. Это не совсем так. При увеличении m возрастает объем расчетов, а это не всегда оправдано. Число шагов нужно выбирать с учетом двух обстоятельств: 1) шаг должен

быть достаточно мелким для того, чтобы процедура оптимизации шагового управления была достаточно проста, и 2) шаг должен быть не слишком мелким, чтобы не производить ненужных расчетов, только усложняющих процедуру поиска оптимального решения, но не приводящих к существенному изменению оптимума целевой функции. В любом случае практики нас интересует не строго оптимальное, а «приемлемое» решение, не слишком отличающееся от оптимального по значению выигрыша W^*.

Сформулируем общий принцип, лежащий в основе решения всех задач динамического программирования (его часто называют «принципом оптимальности»):

Каково бы ни было состояние системы S перед очередным шагом, надо выбирать управление на этом шаге так, чтобы выигрыш на данном шаге плюс оптимальный выигрыш на всех последующих шагах был максимальным.

По-видимому, полное понимание этого принципа делается возможным (для лиц с обычным математическим развитием) только после рассмотрения ряда примеров, поэтому мы и приводим этот основной принцип не в начале главы (как это было бы естественно для математика), а лишь после решения ряда примеров.

А теперь сформулируем несколько практических рекомендаций, полезных начинающему при постановке задач динамического программирования. Эту постановку удобно проводить в следующем порядке.

1. Выбрать параметры (фазовые координаты), характеризующие состояние S управляемой системы перед каждым шагом.

2. Расчленить операцию на этапы (шаги).

3. Выяснить набор шаговых управлений x_i для каждого шага и налагаемые на них ограничения.

4. Определить, какой выигрыш приносит на *i-м* шаге управление x_i, если перед этим система была в состоянии S, т. е. записать «функции выигрыша»:

$$w_i = f_i(S, x_i). \qquad (14.1)$$

5. Определить, как изменяется состояние S системы S под влиянием управления x_i на i-м шаге: оно переходит в новое состояние

$$S' = \varphi_i(S, x_i). \qquad (14.2)$$

«Функции изменения состояния» (14.2) тоже должны быть записаны[1].

6. Записать основное рекуррентное уравнение ди намического программирования, выражающее услов ный оптимальный выигрыш $W_i(S)$ (начиная с i-го шага и до конца) через уже известную функцию $W_{i+1}(S)$:

$$W_i(S) = \max_{x_i} \{f_i(S, x_i) + W_{i+1}(\varphi_i(S, x_i))\}. \quad (14.3)$$

Этому выигрышу соответствует условное оптимальное управление на i-м шаге $x_i(S)$ (подчеркнем, что в уже известную функцию $W_{i+1}(S)$ надо вместо S подставить и з м е н е н н о е состояние $S' = \varphi_i(S, x_i)$).

7. Произвести условную оптимизацию последнего (m-го) шага, задаваясь гаммой состояний S, из кото рых можно за один шаг дойти до конечного состояния, вычисляя для каждого из них условный оптимальный выигрыш по формуле

$$W_m(S) = \max_{x_m} \{f_m(S, x_m)\} \quad (14.4)$$

и находя условное оптимальное управление $x_m(S)$, для которого этот максимум достигается.

8. Произвести условную оптимизацию (m — 1)-го, (m — 2)-го и т. д. шагов по формуле (14.3), полагая в ней i = (m — 1), (m —2), ..., и для каждого из ша гов указать условное оптимальное управление $x_i(S)$, при котором максимум достигается.

Заметим, что если состояние системы в начальный момент известно (а это обычно бывает так), то на первом шаге варьировать состояние системы не нужно — прямо находим оптимальный выигрыш для данного начального состояния S_0. Это и есть оптимальный выигрыш за всю операцию

$$W^* = W_1(S_0).$$

9. Произвести безусловную оптимизацию управле ния, «читая» соответствующие рекомендации на каж дом шаге. Взять найденное оптимальное управление на первом шаге $x_1^* = x_1(S_0)$; изменить состояние систе-

---

[1] Заметим, что аргументы функций (14.1), (14.2) в общем случае —не числа, а совокупности чисел (векторы).

мы по формуле (14.2); для вновь найденного состояния найти оптимальное управление на втором шаге x_2^* и т. д. до конца.

* * *

Сделаем несколько дополнительных замечаний общего характера. До сих пор мы рассматривали только аддитивные задачи динамического программирования, в которых выигрыш за всю операцию равен сумме выигрышей на отдельных шагах. Но метод динамического программирования применим также и к задачам с так называемым «мультипликативным» критерием, имеющим вид произведения:

$$W = \prod_{i=1}^{m} w_i \qquad (14.5)$$

(если только выигрыши w_i положительны). Эти задачи решаются точно так же, как задачи с аддитивным критерием, с той единственной разницей, что в основном уравнении (14.3) вместо знака «плюс» ставится знак умножения X:

$$W_i(S) = \max_{x_i} \{f_i(S, x_i) \times W_{i+1}(\varphi_i(S, x_i))\}. \qquad (14.6)$$

- В заключение — несколько слов о так называемых «бесконечношаговых» задачах динамического программирования. На практике встречаются случаи, когда планировать операцию приходится не на строго определенный, а на неопределенно долгий промежуток времени, и нас может интересовать решение задачи оптимального управления безотносительно к тому, на каком именно шаге операция заканчивается. В таких случаях бывает удобно рассмотреть в качестве модели явления бесконечношаговый управляемый процесс, где не существует «особенного» по сравнению с другими последнего шага (все шаги равноправны). Для этого, разумеется, нужно, чтобы функции f_i выигрыша и функции φ_i изменения состояния не зависели от номера шага. Интересующегося этим случаем читателя отошлем к руководству [10]. Вообще, для более подробного ознакомления с методом динамического программирования полезно обратиться к руководствам [6, 10, 11, 7].