Златопольский Дмитрий Михайлович
Кандидат технических наук, доцент кафедры информатики и прикладной математики Московского городского педагогического университета.

Что такое динамическое программирование

На примере конкретных задач в статье разъясняются основные принципы динамического программирования – особого метода поиска оптимальных решений для многоэтапных операций.

В № 9 журнала «Потенциал» [1] была опубликована статья «Задача о джипе», в которой описывался один из методов разработки алгоритма решения задач – «метод отрабатывания назад». Его особенность в том, что анализ и принятие решения начинается «от конца» цели, после чего они проводятся по направлению к начальной постановке задачи, а затем, если эти действия обратимы, происходит движение обратно, от постановки к цели. Отмечалось, что он положен в основу динамического программирования[1] – особого метода поиска оптимальных решений, специально приспособленного к так называемым «многошаговым», или «многоэтапным», операциям. При этом многошаговость либо отражает реальное протекание процесса принятия решений во времени, либо вводится в задачу искусственно за счёт разделения процесса принятия однократного решения на отдельные этапы.

Как указывается в [2], полное понимание основных положений динамического программирования, как правило, возможно только после рассмотрения ряда примеров, поэтому разберём несколько задач, в которых рассматриваются «многошаговые», т. е. представляющие собой последовательность «шагов» («этапов»), операции.

Задача 1. Нужно проложить путь, соединяющий два пункта – *A* и *B* (рис. 1), второй из которых лежит к северо-востоку от первого. Прокладка пути состоит из ряда шагов, и на каждом шаге мы можем двигаться либо строго на север, либо строго на восток, т. е. любой путь из

[1] Термин «программирование» в названии метода происходит от слова «программа» в значении *план действий* и не связан с понятием *компьютерная программа*. Поэтому динамическое программирование иначе называют динамическим планированием. Происхождение же термина «динамический» связано с использованием метода в задачах принятия решений через фиксированные промежутки времени.

A в B представляет собой ступенчатую ломаную линию, отрезки которой параллельны одной из координатных осей. Затраты на прокладку каждого из таких отрезков известны. Требуется проложить такой путь из A в B, при котором суммарные затраты минимальны.

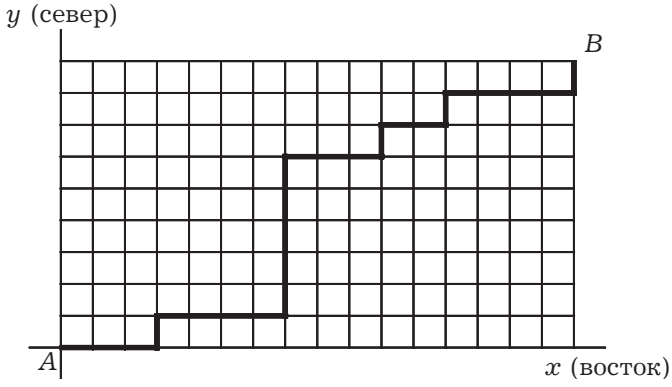


Рис. 1

Задача 2. Имеется определенный набор предметов – $П_1$, $П_2$, …, $П_n$ (каждый в единственном экземпляре); известны их массы – M_1, M_2, …, M_n и стоимости – C_1, C_2, …, C_n. Грузоподъёмность машины равна Q. Спрашивается: какие из предметов нужно взять в машину, чтобы их суммарная стоимость (при суммарной массе $\le Q$) была максимальна?

Задача 3. Имеется какой-то объём денежных средств Q, который должен быть распределён между предприятиями $П_1$, $П_2$, …, $П_n$. Каждое из предприятий $П_i$ при вложении в него каких-то средств в объёме x приносит доход, зависящий от x, т. е. представляющий собой какую-то функцию $f_i(x)$. Все функции $f_i(x)$ (i = 1, 2, ..., n) заданы (разумеется, эти функции – неубывающие). Спрашивается: сколько средств нужно выделить каждому предприятию, чтобы в сумме они дали максимальный доход?

В первой задаче операцией является строительство пути из точки A в точку B. Здесь шаги выделены уже в условии задачи – шагом является прокладывание очередного отрезка пути.

В постановке второй задачи нет упоминания о времени. Но процесс загрузки машины можно представить как состоящий из n шагов, считая за первый шаг принятие решения о том, брать или не брать первый предмет, за второй – то же относительно второго предмета и т. п.

В третьей задаче также можно операцию распределения средств мысленно развернуть во времени в какой-то последовательности и рассматривать решение вопроса о вложении средств в предприятие $П_1$ как первый шаг, в предприятие $П_2$ – как второй и т. д.

Как можно решать подобного рода задачи? Очевидно, что это можно сделать по-разному.

Можно перебрать все возможные варианты решения и выбрать среди них лучший. Например, в первой задаче можно рассмотреть все возможные варианты пути и выбрать тот, на котором затраты минимальны. А можно выбирать лучший вариант пути шаг за шагом, на каждом этапе расчёта оптимизировать только один шаг. Обычно второй

способ оптимизации оказывается проще, чем первый, особенно при большом числе шагов[1].

Такая идея постепенной, пошаговой оптимизации и лежит в основе метода динамического программирования. На первый взгляд она может показаться довольно тривиальной. В самом деле, чего, казалось бы, проще: если трудно оптимизировать процесс в целом, надо разбить его на ряд шагов и оптимизировать каждый шаг. Это сделать нетрудно – надо выбрать на этом шаге такое решение (будем называть решение – «управлением»), 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, управление на каждом шаге должно выбираться дальновидно, с учётом его последствий в будущем. Что толку, если мы выберем на данном шаге управление, при котором эффективность *этого* шага максимальна, если он приведёт нас к проигрышу на последующих шагах?

А как выбрать оптимальное управление на том или ином шаге? Ведь на каждом шаге (естественно, кроме первого) система может находиться в общем случае в нескольких состояниях (в различных точках местности, на которой сооружается путь из A в B, с различным весом груза, который ещё можно взять в машину, и т. п.), а в каждом из этих состояний в общем случае возможны несколько вариантов управления (строить очередной участок пути в северном или в восточном направлении, брать или не брать i-й предмет, вкладывать в i-е предприятие один из возможных объёмов средств или не вкладывать вообще). Какое же решение является оптимальным? Ведь мы не знаем даже, чем закончился предыдущий шаг.

Так как ответить на этот вопрос непросто, давайте решим другую задачу: определим, какое управление является лучшим для *каждого* из состояний системы на том или ином шаге (иными словами, сделаем разные предположения о том, чем закончился *предыдущий* шаг). Такое управление назовём «условным оптимальным управлением для этого состояния», а выигрыш, который даёт это управление, – «условным оптимальным выигрышем» («условное» потому, что оно выбирается исходя из условия, что предыдущий шаг кончился так-то и так-то).

Выбрать такой вариант управления можно, сравнив все возможные управления по следующему критерию: *сумма выигрыша от реализации того или иного управления на данном шаге и условного оптимального выигрыша в состоянии системы, в которое она перейдёт в результате этого управления* (помните о необходимости на каждом шаге принимать «дальновидные» решения?)[2].

---

[1] При большом числе шагов в ряде случаев решение первым способом вообще не может быть реализовано. В задаче 1, например, общее число возможных вариантов пути равно числу сочетаний из $(N_1 + N_2)$ по N_1, где N_1 – число отрезков от точки A до точки B в восточном направлении, N_2 – в северном. При $N_1 = 10$ и $N_2 = 10$ общее число вариантов равно 184756. При увеличении N_1 и N_2 это число существенно возрастает (при $N_1 = 30$ и $N_2 = 30$ компьютер, выполняющий 10000000 операций в секунду, рассмотрит все возможные варианты пути за более чем 10 000 лет!).

[2] Не вызывает сомнения тот факт, что именно по такому критерию пришлось бы выбирать оптимальный вариант управления в случае, когда какое-то состояние было бы *исходным*.

Очевидно, что для использования такого критерия необходимо знать условные оптимальные выигрыши для всех состояний системы на *следующем* шаге. А из этого, в свою очередь, следует очень важный вывод: определение условных оптимальных параметров надо начинать с последнего шага, затем найти их для предпоследнего шага и т. д.

Для последнего шага сделать это не так уж сложно. Ведь для каждого состояния перед последним шагом известно, какое управление необходимо применить, – только то, которое приведёт к конечному состоянию системы (путь должен закончиться в точке *B*, автомобиль должен быть полностью загружен, все средства должны быть вложены). Такое управление будем называть «вынужденным».

Проиллюстрируем принципы динамического программирования на примере решения задачи 1.

Разделим расстояние от *A* до *B* в восточном направлении, скажем, на 7 частей, а в северном – на 5 (в принципе дробление может быть сколь угодно мелким). Тогда любой путь из *A* в *B* состоит из 7 + 5 = 12 отрезков, направленных на восток или на север (рис. 2). Проставим на каждом из отрезков число, выражающее (в каких-то условных единицах) стоимость прокладки пути по этому отрезку. По требованию условия задачи необходимо выбрать такой путь из *A* в *B*, при котором сумма чисел, стоящих на отрезках, минимальна.

Будем рассматривать сооружаемый путь как управляемую систему, перемещающуюся под влиянием управления (строительства очередного участка) из начального состояния *A* в конечное *B*. Состояние системы перед началом каждого шага будем характеризовать двумя координатами: восточной x и северной y, обе целочисленные ($0 \le x \le 7$, $0 \le y \le 5$). Для каждого из состояний системы (узловой точки прямоугольной сетки на рис. 2) мы должны найти условное оптимальное управление: идти нам на север или на восток? Выбирается это управление так, чтобы *стоимость всех оставшихся до конца шагов (включая данный) была минимальна*. Эту стоимость мы договорились называть «условным оптимальным выигрышем» для данного состояния системы (хотя в данном

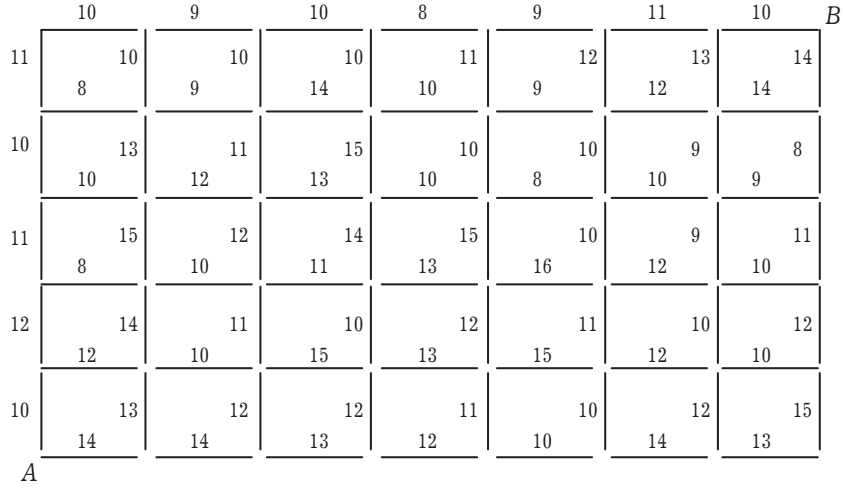


Рис. 2

случае это, конечно, не «выигрыш», а «проигрыш»).

Как указывалось выше, определение условных оптимальных параметров (управления и выигрыша) надо начинать с последнего, 12-го шага. Рассмотрим отдельно правый верхний угол нашей прямоугольной сетки (рис. 3). Где мы можем находиться после 11-го шага? Только 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_1, у нас нет выбора (управление вынужденное): надо идти на восток, и это обойдётся нам в 10 единиц. Запишем это число 10 в кружке у точки B_1, а оптимальное управление покажем короткой стрелкой, исходящей из B_1 и направленной на восток. Для точки B_2 управление тоже вынужденное (на север), затраты до конца равны 14, мы их запишем в кружке у точки B_2. Таким образом, условная оптимизация последнего шага сделана, условный оптимальный выигрыш для обоих состояний этого шага найден и записан в соответствующем кружке.

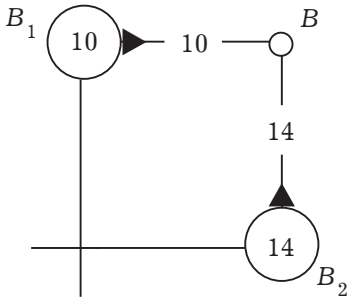


Рис. 3

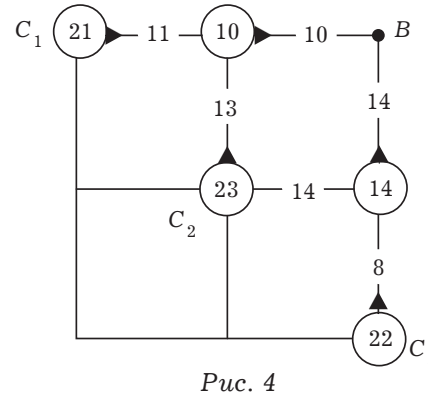


Рис. 4

Теперь давайте оптимизировать предпоследний (11-й) шаг. После предпредпоследнего (10-го) шага мы могли оказаться в одной из точек C_1, C_2, C_3 (рис. 4). Найдём для каждой из них условное оптимальное управление и условный оптимальный выигрыш. Для точки C_1 управление вынужденное – идти на восток; обойдётся это нам до конца в 21 единицу (11 на данном шаге плюс 10 – условный оптимальный выигрыш для точки B_1, записанный в кружке). Число 21 записываем в кружок 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 – ещё 14, всего 28 единиц. Если пойдём на север, затратим 13 + 10 = 23 единицы. Значит, условное оптимальное управление в точке C_2 – идти на север (внимательный читатель, очевидно, заметил, что найдено это управление по критерию, о котором говорилось выше). Отмечаем его стрелкой, а число 23 записываем в кружок у точки C_2. Для точки C_3 управление снова вынужденное – идти на север; обойдётся это до конца в 22 единицы (ставим стрелку на север, число 22 записываем в кружок у точки C_3).

Аналогично, «пятясь» от предпоследнего шага назад, найдём для каждой точки условное оптимальное

управление, которое обозначим стрелкой, и условный оптимальный выигрыш, который запишем в кружке. Вычисляется он так: расход на данном шаге складывается с *уже оптимизированным* расходом, записанным в кружке, куда ведёт стрелка. Таким образом, на каждом шаге мы оптимизируем только этот шаг, а следующие за ним уже оптимизированы.

Конечный результат показан на рис. 5. Из него видно, что в какой бы из узловых точек мы ни находились, мы знаем, куда идти (стрелка) и во что нам обойдётся путь до конца (число в кружке), а в кружке у точки *A* записан оптимальный выигрыш (минимальные затраты) на всё сооружение пути из *A* в *B*!

Теперь остаётся построить путь, ведущий из *A* в *B*, самым дешёвым способом. Для этого надо только «слушаться стрелок», т. е. прочитать, что они предписывают делать на каждом шаге. Такая оптимальная траектория пути отмечена на рис. 5 оттенёнными кружками.

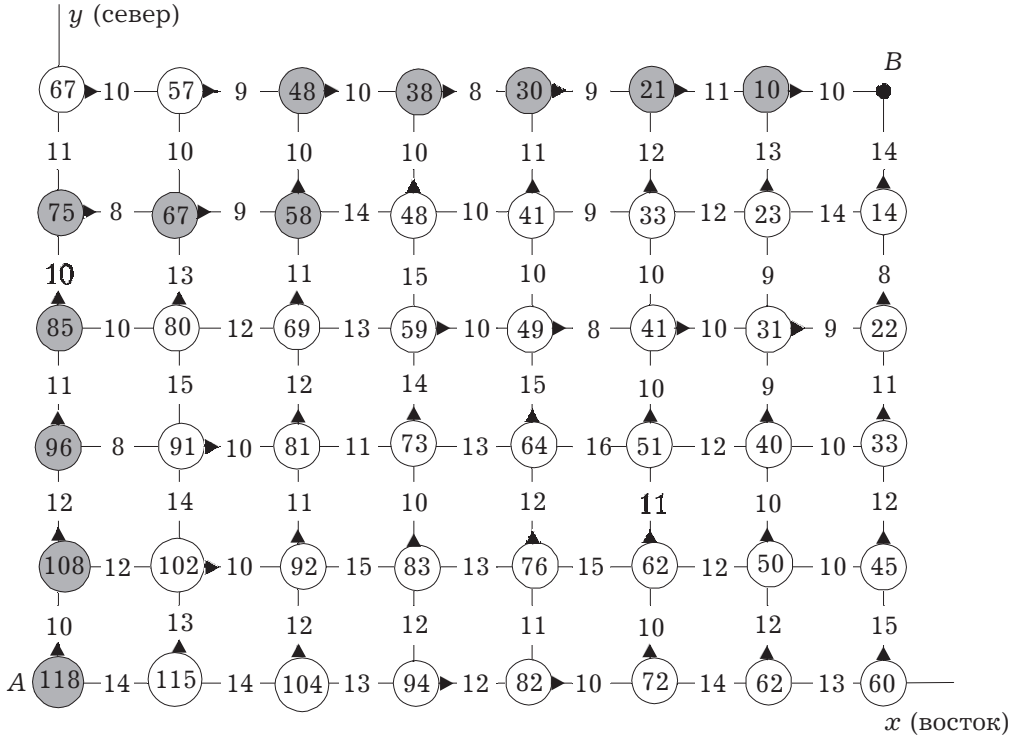


Рис. 5

Из рис. 5 следует, что оптимальный выигрыш (минимальные затраты) на сооружение пути из *A* в *B* равен 118, а достигается он при следующем наборе отрезков пути:

север, север, север, север, восток, восток, север, восток, восток, восток, восток, восток.

Интересно, а какой результат был бы получен, если бы мы решали задачу наивным способом, о котором говорилось выше, т. е. выбирая на каждом шаге, начиная с первого, самое выгодное *для этого шага* направление? Из рис. 2 видно, что таким способом мы получим путь, состоящий из отрезков:

север, север, восток, восток, восток, восток, север, восток, восток, восток, север, север.

Подсчитаем расходы для такого пути. Они будут равны:

$$10 + 12 + 8 + 10 + 11 + 13 + 15 + 8 + \\ + 10 + 9 + 8 + 14 = 128,$$

что больше, чем найденное методом динамического программирования значение 118. В данном случае разница не очень велика, но в других случаях она может быть существеннее.

Отметим также ещё один момент. В ходе условной оптимизации можно столкнуться со случаем, когда оба возможных управления для какого-то состояния (какой-то точки на плоскости) являются оптимальными, т. е. приводят к одинаковому расходу средств от этой точки до конца. Например, в точке с координатами (1; 4) оба управления являются оптимальными и дают расход до конца, равный 67. Из них можно выбрать любое (в нашем случае мы выбрали «восток»; с тем же успехом мы могли выбрать «север» – общий выигрыш был бы тем же). Такие случаи неоднозначного выбора оптимального управления часто встречаются при использовании метода динамического программирования.

Проверь себя

Справа изображён треугольник из чисел. Найдите максимальную сумму чисел, расположенных на пути, начинающемся в верхней точке треугольника и заканчивающемся на его основании.

						8								
					5		7							
				8		2		1						
			2		7		5		5					
		7		10		1		9		7				

Литература

1. *Златопольский Д.М.* Задача о джипе. // Потенциал. – 2013. – № 9.

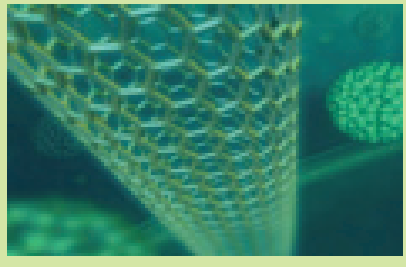
2. *Вентцель Е.С.* Исследование операций: задачи, принципы, методология. – М.: Наука, 1988.

3. *Шилов Н.В., Шилова С.О.* Кирпичи и динамическое программирование. // Потенциал. – 2012. – № 9.

4. *Златопольский Д.М.* Программирование: типовые задачи, алгоритмы, методы. – М.: БИНОМ. Лаборатория знаний, 2007.

Калейдоскоп Калейдоскоп Калейдоскоп

Первые нанотрубки



Сенсационное открытие в конце XX века углеродных нанотрубок подтвердило известный афоризм: «Новое – это хорошо забытое старое». Ведь первые нанотрубки были обнаружены в природном минерале имоголите задолго до углеродных аналогов в середине того века. Этот минерал имеет уникальное внутреннее строение, представляющее собой плотно упакованные пучки длинных тонких трубок. В отличие от углеродных нанотрубок, которые бывают однослойными, многослойными и имеют различные диаметры, имоголит обладает однозначной структурой с диаметром трубок 2 нм.