Математические модели и методы принятия оптимальных решений

Береснев Владимир Леонидович

ФИТ НГУ

Авторы: Кондренко К.П., Воробьёв А.Д.

2025 г.

Оглавление

1 Введение

Определение

Организованные системы — системы, в которых решения принимаются «сознательно». Примеры таких систем: люди, промышленные предприятия, магазины.

Примеры задач:

- где построить магазин, чтобы получать наибольшую прибыль?
- сколько производить деталей на заводе, чтобы отношение между доходом и выручкой было наибольшим?

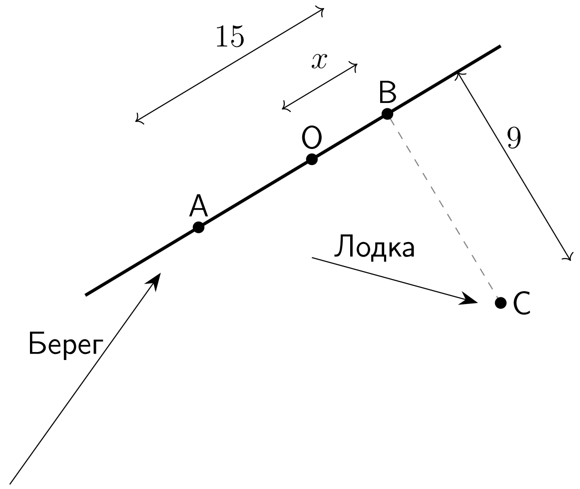
Примерно до второй мировой войны все сложные решения принимались лишь на основе опыта и здравого смысла. Однако позже появились сложные системы, в которых опыта и здравого смысла оказалось недостаточно. Тогда же появилась **идея** рассматривать числовые характеристики систем для принятия решения, при этом должны использоваться *математические модели* — упрощённые, но адекватные описания реальной жизни.

1.1 Построение математических моделей

Математические модели строятся на основании *исходных данных* — конкретных проблем в конкретных жизненных ситуациях.

Алгоритм (построения математических моделей)

1. Нужно понять, **что будем оптимизировать?** По каким критериям будем оценивать решения? Например, если нас спрашивают, где построить магазин, то нам нужно понять, как выбрать для этого место. Нужно, чтобы была максимальная прибыль или чтобы была наибольшая удалённость от конкурентов? Если мы закупаем детали для предприятия, то нужно узнать, хочется ли нам наибольшую прибыль или же минимальные издержки. А может нам важно количество произведённой продукции?
2. Нужно понять, **какие характеристики существенно влияют на оптимизируемую характеристику**. Например, если наша оптимизируемая характеристика — это прибыль предприятия, то нужно определить, из чего складываются выручка и затраты.
3. **Формулировка задачи** с точным указанием всех характеристик. Каждая из характеристик должна относиться либо к *переменным*, либо к *параметрам*. Первые могут меняться (обозначаются они через $x, y, z, \dots$), а вторые — это константы (обозначаются они через



$a, b, c, \dots$). Например, переменные — это выручка и затраты предприятия, а параметры — это площадь помещения, количество станков, количество работников.

4. Выбор всех обозначений и **математическая запись** с учётом ограничений и требований для переменных.
5. ~~Понимание, что изначальная проблема состоит в другом, не учтены такие-то параметры, а значит нужно вернуться к самому началу.~~

Замечание

Алгоритм примерный, поэтому на практике этапы и их содержание могут отличаться от того, что написано в алгоритме.

Задача (о лодке)

Для примера рассмотрим следующую задачу: ваш знакомый плывёт на лодке, и ему нужно попасть в определённую точку на берегу. Он спрашивает вас, куда ему нужно причалить.

Характеристики рассматриваемой системы

- C — текущее местоположение лодки;
- A — точка, в которую нужно попасть;
- B — ближайшая к лодке точка берега;
- O — точка причаливания;
- $AB =$ 15 км;
- $BC =$ 9 км (расстояние от лодки до берега);
- $v_{\text{по суше}} = 5$ км/ч;
- $v_{\text{лодки}} = 4$ км/ч;
- $OB = x$;
- $0 \le x \le 15$, потому что иначе решение точно не будет оптимальным;
- течения воды нет, то есть скорость течения равна нулю;
- ~~цвет лодки не существенен.~~

Решение

Будем оптимизировать время причаливания t

$$t = \underbrace{\frac{OC}{v_{\text{по воде}}}}_{\text{движение по воде}} + \underbrace{\frac{OA}{v_{\text{по суше}}}}_{\text{движение по суше}} = \boxed{\frac{\sqrt{x^2+81}}{4} + \frac{15-x}{5} \to \min_x}$$

Для решения задачи найдём нули производной

$$\frac{\mathrm{d}t}{\mathrm{d}x} = \frac{2x}{8\sqrt{x^2+81}} - \frac{1}{5} = 0,$$

$$\frac{x}{4\sqrt{x^2+81}} = \frac{1}{5},$$

$$\frac{x^2}{16(x^2+81)} = \frac{1}{25},$$

$$25x^2 = 16x^2 + 16 \cdot 81,$$

$$9x^2 = 16 \cdot 81,$$

$$x^2 = 16 \cdot 9,$$

$$x_1 = -12, \qquad x_2 = 12.$$

Решение $x = -12$ не подходит ввиду ограничения выше, а вот $\boxed{x = 12}$ является ответом.

1.2 Экстремальные задачи

Определение

Экстремальная задача формулируется следующим образом

1. Есть функция $f(x)$, значение которой нужно оптимизировать;
2. Есть набор ограничений $g_1(x) \le b_1, g_2(x) \le b_2, \ldots$;
3. $x \in X$, X — *множество всех решений*,

при этом нужно найти

$$\min_{x \in X} f(x) \quad \text{или} \quad \max_{x \in X} f(x).$$

Замечание

Ограничения $\{g_i(x)\}$ могут быть какими угодно.

Определение

Допустимые решения — множество всех значений $x \in X$, которые удовлетворяют всем ограничения $\{g_i\}$.

Определение

Допустимое решение x^* называется *оптимальным решением задачи*, если

$$\forall x \in X \qquad f(x^*) \ge f(x)$$

или то же самое

$$f(x^*) = \max_{x \in X} f(x).$$

Вторая запись означает, что мы ищем максимальное значение $f(x)$, перебирая все x из множества X.

Пример

Пусть наши исходные данные это

$$f(x) = c_1 x_1 + c_2 x_2 \to \max_x,$$

$$g(x) = a x_1 + b x_2 \leq d,$$

$$x_1 \geq 0, \quad x_2 \geq 0,$$

а задача состоит в поиске оптимальных значений x_1 и x_2.

Определение

Будем говорить, что есть *общая задача* $\mathcal{P}$, а $p \in \mathcal{P}$ — конкретная задача, в которой у всех параметров есть конкретные значения, например, если бы в задаче выше c_1, c_2, a, b, d были бы конкретными числами. То есть *общая задача* — это множество конкретных задач.

Определение

Длина входа задачи — это количество ячеек в памяти, которое занимает задача с допущением, что каждое число занимает в памяти ровно одну ячейку. Будем обозначать это $|p|$.

1.3 Алгоритмы и их трудоёмкость

Определение

Элементарные операции — арифметические операции и операции сравнения.

Определение

Трудоёмкость алгоритма A решения задачи $p \in \mathcal{P}$ — это количество элементарных операций, используемых в этом алгоритме. Будем обозначать это $T_A(p)$.

Замечание

Чем больше $|p|$, тем больше $T_A(p)$, поэтому целесообразно оценивать трудоёмкость так

$$T_A(p) \leq f_A\big(|p|\big).$$

Определение

Алгоритм A будем называть *«эффективным»* (*полиномиальным*), если

$$f_A\big(|p|\big) = \underbrace{C}_{const} \cdot |p|^k.$$

Примерами задач, для которых существуют «хорошие» алгоритмы, являются, например математические задачи, в которых x — множество векторов (линейное и нелинейное программирование), и задачи комбинаторики (например, перестановки).

Определение

Задача линейного программирования определяется следующим образом

$$\sum_{j=1}^{n} c_j x_j \to \max_{(x_j)},$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = 1 \dots m,$$

$$x_j \in \{0, 1, 2, \dots\}, \quad j = 1 \dots n.$$

Распространённый частный случай: $x_j \in \{0, 1\}$.

1.4 Примеры задач

1.4.1 Задача о машине

Задача (о машине)

На некотором острове есть машина; для работы машины нужны детали, которые могут изнашиваться. В скором времени нужно будет вызвать самолёт, который сможет доставить необходимые детали, однако максимальная масса груза ограничена. Сколько деталей каждого типа нужно заказать?

Математическая модель

1. Что будем оптимизировать? Ответ: машина должна работать максимальное время.

2. Что существенно влияет на оптимизируемую характеристику?
Пусть $i = 1 \dots n$ — вид детали.

Параметры (то есть фиксированные значения)
- $m = \{m_i\}_{i=1}^{n}$ — масса деталей;
- $t = \{t_i\}_{i=1}^{n}$ — срок службы деталей (часов, дней, месяцев — неважно);
- $a = \{a_i\}_{i=1}^{n}$ — количество работающих деталей в машине;
- M — максимальная масса посылки.

Переменные (то что может меняться)
- $x = \{x_i\}_{i=1}^{n}$ — сколько нужно взять деталей в посылку.

3. Математическая формулировка задачи
Пусть T — время работы машины, тогда

$$T = \min_{i=1\dots n} \big((x_i + a_i) \cdot t_i\big) \to \max_{(x_i)},$$

$$\sum_{i=1}^{n} m_i x_i \le M,$$

$$x_i \ge 0.$$

Первое выражение говорит о том, что мы максимизируем время работы машины T при различных (x_i), то есть нам нужно подобрать такие значения $x_1, x_2, \ldots, x_n$, при которых время работы будет максимальным. Выражение (x_i+a_i) — это то, сколько деталей вида i будет после прилёта самолёта с посылкой. Если учесть, что каждая деталь типа i имеет срок службы t_i, то все детали данного типа проработают $(x_i + a_i) \cdot t_i$. Ясно, что машина перестанет работать, когда какой-то вид деталей в ней отработает свой срок службы, значит время работы машины — это минимум из времён работы всех деталей.

Второе выражение отражает ограничение задачи, которое состоит в том, что мы не можем заказать груз большей массы, чем установленная максимальная масса посылки.

Третье выражение говорит о том, что количество заказываемых деталей не может быть отрицательным — оно и понятно.

1.4.2 Задача о салфетках

Задача (о салфетках)

Пусть есть некоторое кафе, в которое каждый день ходят люди, при этом известно, сколько человек посещает кафе каждый день недели. Каждому гостю на день выдают салфетку, которую вечером стирают. Салфетки можно стирать с помощью <u>быстрой</u> и <u>медленной</u> стирок. Первая — дорогая, но работает условно моментально, вторая — дешёвая, но выдача постиранных салфеток происходит лишь через день. Как сэкономить деньги на стирке так, чтобы всем посетителям всегда хватало салфеток?

Математическая модель

1. Что будем оптимизировать? Ответ: нужно минимизировать затраты на стирку.

2. Что существенно влияет на оптимизируемую характеристику?

Пусть $i = 1 \ldots 7$ — день недели.

Параметры

- c_1 — цена быстрой стирки;
- c_2 — цена медленной стирки;
- T — общее количество салфеток в кафе;
- p_i — количество гостей в i-ый день недели;

Переменные

- x_i — количество салфеток, отданных в <u>быструю</u> стирку в i-ый день недели;
- y_i — количество салфеток, отданных в <u>медленную</u> стирку в i-ый день недели.

3. Математическая формулировка задачи

Пусть C — затраты на стирку за неделю, тогда

$$C = \sum_{i=1}^{7}(c_1 x_i + c_2 y_i) \to \min_{(x_i),(y_i)},$$

$$x_i + y_i = p_i,$$

$$x_i \geq 0, \quad y_i \geq 0,$$

$$\begin{cases} T - y_7 \geq p_1, \\ T - y_1 \geq p_2, \\ T - y_2 \geq p_3, \\ \dots \\ T - y_5 \geq p_6, \\ T - y_6 \geq p_7. \end{cases}$$

Первое выражение означает, что мы минимизируем затраты на стирку в неделю при различных $x_1, x_2, \dots, x_7$ и $y_1, y_2, \dots, y_7$. Выражение $(c_1 x_i + c_2 y_i)$ означает затраты на стирку в i-ый день недели.

Второе выражение означает, что каждый день все грязные салфетки отправляются в стирку, то есть то, что не остаётся не постиранных.

Последние семь неравенств означают, что всем посетителям всегда хватает салфеток.

1.4.3 Задача раскроя

Задача (раскроя)

Вашему знакомому сантехнику нужно определённое количество коротких труб, однако в магазине можно купить только длинные. Сколько нужно купить длинных труб, чтобы их можно было раскроить на короткие? Пусть нужны

- 10 труб по 6 м,
- 15 труб по 5 м,
- 26 труб по 4 м,

а в магазине продаются лишь трубы по 13 м.

Математическая модель

1. Что будем оптимизировать? Ответ: нужно минимизировать число покупаемых труб по 13 м.

2. Что существенно влияет на оптимизируемую характеристику?

Рассмотрим все варианты, как можно раскроить длинную трубу на короткие

№	6 м	5 м	4 м
1	2	0	0
2	1	1	0
3	1	0	1
4	0	2	0
5	0	1	2
6	0	0	3

То есть раскроить длинную трубу на короткие можно 6 разными способами. Например, на две трубы длиной 6 метров (первая строка таблицы), на одну трубу длиной 6 метров и на одну трубу длиной 5 метров (вторая строка таблицы) и так далее.

Параметры

- x_i — количество длинных труб, раскроенных i-ым способом;

3. Математическая формулировка задачи

Пусть T — количество изрезанных длинных труб, тогда

$$T = \sum_{i=1}^{6} x_i \to \min_{(x_i)},$$

$$\begin{cases} 2x_1 + x_2 + x_3 \geq 10, \\ x_2 + 2x_4 + x_5 \geq 15, \\ x_3 + 2x_5 + 3x_6 \geq 26. \end{cases}$$

Первое выражение означает, что мы минимизируем количество раскроенных длинных труб, таким образом минимизируя затраты на их покупку.

Последние три неравенства означают, что после раскроя длинных труб мы получили достаточное количество коротких: первое неравенство для труб длиной 6 метров, второе — 5 метров, третье — 4 метра.

Замечание

В общем случае задача раскроя является NP-полной.

1.5 Свойства оптимизационных задач

Утверждение (от максимума к минимуму)

Пусть есть задача, в которой нужно максимизировать значение функции $f(x)$ при $x \in X$, тогда

$$\max_{x \in X} f(x) = \max_{x \in X} \Big((-1) \cdot (-f(x)) \Big) = \underbrace{-\min_{x} \big(-f(x)\big)}_{\text{новая задача}}.$$

То есть задача нахождения максимума функции $f(x)$ эквивалентна задаче нахождения минимума функции $\big(-f(x)\big)$.

Утверждение (оценка сверху, релаксированные задачи)

Если $X \subseteq X'$, то

$$\max_{x \in X} f(x) \leq \max_{x \in X'} f(x).$$

Определение

Пусть есть две общие задачи: $(\mathcal{P}) \max x \in X f(x)$ и $(\mathcal{Q}) \max_{y \in Y} g(y)$. Будем говорить, что задача $\mathcal{P}$ *сводится к задаче* $\mathcal{Q}$, если $\forall p \in \mathcal{P}\ \forall q \in \mathcal{Q}$

1. существует полиномиальный алгоритм A_1, который переводит входные данные задачи p во входные данные задачи q;
2. существует полиномиальный алгоритм A_2, с помощью которого можно из оптимального решения y^* задачи q построить оптимальное решение x^0 задачи p,

то есть

$$\begin{aligned} p &\xrightarrow{A_1} q, \\ x^0 &\xleftarrow{A_2} y^*. \end{aligned}$$

Остаётся вопрос: как понять, что построенное решение x^0 — оптимальное?

Замечание

Везде далее всегда будем оптимизировать именно $\max$, а не $\min$.

Утверждение (сведение к другой задаче)

Пусть есть две задачи: $\mathcal{P}$ и $\mathcal{Q}$, при этом

1. x^0 — допустимое решение задачи $\mathcal{P}$,
2. y^* — оптимальное решение задачи $\mathcal{Q}$,
3. $f(x^0) \geq g(y^*)$,
4. $\forall x \in X \; \exists y \in Y \quad f(x) \leq g(y)$,

тогда x^0 — оптимальное решение задачи $\mathcal{P}$.

Доказательство

Пусть задача $\mathcal{P}$ имеет оптимальное решение x^*. По четвёртому условию для x^* существует некоторый y^0 такой, что

$$f(x^*) \leq g(y^0). \tag{*}$$

По второму условию y^* является оптимальным решением задачи $\mathcal{Q}$, в то время как y^0 — допустимое (необязательно оптимальное) решение задачи $\mathcal{Q}$, значит верно следующее

$$g(y^*) \geq g(y^0). \tag{**}$$

Составим цепочку неравенств

$$f(x^0) \overset{(3)}{\geq} g(y^*) \overset{(**)}{\geq} g(y^0) \overset{(*)}{\geq} f(x^*).$$

Таким образом имеем неравенство

$$f(x^0) \geq f(x^*),$$

хотя x^* — оптимальное решение задачи $\mathcal{P}$. Значит $f(x^0) = f(x^*)$, то есть x^0 тоже является оптимальным решением.

Пример

$(\mathcal{P})$

$$\sum_{j=1}^{n} c_j x_j \to \max_{(x_j)},$$

$$\sum_{j=1}^{n} a_j x_j \leq b, \tag{1}$$

$$x_1 + x_2 = d, \tag{2}$$

$$x_j \geq 0, \quad j = 1 \ldots n. \tag{3}$$

Заметим, что x_1 можно выразить через x_2 (и наоборот). Рассмотрим другую задачу

$(\mathcal{Q})$

$$c_1(d-y_2)+\sum_{j=2}^{n} c_j y_j \to \max_{(y_j)},$$

$$a_1(d-y_2)+\sum_{j=2}^{n} a_j y_j \le b,$$

$$y_j \ge 0, \quad j=1\dots n,$$

$$y_2 \le d.$$

Покажем, что задача $\mathcal{P}$ сводится к задаче $\mathcal{Q}$.

Доказательство

Пусть $y^* = \begin{pmatrix} y_1^* & \dots & y_n^* \end{pmatrix}$ — это оптимальное решение задачи $\mathcal{Q}$. Будем строить решение x^0 задачи $\mathcal{P}$ следующим образом

$$x_j^0 = \begin{cases} d-y_2^*, & j=1, \\ y_j^*, & j>1; \end{cases}$$

то есть $x^0 = \begin{pmatrix} d-y_2^* & y_2^* & y_3^* & \dots & y_n^* \end{pmatrix}$.

Допустимость решения

Покажем, что x^0 — допустимое решение задачи $\mathcal{P}$. Для этого нужно показать, что оно удовлетворяет всем ограничениям. Заметим, что без условия $y_2 \le d$ значение x_1 может быть меньше нуля, а значит решение x^0 точно было бы не допустимым (3).

1. Проверим (2)

$$x_1^0 + x_2^0 = d - y_2^* + y_2^* = d.$$

2. Проверим (1)

$$\begin{aligned}\sum_{j=1}^{n} a_j x_j^0 &= a_1 x_1^0 + a_2 x_2^0 + \dots + a_n x_n^0 \\ &= a_1(d-y_2^*) + a_2 y_2^* + \dots + a_n y_n^* \\ &= a_1(d-y_2^*) + \sum_{j=2}^{n} a_j y_j^* \le b.\end{aligned}$$

То есть x^0 удовлетворяет всем ограничениям, значит x^0 — допустимое решение задачи $\mathcal{P}$.

Оптимальность решения

Мы показали, что x^0 — допустимое решение задачи $\mathcal{P}$. Осталось показать, что оно оптимальное. Для этого будем использовать сведение к другой задаче. Для этого нужно проверить, что выполняются все условия для его использования.

1. Выполнимость условия $f(x^0) \ge g(y^*)$ следует из того, что при подстановке, использованной при проверке (1), получится равенство

$$f(x^0) = g(y^*).$$

2. Выполнимость условия

$$\forall x \in X \; \exists y \in Y \quad f(x) \le g(y)$$

следует из того, что по любому $x = (x_j)$ можно построить $y = (y_j)$ следующим образом

$$y_j = \begin{cases} d - x_2, & j = 1, \\ x_j, & j > 1. \end{cases}$$

И вновь, если всё аккуратно подставить, то получится

$$f(x) = g(y) \quad \to \quad f(x) \le g(y).$$

Таким образом мы доказали, что утверждение можно применить. Следствием этого является то, что задача $\mathcal{P}$ действительно сводится к задаче $\mathcal{Q}$.

Мы взяли задачу с n переменными и перешли от неё к задаче с $n-1$ переменными. Разве не круто?!

1.6 Решение задачи о салфетках

Решение (задачи о салфетках)

Решим задачу о салфетках с помощью сведения к другой задаче. Запишем математическую формулировку нашей исходной задачи

($\mathcal{P}$)

$$C = \sum_{i=1}^{7} (c_1 x_i + c_2 y_i) \to \min_{(x_i),(y_i)},$$

$$x_i + y_i = p_i,$$

$$x_i \ge 0, \quad y_i \ge 0,$$

$$\begin{cases} T - y_7 \ge p_1, \\ T - y_1 \ge p_2, \\ T - y_2 \ge p_3, \\ \dots \\ T - y_5 \ge p_6, \\ T - y_6 \ge p_7. \end{cases}$$

Новая задача

Новая задача $\mathcal{Q}$ будет эквивалентна исходной задаче $\mathcal{P}$, однако в ней все x_i будут выражены через y_i и p_i следующим образом

$$x_i = p_i - y_i.$$

Запишем оптимизируемую характеристику в новой задаче $\mathcal{Q}$

$$
\begin{aligned}
C &= \sum_{i=1}^{7}(c_1 x_i + c_2 y_i) \to \min_{(x_i),(y_i)} \\
&= \sum_{i=1}^{7} \big(c_1(p_i - y_i) + c_2 y_i\big) \to \min_{(y_i)} \\
&= \sum_{i=1}^{7} \big(y_i(c_2 - c_1) + c_1 p_i\big) \to \min_{(y_i)} \\
&= \underbrace{(c_2 - c_1)}_{\textsf{const}, <0} \sum_{i=1}^{7} y_i + \underbrace{c_1 \sum_{i=1}^{7} p_i}_{\textsf{const}} \to \min_{(y_i)} \\
&\sim \sum_{i=1}^{7} y_i \to \max_{(y_i)}.
\end{aligned}
$$

То есть фактически наша новая задача $\mathcal{Q}$ выглядит как

$$\sum_{i=1}^{7} y_i \to \max_{(y_i)}$$

с учётом ограничений

$$
\begin{cases}
T - y_7 \geq p_1, \\
T - y_1 \geq p_2, \\
T - y_2 \geq p_3, \\
\dots \\
T - y_5 \geq p_6, \\
T - y_6 \geq p_7.
\end{cases}
$$

Стоит заметить, что в задаче $\mathcal{P}$ все $x_i \geq 0$, а значит нужно ввести ограничения на y_i

$$y_i \leq p_i.$$

Итого, новая задача формулируется следующим образом

($\mathcal{Q}$)

$$\sum_{i=1}^{7} y_i \to \max_{(y_i)},$$

$$
\begin{cases}
T - y_7 \geq p_1, \\
T - y_1 \geq p_2, \\
T - y_2 \geq p_3, \\
\dots \\
T - y_5 \geq p_6, \\
T - y_6 \geq p_7;
\end{cases}
$$

$$y_i \leq p_i, \quad i = 1 \dots 7.$$

Решение новой задачи

Нам нужно максимизировать сумму y_i, при этом на каждый y_i есть два ограничения сверху. Так, например для y_2 есть ограничения

$$\begin{cases} T - y_2 \geq p_3, \\ y_2 \leq p_2; \end{cases}$$

$$\Updownarrow$$

$$\begin{cases} y_2 \leq T - p_3, \\ y_2 \leq p_2. \end{cases}$$

Ясно, что если все y_i выбрать максимально возможными, то и их сумма будет максимально возможной. Максимально возможное значение y_i, которое удовлетворяет условие — это минимум из двух верхних границ. Например, для y_2 это будет

$$\min\{T - p_3, p_2\}.$$

Таким образом, мы уже можем записать оптимальное решение y^* задачи $\mathcal{Q}$

$$y_1^* = \min\{T - p_2, p_1\},$$
$$y_2^* = \min\{T - p_3, p_2\},$$
$$\dots$$
$$y_6^* = \min\{T - p_7, p_6\},$$
$$y_7^* = \min\{T - p_1, p_7\}.$$

Возвращение к исходной задаче

После того, как мы нашли оптимальное решение новой задачи $\mathcal{Q}$, нужно вернуться к исходной задаче $\mathcal{P}$. Её оптимальное решение x^0 выражается следующим образом

$$x_i^0 = p_i - y_i^*.$$

Таким образом, мы свели исходную задачу $\mathcal{P}$ к новой задаче $\mathcal{Q}$ с меньшим числом переменных, решили новую задачу и по её оптимальному решению построили оптимальное решение исходной задачи.

Почему x^0 — оптимальное решение задачи $\mathcal{P}$? Для доказательства этого можно использовать сведение к другой задаче по аналогии с тем, как это было сделано в примере.

1.7 Использование булевых переменных

Очень часто в реальных задачах встречаются самые разные логические условия. Например: «если верно ..., то должно быть верно ...». Данные условия можно записать на языке формул математической логики, однако в рамках данного курса будет удобнее, если они будут записаны с использованием алгебраических выражений. Таким образом мы сможем записать любые ограничения любых задач на языке алгебры. Для записи логических условий хорошо подходят *булевы переменные* (переменные, которые могут принимать лишь значения $0, 1$).

Утверждение (простые условия)

Если x и y — булевы переменные, то имеют место следующие соответствия логических условий и алгебраической записи

Условие	Запись
если $x = 0$, то $y = 0$	$x \geq y$
если $x = 0$, то $y = 1$	$x \geq 1 - y$
если $x = 1$, то $y = 0$	$1 - x \geq y$
если $x = 1$,то $y = 1$	$1 - x \geq 1 - y$

Доказательство

Для доказательства рассмотрим лишь первые два логических условия, поскольку доказательства для остальных аналогичны.

1. Если $x = 0$, то $y = 0$
   - если $x = 0$, то y не может быть равен 1, потому что $0 \ngeq 1$, значит y может равняться лишь 0;
   - если $x = 1$, то y может равняться как 0, так и 1, поскольку $1 \geq 0$ и $1 \geq 1$.
2. Если $x = 0$, то $y = 1$
   - если $x = 0$, то $1 - y$ не может быть равен 1, а значит $1 - y = 0$, как следствие $y = 1$;
   - если $x = 1$, то $1 - y$ может равняться как 0, так и 1, значит y может принимать любое значение из $\{0, 1\}$.

Пример

Используя предыдущее утверждение, можно алгебраически записывать различные условия задач. Например, пусть в некоторой рассматриваемой задаче есть два логических условия A и B, а нам нужно записать на языке алгебры логическое выражение «если верно A, то верно B». Для этого введём две булевы переменные x и y

$$x = \begin{cases} 1, & A \text{ — истина}, \\ 0, & A \text{ — ложь}; \end{cases}$$

$$y = \begin{cases} 1, & B \text{ — истина}, \\ 0, & B \text{ — ложь}. \end{cases}$$

Тогда «если верно A, то верно B» эквивалентно утверждению «если $x = 1$, то $y = 1$», а по предыдущему утверждению алгебраически это можно записать как

$$\boxed{1 - x \geq 1 - y}.$$

Данное неравенство будет добавлено в ограничения рассматриваемой задачи, таким образом исходное условие про A и B будет учтено в математической модели.

Утверждение (сложные условия)

Пусть

- $x = (x_i)$ и $y = (y_k)$ — булевы векторы;
- $i \in I$, $k \in K$;

- $I^0 \subseteq I$ и $I^1 \subseteq I$ — непересекающиеся множества;
- $K^0 \subseteq K$ и $K^1 \subseteq K$ — непересекающиеся множества.

Векторы x и y будем называть «хорошими», если верно следующее

$$x_i = \begin{cases} 0, & i \in I^0, \\ 1, & i \in I^1; \end{cases} \qquad y_k = \begin{cases} 0, & k \in K^0. \\ 1, & k \in K^1. \end{cases}$$

Тогда имеют место следующие соответствия логических условий и алгебраической записи

Условие	Запись
x — «хороший» $\Rightarrow y = 0$	$\sum\limits_{i \in I^1}(1 - x_i) + \sum\limits_{i \in I^0} x_i \geq y$
$x = 0 \Rightarrow y$ — «хороший»	$\lVert K^0 \cup K^1 \rVert \cdot x \geq \sum\limits_{k \in K^1}(1 - y_k) + \sum\limits_{k \in K^0} y_k$
x — «хороший» $\Rightarrow y$ — «хороший»	$\lVert K^0 \cup K^1 \rVert \cdot \left(\sum\limits_{i \in I^1}(1 - x_i) + \sum\limits_{i \in I^0} x_i\right) \geq \sum\limits_{k \in K^1}(1 - y_k) + \sum\limits_{k \in K^0} y_k$

В первое условии y — вектор из одного элемента, то есть фактически булева переменная. Во втором условии x — вектор из одного элемента, то есть фактически булева переменная.

Доказательство

1. Если x — «хороший», то $y = 0$
   - если вектор x является «хорошим», то обе суммы равняются нулю, значит y ничего не остаётся кроме как быть равным нулю;
   - если вектор x не является «хорошим», то слева будет число ≥ 1, а значит y может быть как 0, так и 1.
2. Если $x = 0$, то y — «хороший»
   - если $x = 0$, то обе суммы должны равняться нулю, значит $y_k = 1 \Leftrightarrow k \in K^1$ и $y_k = 0 \Leftrightarrow k \in K^0$, значит y — «хороший»;
   - если $x \neq 0$, то $x = 1$, значит справа сумма сумм может быть какой угодно, поэтому y может иметь любой вид.

   Откуда взялся коэффициент $\lVert K^0 \cup K^1 \rVert$? Если $x \neq 0$, то y должен иметь возможность принимать любые значения (посылка ложна), однако алгебраически это не так. Теоретически правая часть может быть сколь угодно большой, а левая часть без коэффициента может быть лишь не больше 1. Это означает, что наше алгебраическое выражение по смыслу не совпадает с изначальным логическим условием. Чтобы оно совпадало, нужно разрешить y принимать любые значения при $x \neq 0$. Для этого как раз и добавлен коэффициент в левой части неравенства, чтобы неравенство оставалось верным при $x = 1$ и сколь угодно большой правой части.

3. Аналогично предыдущим пунктам.

Пример

Данное утверждение, в отличие от простых условий, позволяет алгебраически записывать логически условия, в которых истинными или ложными должны быть сразу несколько условий.

Например, пусть при решении некоторой задачи нам нужно записать логическое выражение «если условия A_1 и A_2 истины, то условие B_1 ложно, а B_2 и B_3 истинны». Определим два булевых вектора $x = (x_1, x_2)$ и $y = (y_1, y_2, y_3)$ следующим образом

$$x_i = \begin{cases} 1, & A_i \text{ — истина}, \\ 0, & A_i \text{ — ложь}; \end{cases} \qquad y_k = \begin{cases} 1, & B_k \text{ — истина}, \\ 0, & B_k \text{ — ложь}. \end{cases}$$

Исходное логическое выражение эквивалентно «если $x = (1, 1)$, то $y = (0, 1, 1)$». По предыдущему утверждению алгебраически это можно записать так (третий случай)

$$\boxed{3 \cdot (1 - x_1 + 1 - x_2) \geq (1 - y_2) + (1 - y_3) + y_1}. \tag{*}$$

Коэффициент 3 — это $\|K^0 \cup K^1\|$ из утверждения. Объясним его необходимость на данном примере.

Исходное логическое выражение звучит как «если $x = (1, 1)$, то $y = (0, 1, 1)$». Это означает, что если $x \neq (1, 1)$, то на y нет никаких ограничений, то есть он может принимать любые значения. Проверим, верно ли это, определим $\hat{x} = (0, 0)$ и подставим его в $(^*)$ без коэффициента

$$1 - 0 + 1 - 0 \geq (1 - y_2) + (1 - y_3) + y_1,$$

$$2 \geq (1 - y_2) + (1 - y_3) + y_1.$$

Неравенству выше не удовлетворяет вектор $\hat{y} = (1, 0, 0)$, поскольку $2 \ngeq 3$. То есть несмотря на то, что $\hat{x}$ не удовлетворяет условию, на y накладываются какие-то ограничения. Значит если из $(*)$ убрать коэффициент 3, то неравенство не будет эквивалентно логическому выражению «если $x = (1, 1)$, то $y = (0, 1, 1)$».

Утверждение (альтернативные условия)

Пусть в некоторой задаче сформулированы два ограничения

$$f_1(x) \geq b_1, \tag{1}$$

$$f_2(x) \geq b_2. \tag{2}$$

Введём булеву переменную y следующим образом

$$y = \begin{cases} 0, & \text{выполняется (1)}, \\ 1, & \text{выполняется (2)}; \end{cases}$$

и пусть W — «большая величина», то есть

$$W \gg b_1, \quad W \gg b_2.$$

Тогда логическое выражение «выполняется либо (1), либо (2)» записывается алгебраически следующим образом

$$f_1(x) \geq b_1 - W(1 - y),$$

$$f_2(x) \geq b_2 - Wy.$$

Доказательство

- Если $y = 1$, то
$$f_1(x) \geq b_1,$$
$$f_2(x) \geq b_2 - W \to -\infty.$$
Первое неравенство верно, как и второе. Однако если со вторым есть вопросы, а точно ли оно верно, то вот первое выполняется гарантировано.
- Если $y = 0$, то имеем
$$f_1(x) \geq b_1 - W \to -\infty,$$
$$f_2(x) \geq b_2.$$
Аналогично, оба неравенства выполнены, но важно, что второе неравенство выполняется гарантировано.

Значит при любых значениях y выполняется либо (1), либо (2).

Замечание

То же самое можно записать с использованием двух булевых переменных y_1 и y_2

$$y_1 = \begin{cases} 1, & \text{(1) выполняется,} \\ 0, & \text{(1) не выполняется;} \end{cases} \qquad y_2 = \begin{cases} 1, & \text{(2) выполняется,} \\ 0, & \text{(2) не выполняется;} \end{cases}$$

$$f_1(x) \geq b_1 - W(1 - y_1),$$
$$f_2(x) \geq b_2 - W(1 - y_2),$$
$$y_1 + y_2 \geq 1.$$

Замечание

Если ограничения записаны в другую сторону, то есть

$$f_1(x) \leq b_1,$$
$$f_2(x) \leq b_2,$$

то записать алгебраически их можно следующим образом

$$f_1(x) \leq b_1 + W(1 - y_1),$$
$$f_2(x) \leq b_2 + Wy_2,$$
$$y_1 + y_2 \geq 1.$$

Аналогично можно записать через y.

Замечание

W — это некоторая «большая величина», какое конкретно значение эта она принимает, зависит от конкретной задачи. Где-то можно положить $W = 10^6$, где-то $W = 50$, однако полностью избавиться от W и записать условие без него не получится.

Замечание

Смысл булевых переменных y, y_1 и y_2 следующий. Предположим, что мы реализуем алгоритм, который решает нашу задачу, при этом он находит решения, которые удовлетворяют всем условиям. Наш алгоритм такой, что он сам выберет «наилучшие» значения этих переменных и на их основании построит оптимальное решение.

Утверждение (замена нелинейностей)

Пусть при решение задачи в некотором выражении нам встретилась нелинейность $\hat{x} \cdot \hat{y}$ ($\hat{x}$ и $\hat{y}$ — булевы переменные), тогда после замены $\hat{z} = \hat{x} \cdot \hat{y}$ необходимо ввести ограничения

$$\begin{cases} 2\hat{z} \le \hat{x} + \hat{y}, \\ \hat{z} + 1 \ge \hat{x} + \hat{y}. \end{cases}$$

Работать с нелинейностями неудобно, поэтому всегда хочется избавить от нелинейностей, однако просто заменить в выражении $\hat{x} \cdot \hat{y}$ на $\hat{z}$ нельзя, поскольку нужно изменить ограничения задачи.

Доказательство

Поскольку $\hat{z} = \hat{x} \cdot \hat{y}$, верно следующее

$$\hat{z} = 1 \quad \Longleftrightarrow \quad \hat{x} = 1 \,\&\, \hat{y} = 1.$$

Данное логические условие можно расписать через две импликации

1. Если $\hat{z} = 1$, то $\hat{x} = 1$ и $\hat{y} = 1$. Это можно записать алгебраически, используя сложные условия

$$x = \begin{pmatrix} \hat{z} \end{pmatrix}, \qquad y = \begin{pmatrix} \hat{x} & \hat{y} \end{pmatrix},$$

$$I^0 = \varnothing, \quad I^1 = \{1\}, \qquad K^0 = \varnothing, \quad K^1 = \{1, 2\},$$

$$\left|K^0 \cup K^1\right| \cdot \left(\sum_{i \in I^1} (1 - x_i) + \sum_{i \in I^0} x_i\right) \ge \sum_{k \in K^1} (1 - y_k) + \sum_{k \in K^0} y_k,$$

$$\Updownarrow$$

$$2 \cdot (1 - \hat{z}) \ge (1 - \hat{x}) + (1 - \hat{y}),$$

$$2\hat{z} \le \hat{x} + \hat{y}.$$

2. Если $\hat{x} = 1$ и $\hat{y} = 1$, то $\hat{z} = 1$. Данное логическое условие можно записать алгебраически, используя сложные условия. Будем использовать самый общий случай

$$x = \begin{pmatrix} \hat{x} & \hat{y} \end{pmatrix}, \qquad y = \begin{pmatrix} \hat{z} \end{pmatrix},$$

$$I^0 = \varnothing, \quad I^1 = \{1, 2\}, \qquad K^0 = \varnothing, \quad K^1 = \{1\},$$

$$\left|K^0 \cup K^1\right| \cdot \left(\sum_{i \in I^1} (1 - x_i) + \sum_{i \in I^0} x_i\right) \ge \sum_{k \in K^1} (1 - y_k) + \sum_{k \in K^0} y_k,$$

$$\Updownarrow$$

$$(1 - \hat{x}) + (1 - \hat{y}) \ge 1 - \hat{z},$$

$$\hat{z} + 1 \ge \hat{x} + \hat{y}.$$

1.8 Задача о проектах

Задача (о проектах)

Пусть есть 5 проектов, в которые можно вложиться. Для вложения в каждый проект нужно внести определённую сумму денег. Все проекты после вложения в них принесут определённый доход через какое-то время. Есть определённые условия, на которых можно вкладываться в проекты. Как получить наибольшую прибыль, имея ограниченные ресурсы?

Математическая модель

1. **Что будем оптимизировать?** Ответ: нужно максимизировать получаемую прибыль.

2. **Что существенно влияет на оптимизируемую характеристику?**

Проект	A	B	C	D	E
Доход	3	2	1	4	2
Начальные вложения	1.5	0	0.5	4	1

Условия вложения в проекты

1. нужно вложиться хотя бы в один проект;
2. если вложились в A, то необходимо вложиться в D;
3. если вложились в B и C, то необходимо вложиться в A;
4. если вложились в B, то необходимо вложиться в C и E.

Параметры

- c_i — доход с проекта i;
- d_i — начальные вложения в проект i;
- Q — стартовый капитал;
- C — общий доход.

Переменные

- x_1, x_2, x_3, x_4, x_5 — булевы переменные, которые означают, будет ли вложение в соответствующий проект

$$x_i = \begin{cases} 1, & \text{будем вкладываться в } i\text{-ый проект,} \\ 0, & \text{иначе.} \end{cases}$$

3. **Математическая формулировка задачи**

Пусть D — сумма всех расходов, C — сумма всех доходов, F — прибыль, тогда

$$D = \sum_{i=1}^{5} x_i d_i, \quad C = \sum_{i=1}^{5} x_i c_i,$$

$$F = C - D = \sum_{i=1}^{5} x_i c_i - \sum_{i=1}^{5} x_i d_i = \sum_{i=1}^{5} x_i (c_i - d_i) \to \max_{(x_i)},$$

$$D \leq Q \iff \sum_{i=1}^{5} x_i d_i \leq Q.$$

Последнее ограничение означает, что мы не можем вложить в проекты больше стартового капитала.

Запись условий вложения в проекты

1. Первое условие можно записать алгебраически через
$$\sum_{i=1}^{5} x_i \geq 1.$$
2. Второе условие эквивалентно «если $x_1 = 1$, то $x_4 = 1$», алгебраически это записывается как (простые условия)
$$1 - x_1 \geq 1 - x_4 \iff x_1 \leq x_4.$$
3. Третье условие эквивалентно «если $x_2 = 1$ и $x_3 = 1$, то $x_1 = 1$», алгебраически это записывается так (сложные условия)
$$(1 - x_2) + (1 - x_3) \geq 1 - x_1 \iff 1 + x_1 \geq x_2 + x_3.$$
4. Четвёртое условие эквивалентно «если $x_2 = 1$, то $x_3 = 1$ и $x_5 = 1$», алгебраически это записывается так (сложные условия)
$$2 \cdot (1 - x_2) \geq 1 - x_3 + 1 - x_5 \iff 2x_2 \leq x_3 + x_5.$$

Итоговая модель

$$\sum_{i=1}^{5} x_i (c_i - d_i) \to \max_{(x_i)},$$

$$\sum_{i=1}^{5} x_i d_i \leq Q,$$

$$\begin{cases} \sum\limits_{i=1}^{5} x_i \geq 1, \\ x_1 \leq x_4, \\ 1 + x_1 \geq x_2 + x_3, \\ 2x_2 \leq x_3 + x_5; \end{cases}$$

$$x_i \in \{0, 1\}.$$

1.9 Задача о предприятии

Задача (о предприятии)

Пусть есть предприятие, которое производит определённые виды продукции, затрачивая некоторые свои ресурсы. Для простоты будем считать, что у нас есть лишь один вид ресурсов, который можно использовать для производства продукции. Как получить наибольший доход?

Математическая модель

1. Что будем оптимизировать? Ответ: нужно максимизировать получаемый доход.

2. Что существенно влияет на оптимизируемую характеристику?

Пусть $i = 1 \dots n$ — вид продукции, $I = \{1, 2, \dots, n\}$ — список всех производимых товаров.

Параметры

- c_i — доход продукции;
- b_i — расход ресурса на производство одной единицы продукции;
- B — запасы ресурса.

Переменные

- x_i — количество единиц производимой продукции.

3. Математическая формулировка задачи

$$C = \sum_{i \in I} c_i x_i \to \max_{(x_i)},$$

$$\sum_{i \in I} b_i x_i \leq B,$$

$$\forall i \in I \quad x_i \geq 0.$$

Изменения в задаче

Казалось бы, математическая модель составлена, однако тут приходит директор предприятия и говорит, что правительство определило два списка социально значимых товаров I_1, I_2, при этом нужно либо из первого списка производить не менее a_1 единиц продукции, либо из второго не менее a_2 единиц продукции. Оба условия можно записать следующим образом

$$\sum_{i \in i_1} x_i \geq a_1, \quad \sum_{i \in I_2} x_i \geq a_2,$$

Нам нужно, чтобы выполнялось хотя бы одно из них. Для этого введём булеву переменную

$$y = \begin{cases} 0, & \text{производим не менее } a_1 \text{ из } I_1, \\ 1, & \text{производим не менее } a_2 \text{ из } I_2; \end{cases}$$

и запишем с её помощью требуемое условие (альтернативные условия)

$$\sum_{i \in I_1} x_i \geq a_1 - Wy,$$

$$\sum_{i \in I_2} x_i \geq a_2 - W(1 - y).$$

Новые изменения в задаче

Казалось бы, сейчас математическая модель предприятия окончательно составлена, но... К вам вновь приходит директор предприятия и говорит, что правительство издало приказ, по которому если производится достаточно продукции из обоих списков, то предприятие получает надбавку к финансированию.

Пусть C_0 — надбавка за достаточное производство продукции из обоих списков социально значимых товаров. Введём две новые булевы переменные

$$y_1 = \begin{cases} 1, & \text{производим не менее } a_1 \text{ из } I_1, \\ 0, & \text{иначе;} \end{cases}$$

$$y_2 = \begin{cases} 1, & \text{производим не менее } a_2 \text{ из } I_2, \\ 0, & \text{иначе.} \end{cases}$$

Изменим выражения для оптимизируемой характеристики и логических условий

$$C = \sum_{i \in I} c_i x_i + C_0 y_1 y_2 \to \max_{(x_i), y_1, y_2},$$

$$\sum_{i \in I_1} x_i \geq a_1 - W(1 - y_1),$$

$$\sum_{i \in I_2} x_i \geq a_2 - W(1 - y_2).$$

Замена нелинейности

В общем и целом теперь нас всё устраивает, кроме нелинейности в виде $y_1 y_2$. Произведём замену нелинейности с помощью замены нелинейностей и запишем ограничения на новую булеву переменную

$$z = y_1 y_2,$$

$$C = \sum_{i \in I} c_i x_i + C_0 z \to \max_{(x_i), z},$$

$$2z \leq y_1 + y_2,$$

$$z + 1 \geq y_1 + y_2.$$

Итоговая модель

Параметры: $\{c_i\}$, $\{b_i\}$, B, a_1, a_2, C_0.
Переменные: $\{x_i\}$, y_1, y_2, z.

$$C = \sum_{i \in I} c_i x_i + C_0 z \to \max_{(x_i), z},$$

$$\sum_{i \in I} b_i x_i \leq B,$$

$$\sum_{i \in I_1} x_i \geq a_1 - W(1 - y_1),$$

$$\sum_{i \in I_2} x_i \geq a_2 - W(1 - y_2),$$

$$2z \leq y_1 + y_2, \quad z + 1 \geq y_1 + y_2,$$

$$y_1 \in \{0, 1\} \quad y_2 \in \{0, 1\}, \quad z \in \{0, 1\},$$

$$\forall i \in I \quad x_i \geq 0.$$

2 Динамическое программирование

Динамическое программирование — метод решения сложных задач путём разбиения их на более простые подзадачи. Этот метод эффективен, если процесс принятия решения состоит из многих шагов, то есть когда итоговое решение — последовательность принимаемых решений (*стратегия*). Теоретически с помощью динамического программирования можно решить любую экстремальную задачу, однако практически это не всегда возможно, так как появляется слишком много состояний.

2.1 Задача загрузки судна

Начнём рассмотрение динамического программирования на примере задачи.

Задача (загрузки судна)

Пусть есть грузовое судно и набор различных контейнеров. Требуется загрузить судно контейнерами таким образом, чтобы при их дальнейшей продаже заработать как можно больше, при этом размер грузовой площадки судна ограничен.

Математическая модель

1. Что будем оптимизировать? Ответ: доход с продажи контейнеров должен быть максимальным.

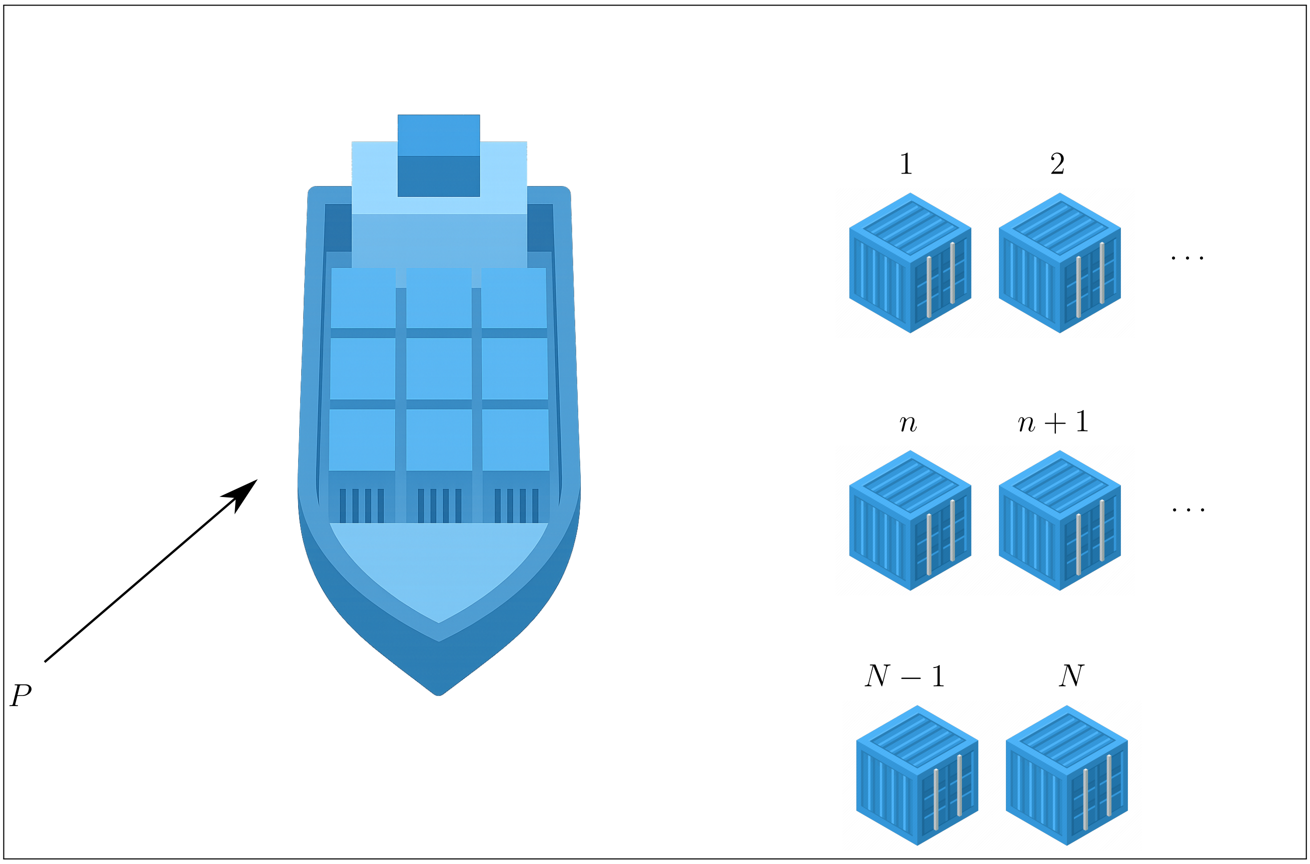
2. Что существенно влияет на оптимизируемую характеристику?
Пусть $i = 1 \dots N$ — номера контейнеров.

Параметры

- h_i — место на грузовой площадке, занимаемое контейнером i;
- c_i — ценность контейнера i;
- P — размер грузовой площадки судна.

Переменные

- x_i — сколько контейнеров с номером i нужно взять на судно.



3. Математическая формулировка задачи

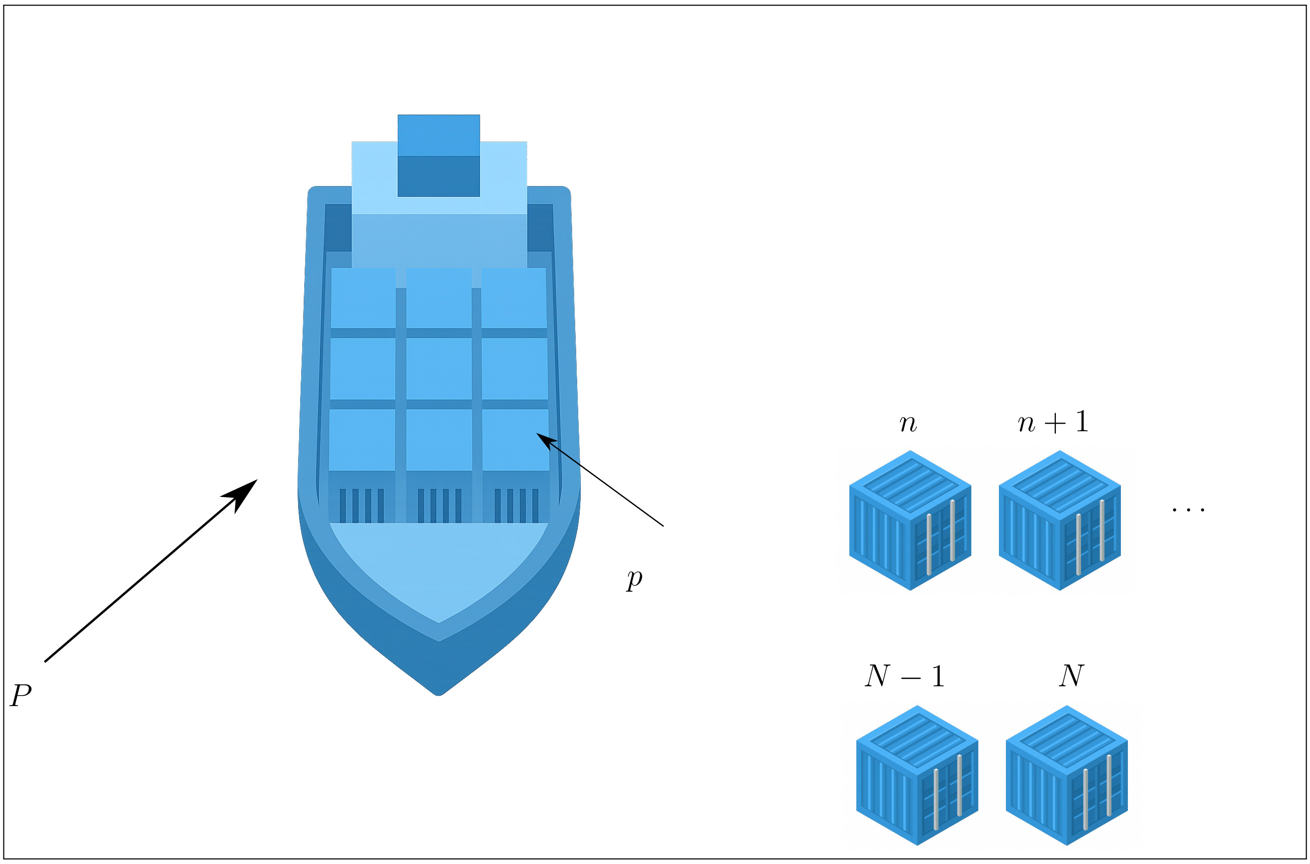
$$\sum_{i=1}^{N} c_i x_i \to \max_{(x_i),\, i \in \{1,\dots,N\}},$$

$$\sum_{i=1}^{N} h_i x_i \leq P,$$

$$x_i \in \{0, 1, 2, 3, \dots\}, \quad i \in 1, \dots N.$$

Решение

Для решения задачи будем рассматривать семейства аналогичных задач. Каждое семейство будет охарактеризовываться парой (n, p), где p — место на площадке, а n — минимальный номер для контейнеров, которые у нас есть. То есть вместо рассмотрения исходной задачи про все N контейнеров и всю грузовую площадку размером P будем рассматривать задачи, в которых нам доступны не все контейнеры, а лишь начинающиеся с номера $n \leq N$, а также в которых нам доступна не вся грузовая площадка судна, а лишь её часть размером $p \leq P$.



Запишем целевую функцию и ограничения для семейства (n, p)

$$\sum_{i=n}^{N} c_i x_i \to \max_{(x_i),\, i=n\ldots N},$$

$$\sum_{i=n}^{N} h_i x_i \leq p,$$

$$x_i \in \{0, 1, 2, 3, \ldots\} \quad i = n \ldots N.$$

Зададимся вопросом: а есть ли в этом семействе задачи, которые можно легко решить? Да, например если $n = N$, то есть если у нас в распоряжении есть лишь контейнер с номером N.

Введём обозначение. Пусть $f_n(p)$ — оптимальное значение целевой функции семейства задач (n, p). По своей сути $f_n(p)$ — это максимальный доход, который мы получим, если будем на площадку размера p грузить контейнеры с номерами $n, n+1, n+2, \ldots, N$. Легко заметить, что

$$f_N(p) = c_N \cdot \left[\frac{p}{h_N}\right].$$

Действительно, если у нас в распоряжении есть лишь контейнеры с номером N, то для получения максимального дохода нужно попытаться по-максимуму загрузить ими площадку на грузовом корабле, $\left[\frac{p}{h_N}\right]$ — количество контейнеров, которое поместится на площадку.

Таким образом, мы уже умеем решать задачу (N, p) для любого $p \leq P$. Теперь нужно совершить переход к решению исходной задачи

$$f_N(p) \longrightarrow f_1(P).$$

Для этого сформулируем принцип.

Определение (принцип оптимальности для оптимальной стратегии)

Оптимальная стратегия обладает тем свойством, что каким бы не было первое решение, последующие решения должны образовывать оптимальную стратегию относительно состояния, полученного по итогам первого решения.

Пример

Пусть мы стоим у доски и нам захотелось как можно быстрее выйти из аудитории через дверь. Каким бы ни был наш первый шаг, если мы хотим дойти до двери как можно быстрее, придётся всё время действовать оптимально. То есть даже если первый шаг был оптимальным, но потом мы пошли неоптимально, стратегия точно не получится оптимальной.

Возвращение к задаче

Предположим, что для некоторого $n \leq N$ и для $p \leq P$ мы уже знаем

$$f_{n+1}(p), f_{n+2}(p), \ldots, f_N(p),$$

однако нам бы хотелось посчитать $f_n(p)$, как это можно сделать?

Предположим, что $x_n = x = 1$, тогда груз с номером n даст нам ценность $c_n \cdot x = c_n$ и займёт на площадке место $h_n \cdot x = h_n$. Обобщим для $x_n \neq 1$ и запишем целевую функцию

$$\boxed{\forall p \leq P \quad f_n(p) = \max_{h_n \cdot x \leq p,\ x=0,1,2,\ldots} \{c_n \cdot x + f_{n+1}(p - h_n \cdot x)\}}.$$

Действительно, оптимальное значение при имеющихся контейнерах $n, n+1, n+2, \ldots, N$ складывается из некоторого количества контейнеров с номером n и контейнеров $n+1, n+2, \ldots, N$. Чтобы найти оптимальное значение нужно перебрать все варианты того, сколько взять контейнеров с номером n, при этом ясно, что взять их «слишком много» не получится, потому что размер площадки ограничен p. Это условие отражается через $h_n \cdot x \leq p$.

Мы получили **рекуррентное соотношение динамического программирования**, при этом, как уже говорилось ранее, $f_n(p)$ можно получить перебором значение $x_n = x$, а $f_{n+1}(\ldots)$ по предположению нам уже известно.

Таким образом, для решения исходной задачи нужно для всех $p \leq P$ и для всех $n = 2 \ldots N$ найти значение $f_n(p)$, а затем вычислить $f_1(P)$, это и будет значение целевой функции исходной задачи. Для получения стратегии нужно использовать значения $\{x_n\}_{i=1}^{N}$, на которых на каждом шаге достигается максимум $f_n(p)$.

Почему мы предполагаем, что $f_{n+1}(\ldots)$ нам известно? Ранее мы обсуждали, что мы можем вычислить $f_N(p)$ для любого $p \leq P$. Значит с помощью рекуррентного соотношения можем вычислить $f_{N-1}(p)$ для любого p, после этого можно вычислить $f_{N-2}(p)$ и так далее. То есть, если у нас есть хотя бы одно значение «в конце», то мы можем дойти до «начала» за определённое число шагов. Поэтому предположению о том, что на некотором шаге n нам уже известно оптимальное значение $f_{n+1}(p)$ имеет место.

2.2 N-шаговый процесс принятия решений

Обобщим подход к решению задачи о загрузке судна.

Алгоритм (N-шаговый процесс принятия решений)

Исходные данные

Для осуществления N-шагового процесса необходимо следующее

- *Количество шагов процесса*. Обозначение: $\boxed{N}$.
- *Состояние* — некоторое значение, которое мы имеем в начале каждого шага. Обозначение: $\boxed{p}$.
- *Решение* на шаге i. Обозначение: $\boxed{q_i}$ $(i = 1 \dots N)$.
- *Начальное состояние* — некоторая известная заданная величина. Обозначение: $\boxed{p_0}$.
- *Множество возможных состояний* на шаге i. Обозначение: $\boxed{P_i}$ $(i = 1 \dots N + 1)$.
- *Множество возможных решений* на шаге i. Обозначение $\boxed{Q_i}$ $(i = 1 \dots N)$.
- *Функция перехода* — функция перехода из текущего состояния p в новое состояние p', если на i-ом шаге принимается решение q

$$p' = \boxed{T_i(p, q)}.$$

- *Множество допустимых решений* на i-ом шаге в состоянии p

$$\boxed{Q_i(p) = \{q \in Q_i \mid T_i(p, q) \in P_{i+1}\}}.$$

- $\boxed{f_n(p)}$ — оптимальное значение целевой функции на шаге n, если в начале шага было состояние p.

В задаче о загрузке судна

- *количество шагов процесса*: количество видов контейнеров;
- *состояние*: свободное место на грузовой площадке;
- *решение* на шаге i: сколько грузить контейнеров с номером i;
- *начальное состояние*: P (свободное место на пустой грузовой площадке);
- *множество возможных состояний*: $\{0, 1, 2, \dots, P\}$ (сколько может быть доступного места на грузовой площадке на каждом шаге);
- *множество возможных решений*: $\{0, 1, 2, 3, \dots\}$ (это означает, что можно грузить лишь целое неотрицательное количество контейнеров);
- *функция перехода* на шаге n в состоянии p, если принимается решение q: $(p - h_n \cdot q)$ (это означает, сколько свободного места останется на площадке, если сейчас свободно p и грузим q контейнеров типа n);
- *множество допустимых решений* на шаге n в состоянии p:

$$Q_n(p) = \big\{q \in \{0, 1, 2, 3, \dots\} \bigm| h_n \cdot x \leq p\big\}.$$

Множества возможных состояний P_i определены для $i = 1 \dots N+1$, хотя шага с номером $N + 1$ нет. Это нужно для унификации определения функции перехода. В определении $Q_N(p)$ фигурирует P_{N+1}, поэтому если бы P_{N+1} было не определено, $Q_N(p)$ нужно было бы доопределять отдельно.

Принятие решения

- На самом первом шаге имеем начальное состояние p_0, то есть $i = 1$ и $p = p_0$.

- Пусть мы находимся на шаге i и имеем состояние p_i, нужно выбрать некоторое решение $q \in Q_i(p)$, сменить состояние на $p_{i+1} = T_i(p, q)$ и перейти к следующему шагу с номером $i + 1$.
- Если мы проделали N шагов, то завершаем процесс.

На каждом шаге мы выбирали некоторое решение $q \in Q_i(p)$, значит по итогам процесса у нас будет N значений $q_1, q_2, \ldots, q_N$, это и будет наша стратегия.

Однако нам бы не хотелось не просто допустимую, а оптимальную стратегию. Как понять, что стратегия оптимальна и как её получить?

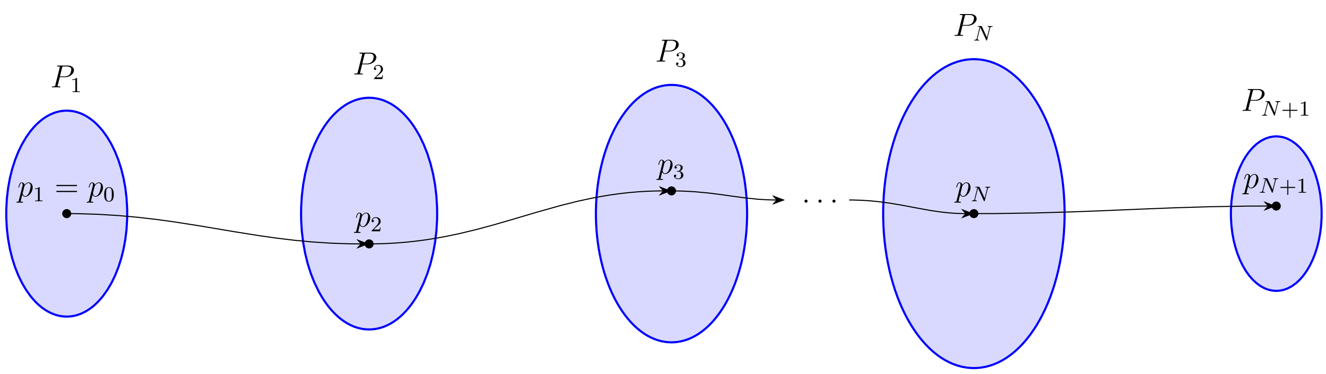


Рис. 2.1: Иллюстрация переходов между состояниями

2.3 Задача выбора оптимальной стратегии N-шагового процесса

Мы уже умеем использовать N-шаговый процесс принятия решений для задач, однако хотелось бы научиться с его помощью не просто решать задачи, а решать их оптимально. Для этого нужно ввести определение.

Определение

Функций дохода будем называть оптимальное значение целевой функции на i-ом шаге, если в состоянии p принимается решение q. Обозначение: $\boxed{g_i(p, q)}$.

Утверждение

Нахождение максимального значения целевой функции в исходной формулировке (без N-шагового процесса) эквивалентно нахождению

$$\max_{\{q_1, q_2, \ldots, q_N\}} \sum_{i=1}^{N} g_i(p_i, q_i).$$

при N-шаговом процессе

$$\begin{cases} p_1 = p_0, & \\ p_i \in P_i, & i \in \{2, \ldots, N+1\}, \\ q_i \in Q_i, & i \in \{1, \ldots, N\}, \\ p_{i+1} = T_i(p_i, q_i), & i \in \{1, \ldots, N\}. \end{cases}$$

Алгоритм (выбора оптимальной стратегии)

Для выбора оптимальной стратегии будем использовать функцию дохода $g_i(p,q)$. Для простоты будем всё рассматривать в рамках задачи загрузки судна. Запишем рекуррентные соотношения на $f_n(p)$ через функцию дохода

$$\forall p \in P_N \quad f_N(p) = \max_{q \in Q_N(p)} g_N(p,q),$$

$$\forall n < N \; \forall p \in P_n \quad f_n(p) = \max_{q \in Q_n(p)} \Big\{ g_n(p,q) + \underbrace{f_{n+1}\big(T_n(p,q)\big)}_{\text{уже знаем решение}} \Big\}.$$

Здесь мы фактически вновь рассматриваем семейства задач (n,p) и используем тот факт, что мы умеем решать задачу при $n = N$, а также то, что с помощью функции дохода можно осуществить переход от задачи, которую мы уже умеем решать (f_{n+1}), к задаче, которую пока не умеем (f_n). В этих обозначениях исходная задача — это $f_1(p_0)$.

Алгоритм выбора оптимальной стратегии состоит из двух этапов: *вычисление значений* $f_n(p)$, построение оптимальной стратегии.

Этап 1 На первом этапе нужно с помощью рекуррентных соотношений последовательно вычислить значения $f_N(p)$, $f_{N-1}(p)$, $f_{N-2}(p)$, $\ldots$, $f_2(p)$, $f_1(p)$ для всех возможных состояний на каждом шаге. При этом на самом деле вычислять значение $f_1(p)$ имеет смысл лишь для $p = p_0$. По итогам этого процесса мы вычислим $f_1(p_0)$ — оптимальное значение целевой функции исходной задачи, однако мы не узнаем, как выглядят оптимальные решения на каждом шаге. Чтобы это исправить помимо $f_n(p)$ каждом шаге будем считать и запоминать $q_n(p)$ — значение q, на котором достигается максимум в состоянии p на шаге n. Всё это удобно записывать в таблицу следующего вида

p	(f_1,q_1)	(f_2,q_2)	$\ldots$	(f_n,q_n)	(f_{n+1},q_{n+1})	$\ldots$	(f_{N-1},q_{N-1})	(f_N,q_N)
0								
1								
$\ldots$								
P								

В первом столбце таблицы представлены состояния, а в остальных — значения $f_n(p)$ и $q_n(p)$ на каждом шаге для каждого состояния p.

Этап 2 После первого этапа у нас есть заполненная таблица, в которой записаны посчитанные на каждом шаге n значения $f_n(p)$ и $q_n(p)$. С их помощью можно найти *оптимальную стратегию* следующим образом

$$\begin{aligned} p_1^* &= p_0, & q_1^* &= q_1(p_1^*), \\ p_2^* &= T_1(p_1^*, q_1^*), & q_2^* &= q_2(p_2^*), \\ p_3^* &= T_2(p_2^*, q_2^*), & q_3^* &= q_3(p_3^*), \\ &\ldots & &\ldots \\ p_N^* &= T_{N-1}(p_{N-1}^*, q_{N-1}^*), & q_N^* &= q_N(p_N^*). \end{aligned}$$

Утверждение

Стратегия, полученная алгоритмом выбора оптимальной стратегии, является оптимальной.

Доказательство

Итак, имеем стратегию $q = \{q_1^*, q_2^*, \ldots, q_N^*\}$. Докажем, что она оптимальна, то есть

$$f_1(p_0) = \sum_{i=1}^{N} g_i(p_i^*, q_i^*).$$

По построению $p_1^* = p_0$, значит

$$f_1(p_0) = f_1(p_1^*) = \max_{q \in Q_1(p_1^*)} \Big\{ g_1(p_1^*, q) + f_2\big(T_1(p_1^*, q)\big) \Big\}.$$

По построению максимум при $n = 1$ происходит при $q = q_1^* = q_1(p_1^*)$, поэтому

$$\begin{aligned} f_1(p_0) &= \max_{q \in Q_1(p_1^*)} \Big\{ g_1(p_1^*, q) + f_2\big(T_1(p_1^*, q)\big) \Big\} \\ &= g_1(p_1^*, q_1^*) + f_2\big(T_1(p_1^*, q_1^*)\big) \\ &\overset{def}{=} g_1(p_1^*, q_1^*) + f_2(p_2^*) \\ &= g_1(p_1^*, q_1^*) + \max_{q \in Q_2(p_2^*)} \Big\{ g_2(p_2^*, q) + f_3\big(T_2(p_2^*, q)\big) \Big\}. \end{aligned}$$

По построению максимум при $n = 2$ происходит при $q = q_2^* = q_2(p_2^*)$, поэтому

$$\begin{aligned} f_1(p_0) &= g_1(p_1^*, q_1^*) + \max_{q \in Q_2(p_2^*)} \Big\{ g_2(p_2^*, q) + f_3\big(T_2(p_2^*, q)\big) \Big\} \\ &= g_1(p_1^*, q_1^*) + g_2(p_2^*, q_2^*) + f_3\big(T_2(p_2^*, q_2^*)\big) \\ &\overset{def}{=} g_1(p_1^*, q_1^*) + g_2(p_2^*, q_2^*) + f_3(p_3^*) \\ &= g_1(p_1^*, q_1^*) + g_2(p_2^*, q_2^*) + \max_{q \in Q_3(p_3^*)} \Big\{ g_3(p_3^*, q) + f_4\big(T_3(p_3^*, q)\big) \Big\}. \end{aligned}$$

Рассуждая аналогично, можно получить, что

$$\begin{aligned} f_1(p_0) &= g_1(p_1^*, q_1^*) + g_2(p_2^*, q_2^*) + g_3(p_3^*, q_3^*) + \cdots + g_N(p_N^*, q_N^*) \\ &= \sum_{i=1}^{N} g_i(p_i^*, q_i^*). \end{aligned}$$

Требуемое доказано, значит стратегия является оптимальной.

Замечание

Если в задаче начальное состояние может принимать несколько значений, то нужно осуществить процесс для каждого начального состояния, а потом сравнить полученные стратегии и выбрать из них наилучшую.

Замечание

N-шаговый процесс удобно использовать, когда понятно, что будет состоянием, какие будут шаги и как будет осуществляться переход от одного состояния к другому. Для использования алгоритма выбора оптимальной стратегии важно, чтобы имела место *сепарабельность*, то есть тот факт, что функция дохода может быть посчитана на каждом шаге отдельно, то есть что эффект на целевую функцию на каждом шаге отделён от эффекта на неё на других шагах.

Замечание

Иногда при решении задачи в качестве состояния на каждом шаге нужно выбрать не само значение p_i, а пару (q_{i-1}, p_i), чтобы знать, какое было принято решение на предыдущем шаге. Это может быть полезно, если за «не сбалансированые» решения предусмотрено штрафы. Например, если стратегия «потратить в этом году все деньги, а в следующем не потратить ничего» нас не устраивает.

2.3.1 Решение задачи о машине

Решим задачу о машине с помощью N-шагового процесса принятия решения.

N-шаговый процесс

Изменим обозначения исходной задачи. Пусть $i = 1 \dots N$ — вид детали.

Параметры

- h_i — масса деталей;
- t_i — срок службы деталей;
- d_i — количество работающих деталей в машине;
- P — максимальная масса посылки.

Переменные

- x_i — сколько нужно взять деталей в посылку.

Исходные данные

- <u>количество шагов процесса</u> = количество видов деталей = N;
- <u>на i-ом шаге</u> будем определять, сколько деталей i-го типа нужно взять в посылку;
- <u>состояние</u> — свободная масса груза в посылке, то есть сколько ещё массы можно использовать.

Будем рассматривать семейства задач, которые определяются парой (n, p). Это будет означать, что у нас распоряжении

- есть детали не всех видов, а лишь от n до N, $n \leq N$;
- есть не вся масса посылки P, а лишь её часть $p \leq P$.

База процесса

Пусть $f_n(p)$ — максимальное время, которое проработает машина в семействе задач (n, p). Заметим, что если $n = N$, то задача легко решается

$$\forall p \leq P \quad f_N(p) = \left(d_N + \underbrace{\left[\frac{p}{h_N}\right]}_{q_N(p)}\right) \cdot t_N.$$

В таком случае решение состоит в том, что нужно заказать детали вида N на максимум, то есть сколько можем, столько и заказываем.

Переход

Теперь когда у нас есть база для решения задачи, осуществим переход $f_{n+1}(p) \to f_n(p)$ в предположении, что $f_{n+1}(p)$ мы знаем $\forall p \le P$.

$$\boxed{f_n(p) = \max_{\substack{x = x_n \\ h_n \cdot x \le p,\ x \ge 0}} \min\Big\{(d_n + x)\cdot t_n\ ;\ f_{n+1}(p - x \cdot h_n)\Big\}}.$$

$(d_n + x)\cdot t_n$ — это сколько проработают детали n-го вида с учётом уже имеющихся в машине и тех, которые будут заказаны. Для решения задачи нужно для всех $p \le P$ найти значение $f_n(p)$ и соответствующие значения x, на которых максимум и достигается.

В исходной формулировке у нас было

$$f_n(p) = \max_{q \in Q_n(p)} \Big[g_n(p,q) + f_{n+1}\big(T_n(p,q)\big)\Big].$$

Отличие в данной задаче состоит в том, что у нас в рекуррентном соотношении не суммирование, а взятие минимума. Теоретически здесь может быть и умножение, но большой роли при решении это не играет. Важно, что рекуррентное соотношение написано.

Решение

Решим задачу с конкретными числовыми данными

- $N = 4$;
- $h = \{3, 2, 3, 2\}$ кг;
- $t = \{6, 3, 2, 4\}$ дней;
- $d = \{1.5, 1.5, 1.5, 1.5\}$. d_i = 1.5 может означать, например, что в машине установлены две детали вида i, при этом одна отработала половину своего срока, а вторую только недавно установили;
- $P = 8$ кг.

В ходе решения задачи будем заполнять следующую таблицу

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)
0				
1				
2				
3				
4				
5				
6				
7				
8				

Таблицу будем заполнять справа налево, запоминая q_i — то значение x, на котором достигается максимум целевой функции.

В левом столбце у нас все возможные состояния p, а p — свободная масса груза в посылке. Заметим, что ни на каком шаге p не может равняться 7, то есть какие бы грузы мы не клали в посылку, свободная масса груза не сможет равняться 7. Это означает, что можно не подсчитывать значения для $p = 7$.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)
0				
1				
2				
3				
4				
5				
6				
7	$\times$	$\times$	$\times$	$\times$
8				

Заполнение таблицы

$\boxed{\text{Шаг 1}}$ На первом шаге $n = N = 4$, запишем выражение для $f_4(p)$

$$f_4(p) = \left(d_n + \left[\frac{p}{h_4}\right]\right) \cdot t_4 = \left(1.5 + \left[\frac{p}{2}\right]\right) \cdot 4.$$

Таким образом мы можем посчитать значение $f_4(p)$ для любого $p \leq P$, при этом нам известно значение, на котором достигается максимум (q_4)

$$q_4 = \left[\frac{p}{2}\right].$$

Посчитаем значение f_4 для всех возможных состояний

$$f_4(0) = 6, \quad q_4 = \left[\frac{0}{2}\right] = 0;$$

$$f_4(1) = 6, \quad q_4 = \left[\frac{1}{2}\right] = 0;$$

$$f_4(2) = 10, \quad q_4 = \left[\frac{2}{2}\right] = 1;$$

$$f_4(3) = 10 \quad q_4 = \left[\frac{3}{2}\right] = 1;$$

$$f_4(4) = 14, \quad q_4 = \left[\frac{4}{2}\right] = 2;$$

$$f_4(5) = 14, \quad q_4 = \left[\frac{5}{2}\right] = 2;$$

$$f_4(6) = 18, \quad q_4 = \left[\frac{6}{2}\right] = 3;$$

$$f_4(8) = 22, \quad q_4 = \left[\frac{8}{2}\right] = 4.$$

Занесём все эти данные в последний столбец таблицы.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)
0				$(6, 0)$
1				$(6, 0)$
2				$(10, 1)$
3				$(10, 1)$
4				$(14, 2)$
5				$(14, 2)$
6				$(18, 3)$
7	$\times$	$\times$	$\times$	$\times$
8				$(22, 4)$

Шаг 2 На втором шаге $n = 3$. Запишем выражение для $f_3(p)$

$$\begin{aligned} f_3(p) &= \max_{\substack{x = x_3 \\ h_3 \cdot x \leq p,\ x \geq 0}} \min \Big\{ (d_3 + x) \cdot t_3 \ ; \ f_4(p - x \cdot h_3) \Big\} \\ &= \max_{\substack{x = x_3 \\ 3x \leq p,\ x \geq 0}} \min \Big\{ 3 + 2x \ ; \ f_4(p - 3x) \Big\}. \end{aligned}$$

На данном шаге нам нужно для всех $0 \leq p \leq P$ вычислить значение $f_3(p)$ и запомнить значение x, на котором достигается максимум в каждом конкретном состоянии. Для каждого состояния мы будем перебирать различные значения x. Для примера найдём $f_3(8)$. Для этого нам нужно перебрать значения $x \in \{0, 1, 2\}$, поскольку при $x > 2$ неравенство $3x \leq 8$ уже не выполняется, а $x < 0$ нам не подходят по условию. Вычислим значение $\min$ для каждого из этих трёх значений x

$$\begin{aligned} x = 0: &\quad \min \Big\{ 3 + 2 \cdot 0 \ ; \ f_4(8) \Big\} = \min\{3, 22\} = 3; \\ x = 1: &\quad \min \Big\{ 3 + 2 \cdot 1 \ ; \ f_4(5) \Big\} = \min\{5, 14\} = 5; \\ x = 2: &\quad \min \Big\{ 3 + 2 \cdot 2 \ ; \ f_4(2) \Big\} = \min\{7, 10\} = 7. \end{aligned}$$

То есть мы рассмотрели три разных значения x (0, 1, 2), для каждого из них вычислили значение $\min \Big\{ (d_n + x) \cdot t_n \ ; \ f_{n+1}(p - x \cdot h_n) \Big\}$, а после этого выбрали из трёх итоговых значение максимальное — 7, при этом запомнили, что это значение достигается при $x = 2$. Однако данные вычисления можно записать существенно короче

$$f_3(8) = \begin{array}{c|l} 0 & \{3 \ ; \ f_4(8) = 22\} = 3 \\ 1 & \{5 \ ; \ f_4(8) = 14\} = 5 \\ 2 & \{7 \ ; \ f_4(8) = 10\} = 7 \end{array}$$

В первом столбце у нас идут перебираемые значения x, а далее для каждого из них вычисление $\min$, само слово «min» писать здесь излишне. В кружок обведено значение $\max$. Данной нотации и будем придерживаться всюду далее. Посчитаем $f_3(p)$ для всех p

$$f_3(0) = \begin{array}{c|l} 0 & \{3 \ ; \ f_4(0) = 6\} = 3 \end{array}$$

$$f_3(1) = \begin{array}{c|l} 0 & \{3 \ ; \ f_4(1) = 6\} = 3 \end{array}$$

$$f_3(2) = \begin{array}{c|l} 0 & \{3\ ;\ f_4(2) = 10\} = 3 \end{array}$$

$$f_3(3) = \begin{array}{c|l} 0 & \{3\ ;\ f_4(3) = 10\} = 3 \\ 1 & \{5\ ;\ f_4(0) = 6\} = 5 \end{array}$$

$$f_3(4) = \begin{array}{c|l} 0 & \{3\ ;\ f_4(4) = 14\} = 3 \\ 1 & \{5\ ;\ f_4(1) = 6\} = 5 \end{array}$$

$$f_3(5) = \begin{array}{c|l} 0 & \{3\ ;\ f_4(5) = 14\} = 3 \\ 1 & \{5\ ;\ f_4(2) = 10\} = 5 \end{array}$$

$$f_3(6) = \begin{array}{c|l} 0 & \{3\ ;\ f_4(6) = 18\} = 3 \\ 1 & \{5\ ;\ f_4(3) = 10\} = 5 \\ 2 & \{7\ ;\ f_4(0) = 6\} = 6 \end{array}$$

$$f_3(8) = \begin{array}{c|l} 0 & \{3\ ;\ f_4(8) = 22\} = 3 \\ 1 & \{5\ ;\ f_4(3) = 10\} = 5 \\ 2 & \{7\ ;\ f_4(2) = 10\} = 7 \end{array}$$

Занесём все данные в таблицу. В столбец (f_3, q_3) возле максимального значения, обведённого в кружочек, для каждого состояния p мы ещё записываем значение x, в котором достигается максимум. Так, для $f_3(8)$ это $x = 2$, для $f_3(5)$ это $x = 1$ и так далее.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)
0			$(3, 0)$	$(6, 0)$
1			$(3, 0)$	$(6, 0)$
2			$(3, 0)$	$(10, 1)$
3			$(5, 1)$	$(10, 1)$
4			$(5, 1)$	$(14, 2)$
5			$(5, 1)$	$(14, 2)$
6			$(6, 2)$	$(18, 3)$
7	$\times$	$\times$	$\times$	$\times$
8			$(7, 2)$	$(22, 4)$

$\boxed{\text{Шаг 3}}$ На третьем шаге $n = 2$. Запишем выражение для $f_2(p)$

$$\begin{aligned} f_2(p) &= \max_{\substack{x = x_2 \\ h_2 \cdot x \leq p,\ x \geq 0}} \min \Big\{ (d_2 + x) \cdot t_2\ ;\ f_3(p - x \cdot h_2) \Big\} \\ &= \max_{\substack{x = x_2 \\ 2x \leq p,\ x \geq 0}} \min \Big\{ 4.5 + 3x\ ;\ f_3(p - 2x) \Big\}. \end{aligned}$$

Посчитаем $f_2(p)$ для всех p

$$f_2(0) = \begin{array}{c|l} 0 & \{4.5\ ;\ f_3(0) = 3\} = 3 \end{array}$$

$$f_2(1) = \begin{array}{c|l} 0 & \{4.5\ ;\ f_3(1)=3\} = 3 \end{array}$$

$$f_2(2) = \begin{array}{c|l} 0 & \{4.5\ ;\ f_3(2)=3\} = 3 \\ 1 & \{7.5\ ;\ f_3(0)=3\} = 3 \end{array}$$

$$f_2(3) = \begin{array}{c|l} 0 & \{4.5\ ;\ f_3(3)=5\} = 4.5 \\ 1 & \{7.5\ ;\ f_3(1)=3\} = 3 \end{array}$$

$$f_2(4) = \begin{array}{c|l} 0 & \{4.5\ ;\ f_3(4)=5\} = 4.5 \\ 1 & \{7.5\ ;\ f_3(2)=3\} = 3 \\ 2 & \{10.5\ ;\ f_3(0)=3\} = 3 \end{array}$$

$$f_2(5) = \begin{array}{c|l} 0 & \{4.5\ ;\ f_3(5)=5\} = 4.5 \\ 1 & \{7.5\ ;\ f_3(3)=5\} = 5 \\ 2 & \{10.5\ ;\ f_3(1)=3\} = 3 \end{array}$$

$$f_2(6) = \begin{array}{c|l} 0 & \{4.5\ ;\ f_3(6)=6\} = 4.5 \\ 1 & \{7.5\ ;\ f_3(4)=5\} = 5 \\ 2 & \{10.5\ ;\ f_3(2)=3\} = 3 \\ 3 & \{13.5\ ;\ f_3(0)=3\} = 3 \end{array}$$

$$f_2(8) = \begin{array}{c|l} 0 & \{4.5\ ;\ f_3(8)=7\} = 4.5 \\ 1 & \{7.5\ ;\ f_3(6)=6\} = 6 \\ 2 & \{10.5\ ;\ f_3(4)=5\} = 5 \\ 3 & \{13.5\ ;\ f_3(2)=3\} = 3 \\ 4 & \{16.5\ ;\ f_3(0)=3\} = 3 \end{array}$$

Занесём все данные в таблицу. Заметим, что при вычислении $f_2(2) = 3$ максимум достигается и при $x = 0$, и при $x = 1$. В таблице это будет отражено как $(3, 0/1)$.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)
0		$(3,0)$	$(3,0)$	$(6,0)$
1		$(3,0)$	$(3,0)$	$(6,0)$
2		$(3,0/1)$	$(3,0)$	$(10,1)$
3		$(4.5,0)$	$(5,1)$	$(10,1)$
4		$(4.5,0)$	$(5,1)$	$(14,2)$
5		$(5,0)$	$(5,1)$	$(14,2)$
6		$(5,1)$	$(6,2)$	$(18,3)$
7	$\times$	$\times$	$\times$	$\times$
8		$(6,1)$	$(7,2)$	$(22,4)$

$\boxed{\text{Шаг 4}}$ На четвёртом шаге $n = 1$

Наша исходная задача — это $f_1(8)$, поэтому для всех остальных значений $p \neq 8$ искать $f_1(p)$ не нужно. Запишем выражение для $f_1(8)$

$$
\begin{aligned}
f_1(8) &= \max_{\substack{x=x_1 \\ h_1 \cdot x \le 8,\ x \ge 0}} \min\Big\{(d_1 + x)\cdot t_1\ ;\ f_2(8 - x \cdot h_1)\Big\} \\
&= \max_{\substack{x=x_1 \\ 3x \le 8,\ x \ge 0}} \min\Big\{9 + 6x\ ;\ f_2(8 - 3x)\Big\}.
\end{aligned}
$$

Посчитаем $f_1(8)$

$$
f_1(8) = \begin{array}{c|l}
0 & \{9\ ;\ f_2(8) = 6\} = 6 \\
1 & \{15\ ;\ f_2(5) = 5\} = 5 \\
2 & \{21\ ;\ f_2(2) = 3\} = 5
\end{array}
$$

Занесём данные в таблицу.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)
0	$\times$	$(3, 0)$	$(3, 0)$	$(6, 0)$
1	$\times$	$(3, 0)$	$(3, 0)$	$(6, 0)$
2	$\times$	$(3, 0/1)$	$(3, 0)$	$(10, 1)$
3	$\times$	$(4.5, 0)$	$(5, 1)$	$(10, 1)$
4	$\times$	$(4.5, 0)$	$(5, 1)$	$(14, 2)$
5	$\times$	$(5, 0)$	$(5, 1)$	$(14, 2)$
6	$\times$	$(5, 1)$	$(6, 2)$	$(18, 3)$
7	$\times$	$\times$	$\times$	$\times$
8	$(6, 0)$	$(6, 1)$	$(7, 2)$	$(22, 4)$

Вспомним, что $f_1(8)$ — максимальное время работы машины для исходной задачи. Поскольку мы получили, что $f_1(8) = 6$, то в исходной задаче после заказа груза машина проработает ещё 6 дней.

Поиск оптимальной стратегии

Заметим, что наше начальное состояние $p_0 = 8$, поскольку в самом начале нам доступна вся масса груза $P = 8$ кг.

$$
\begin{aligned}
p_1^* &= p_0 = 8, & q_1^* &= 0, \\
p_2^* &= p_1^* - h_1 \cdot q_1^* = 8, & q_2^* &= 1, \\
p_3^* &= p_2^* - h_2 \cdot q_2^* = 6, & q_3^* &= 2, \\
p_4^* &= p_3^* - h_3 \cdot q_3^* = 0, & q_4^* &= 0.
\end{aligned}
$$

Наша стратегия — это $q^* = \{0, 1, 2, 0\}$, то есть заказать одну деталь второго типа и две детали третьего типа. При данной стратегии машина проработает ещё $f_1(8) = 6$ дней. При любых других стратегиях время работы будет меньше.

Замечание

Чтобы заполнять меньше значений таблицы можно было в начале прибегнуть к оптимизации, посчитав множество возможных состояний для каждого шага. Тогда бы мы считали на шаге i значения $f_i(p)$ не для всех $p \le P$, а лишь для этих самых возможных состояний.

Например, можно заметить, что для подсчёта $f_1(8)$ нам нужно было знать лишь $f_2(8)$, $f_2(5)$ и $f_2(2)$. Значение $f_2(p)$ для остальных p нам в итоге вообще не пригодились.

2.3.2 Трудоёмкость алгоритма

Утверждение

Трудоёмкость алгоритма выбора оптимальной стратегии N-шагового процесса равняется

$$T = \sum_{i=1}^{N} \|P_i\| \cdot \|Q_i\|.$$

Доказательство

Алгоритм состоит из двух этапов: «обратный ход» (заполнение таблицы) и «прямой ход» (нахождение оптимальной стратегии по заполненной таблице). Осуществить «прямой ход» по заполненной таблице не составляет труда, поэтому будем рассматривать лишь «обратный ход».

При «обратном ходе» у нас есть N шагов, на каждом из которых мы вычисляем значение $f_i(p)$ для всех возможных состояний p на текущем шаге, количество таких состояний равняется $\|P_i\|$. Для вычисления $f_i(p)$ для каждого конкретного p необходимо осуществить перебор допустимых решений для нахождение максимума/минимума целевой функции, количество допустимых решений на текущем шаге i равняется $\|Q_i\|$.

2.4 Задача о фермере

Задача (о фермере)

У фермера имеется стадо коров численностью 50 особей. Увеличение численности стада за год задаётся функцией $\alpha(v)$

$$\alpha(v) = \begin{cases} v + 10, & v \leq 70, \\ v + 20, & v > 70. \end{cases}$$

Затраты на содержание одной коровы в течение года $d = 0.5$, а выручка от продажи $c = 3$. Фермер составляет план продажи коров на ближайшие 5 лет при условии что численность стада не может быть ниже 50, а продажи производятся в конце года.

Математическая модель

1. Что будем оптимизировать? Ответ: фермер должен получить с продажи коров как можно больше денег.

2. Что существенно влияет на оптимизируемую характеристику?
Пусть $i = 1 \ldots 5$ — год.

Параметры

- $d = 0.5$ — стоимость содержания одной коровы за год;
- $c = 3$ — выручка за продажу одной коровы.

Переменные

- $q = \{q_i\}_{i=1}^{5}$ — сколько за год нужно продать коров.

3. Математическая формулировка задачи

Пусть G — выручка за 5 лет, а $p = \{p_i\}_{i=1}^5$ — количество коров в начале года, тогда

$$G = \sum_{i=1}^{5} (\underbrace{cq_i}_{\text{доход}} - \underbrace{dp_i}_{\text{расход}}) \to \max_q,$$

$$\forall i \; p_i \geq 50,$$

$$p_i = \begin{cases} 50, & i = 1, \\ \alpha(p_{i-1}) - q_{i-1}, & i > 1. \end{cases} \qquad (*)$$

Решение

Для упрощения решения задачи сделаем предположение, что фермер может продавать лишь количество коров, кратное 10.

Решим задачу с помощью N-шагового процесса принятия решений с теми же данными, которые были определены в задаче.

Исходные данные

- количество шагов процесса = количество лет = 5;
- на i-ом шаге будем определять, сколько коров нужно продать в конце года (q_i);
- состояние — текущее количество коров (p);
- функция перехода — сколько коров будет к следующему году, если в этом году было p и мы продали q

$$T_i(p, q) = \alpha(p) - q;$$

- множества возможных состояний — сколько коров может быть в начале каждого года; с учётом (*)

$$P_1 = \{50\}, \quad P_2 = \{50, 60\}, \quad P_3 = \{50, 60, 70\},$$

$$P_4 = \{50, 60, 70, 80\}, \quad P_5 = \{50, 60, 70, 80, 90, 100\};$$

- множества возможных решений — сколько коров можно продавать каждый год; из предположения о кратности этого количества 10

$$Q_i = \{0, 10, 20, 30, \dots\} \quad i = 1 \dots 5;$$

- множества допустимых решений — сколько коров можно продавать каждый год с учётом имеющего их количества

$$Q_i(p) = \{q \in Q_i \mid \alpha(p) - q \geq 50\} \quad i = 1 \dots 5;$$

- функция дохода — сколько денег получит фермер по итогам i-го года, если из имеющихся p коров он продаст q

$$g_i(p, q) = -0.5p + 3q = 3q - 0.5p.$$

Для решения будем рассматривать семейства задач, которые определяются парой (n, p). Фактически это означает, что

- рассматриваем доход за года $n, n+1, \dots, 5$, $n \leq 5$;
- поголовье скота на начало n-го года составляет $p \geq 50$ коров.

База процесса

Пусть $f_n(p)$ — максимальная выручка в семействе задач (n, p). Заметим, что если $n = 5$, то задача легко решается: фермеру нужно продать как можно больше коров, но чтобы их осталось не меньше 50

$$\forall p \in P_5 \quad f_5(p) = \max_{q \in Q_5(p)} \{3q - 0.5p\}.$$

Поскольку нам выгодно продать как можно больше коров, то нужно взять максимально возможное q, но чтобы в конце года коров осталось не меньше 50. Для этого можно взять $q_5(p) = \alpha(p) - 50$, тогда выражение для $f_5(p)$ в явном виде будет выглядеть так

$$\forall p \in P_5 \quad f_5(p) = 3(\alpha(p) - 50) - 0.5p.$$

Переход

Теперь когда у нас есть база для решения задачи, осуществим переход $f_{n+1}(p) \to f_n(p)$ в предположении, что $f_{n+1}(p)$ мы знаем $\forall p \geq 50$.

$$\boxed{f_n(p) = \max_{q \in Q_n(p)} \Big\{3q - 0.5p + f_{n+1}(\alpha(p) - q)\Big\}}. \qquad (**)$$

Вспомним множества допустимых состояний P_i. В объединении всех P_i лежит множество $\{50, 60, 70, 80, 90, 100\}$ — все значения, которые может принимать p. Также понятно, что считать $f_n(p)$ для $p \notin P_n$ не имеет смысла, так как эти состояния недопустимы. Исходя из этого составим следующую таблицу

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
50					
60	$\times$				
70	$\times$	$\times$			
80	$\times$	$\times$	$\times$		
90	$\times$	$\times$	$\times$	$\times$	
100	$\times$	$\times$	$\times$	$\times$	

Значение максимальной выручки в исходной задаче равняется $\mathbf{f_1(50)}$. Заполним таблицу, чтобы найти его, а затем вычислить все q_i^*.

Заполнение таблицы

Шаг 1 На первом шаге $n = 5$, запишем выражение для $f_5(p)$

$$f_5(p) = 3(\alpha(p) - 50) - 0.5p.$$

Таким образом мы можем посчитать значение $f_5(p)$ для любого $p \geq 50$, при этом нам известно значение, на котором достигается максимум (q_5)

$$q_5 = \alpha(p) - 50.$$

Посчитаем значение f_5 для всех возможных состояний

$$f_5(50) = 3(\alpha(50) - 50) - 50 \cdot 0.5 = 5, \quad q_5 = \alpha(50) - 50 = 10;$$

$$f_5(60) = 3(\alpha(60) - 50) - 60 \cdot 0.5 = 30, \quad q_5 = \alpha(60) - 50 = 20;$$

$$f_5(70) = 3(\alpha(70) - 50) - 70 \cdot 0.5 = 55, \quad q_5 = \alpha(70) - 50 = 30;$$

$$f_5(80) = 3(\alpha(80) - 50) - 80 \cdot 0.5 = 110, \quad q_5 = \alpha(80) - 50 = 50;$$

$$f_5(90) = 3(\alpha(90) - 50) - 90 \cdot 0.5 = 135, \quad q_5 = \alpha(90) - 50 = 60;$$

$$f_5(100) = 3(\alpha(100) - 50) - 100 \cdot 0.5 = 160, \quad q_5 = \alpha(100) - 50 = 70.$$

Занесём все эти данные в последний столбец таблицы.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
50					$(5, 10)$
60	$\times$				$(30, 20)$
70	$\times$	$\times$			$(55, 30)$
80	$\times$	$\times$	$\times$		$(110, 50)$
90	$\times$	$\times$	$\times$	$\times$	$(135, 60)$
100	$\times$	$\times$	$\times$	$\times$	$(160, 70)$

$\boxed{\text{Шаг 2}}$ На втором шаге $n = 4$. Выражение для этого и последующих шагов выглядит как рекуррентное соотношение (**). Делаем вычисления аналогично шагу 2 решения задачи о машине, с учетом наших ограничений на q_i

$$f_4(50) = \begin{array}{r|l} 0 & -25 + 3 \cdot 0 + f_5(60) = 30 - 25 = 5 \\ 10 & -25 + 3 \cdot 10 + f_5(50) = 30 + 5 - 25 = 10 \end{array}$$

$$f_4(60) = \begin{array}{r|l} 0 & -30 + 3 \cdot 0 + f_5(70) = 55 - 30 = 25 \\ 10 & -30 + 3 \cdot 10 + f_5(60) = 30 + 30 - 30 = 30 \\ 20 & -30 + 3 \cdot 20 + f_5(50) = 60 + 5 - 30 = 35 \end{array}$$

$$f_4(70) = \begin{array}{r|l} 0 & -35 + 3 \cdot 0 + f_5(80) = 110 - 35 = 75 \\ 10 & -35 + 3 \cdot 10 + f_5(70) = 30 + 55 - 35 = 50 \\ 20 & -35 + 3 \cdot 20 + f_5(60) = 60 + 30 - 35 = 55 \\ 30 & -35 + 3 \cdot 30 + f_5(50) = 90 + 5 - 35 = 60 \end{array}$$

$$f_4(80) = \begin{array}{r|l} 0 & -40 + 3 \cdot 0 + f_5(100) = 160 - 40 = 120 \\ 10 & -40 + 3 \cdot 10 + f_5(90) = 30 + 135 - 40 = 125 \\ 20 & -40 + 3 \cdot 20 + f_5(80) = 60 + 110 - 40 = 130 \\ 30 & -40 + 3 \cdot 30 + f_5(70) = 90 + 55 - 40 = 105 \\ 40 & -40 + 3 \cdot 40 + f_5(60) = 120 + 30 - 40 = 110 \\ 50 & -40 + 3 \cdot 50 + f_5(50) = 150 + 5 - 40 = 115 \end{array}$$

Занесём все данные в таблицу.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
50				$(10, 10)$	$(5, 10)$
60	$\times$			$(35, 20)$	$(30, 20)$
70	$\times$	$\times$		$(75, 0)$	$(55, 30)$
80	$\times$	$\times$	$\times$	$(130, 20)$	$(110, 50)$
90	$\times$	$\times$	$\times$	$\times$	$(135, 60)$
100	$\times$	$\times$	$\times$	$\times$	$(160, 70)$

Шаг 3 На третьем шаге $n = 3$. Считаем значения для $f_3(p)$

$$f_3(50) = \begin{array}{r|l} 0 & -25 + 3 \cdot 0 + f_4(60) = 35 - 25 = 10 \\ 10 & -25 + 3 \cdot 10 + f_4(50) = 30 + 10 - 25 = 15 \end{array}$$

$$f_3(60) = \begin{array}{r|l} 0 & -30 + 3 \cdot 0 + f_4(70) = 75 - 30 = 45 \\ 10 & -30 + 3 \cdot 10 + f_4(60) = 35 + 30 - 30 = 35 \\ 20 & -30 + 3 \cdot 20 + f_4(50) = 60 + 10 - 30 = 40 \end{array}$$

$$f_3(70) = \begin{array}{r|l} 0 & -35 + 3 \cdot 0 + f_4(80) = 130 - 35 = 95 \\ 10 & -35 + 3 \cdot 10 + f_4(70) = 30 + 75 - 35 = 90 \\ 20 & -35 + 3 \cdot 20 + f_4(60) = 60 + 35 - 35 = 60 \\ 30 & -35 + 3 \cdot 30 + f_4(50) = 90 + 10 - 35 = 65 \end{array}$$

Занесём все данные в таблицу.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
50			$(15, 10)$	$(10, 10)$	$(5, 10)$
60	$\times$		$(45, 0)$	$(35, 20)$	$(30, 20)$
70	$\times$	$\times$	$(95, 0)$	$(75, 0)$	$(55, 30)$
80	$\times$	$\times$	$\times$	$(130, 20)$	$(110, 50)$
90	$\times$	$\times$	$\times$	$\times$	$(135, 60)$
100	$\times$	$\times$	$\times$	$\times$	$(160, 70)$

Шаг 4 На четвертом шаге $n = 2$. Считаем значения для $f_2(p)$

$$f_2(50) = \begin{array}{r|l} 0 & -25 + 3 \cdot 0 + f_3(60) = 45 - 25 = 20 \\ 10 & -25 + 3 \cdot 10 + f_3(50) = 30 + 15 - 25 = 20 \end{array}$$

$$f_2(60) = \begin{array}{r|l} 0 & -30 + 3 \cdot 0 + f_3(70) = 95 - 30 = 65 \\ 10 & -30 + 3 \cdot 10 + f_3(60) = 45 + 30 - 30 = 45 \\ 20 & -30 + 3 \cdot 20 + f_3(50) = 60 + 15 - 30 = 45 \end{array}$$

Занесём все данные в таблицу. Заметим, что при вычислении $f_2(50) = 20$ максимум достигается и при $q = 0$, и при $q = 10$. В таблице это будет отражено как $(20, 0/10)$.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
50		$(20, 0/10)$	$(15, 10)$	$(10, 10)$	$(5, 10)$
60	$\times$	$(65, 0)$	$(45, 0)$	$(35, 20)$	$(30, 20)$
70	$\times$	$\times$	$(95, 0)$	$(75, 0)$	$(55, 30)$
80	$\times$	$\times$	$\times$	$(130, 20)$	$(110, 50)$
90	$\times$	$\times$	$\times$	$\times$	$(135, 60)$
100	$\times$	$\times$	$\times$	$\times$	$(160, 70)$

$\boxed{\text{Шаг 5}}$ На пятом шагу $n = 1$. Считаем значения для $f_1(p)$

$$f_1(50) = \begin{array}{r|l} 0 & -25 + 3 \cdot 0 + f_2(60) = 65 - 25 = 40 \\ 10 & -25 + 3 \cdot 10 + f_2(50) = 30 + 20 - 25 = 25 \end{array}$$

Занесём все данные в таблицу.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
50	$(40, 0)$	$(20, 0/10)$	$(15, 10)$	$(10, 10)$	$(5, 10)$
60	$\times$	$(65, 0)$	$(45, 0)$	$(35, 20)$	$(30, 20)$
70	$\times$	$\times$	$(95, 0)$	$(75, 0)$	$(55, 30)$
80	$\times$	$\times$	$\times$	$(130, 20)$	$(110, 50)$
90	$\times$	$\times$	$\times$	$\times$	$(135, 60)$
100	$\times$	$\times$	$\times$	$\times$	$(160, 70)$

Поиск оптимальной стратегии
Заметим, что наше начальное состояние $p_0 = 50$, поскольку в самом начале у фермера было 50 коров

$$\begin{aligned} p_1^* &= p_0 = 50, & q_1^* &= 0, \\ p_2^* &= \alpha(p_1^*) - q_1^* = 60, & q_2^* &= 0, \\ p_3^* &= \alpha(p_2^*) - q_2^* = 70, & q_3^* &= 0, \\ p_4^* &= \alpha(p_3^*) - q_3^* = 80, & q_4^* &= 20, \\ p_5^* &= \alpha(p_4^*) - q_4^* = 80, & q_5^* &= 50. \end{aligned}$$

Итоговая стратегия — это $q^* = \{0, 0, 0, 20, 50\}$, то есть в первые три года не продавать коров, в четвёртый год продать 20, а в пятый — 50. При данной стратегии прибыль составит $f_1(50) = 40$.

2.5 Задача о подрядчике

Задача (о подрядчике)

Строительный подрядчик оценивает потребность в количестве рабочей силы на каждую из последующих пяти недель как 5, 7, 8, 4 и 6 рабочих. Подрядчик может нанимать и увольнять рабочих в начале недели, исходя из того, что содержание избыточной рабочей силы обходится в 3 у.е. на одного рабочего в неделю, выплата выходного пособия для увольнения равна 2 у.е., а наем рабочей силы обходится в 4 у.е (за вход на рабочую биржу) и по 2 у.е. за каждого нанятого работника. Чему равны наименьшие затраты на найм, содержание избыточных и увольнение рабочих, если на момент начала первой недели у подрядчика не было ни одного рабочего?

Математическая модель

1. Что будем оптимизировать? Ответ: Подрядчик должен потратить как можно меньше денег, при этом количества рабочих должно хватать для выполнения плана для каждой недели.

2. Что существенно влияет на оптимизируемую характеристику?

Пусть $i = 1 \dots 5$ — неделя. Отметим как λ_i количество рабочих, необходимое для работы на i-ой неделе.

Неделя (i)	1	2	3	4	5
Количество рабочих (λ_i)	5	7	8	4	6

Параметры

- $e = 4$ — стоимость входа на биржу;
- $c = 2$ — стоимость найма рабочего;
- $u = 2$ — выплата выходного пособия для увольнения;
- $d = 3$ — выплата за одного избыточного рабочего.

Переменные

- $q = \{q_i\}_{i=1}^5$ — сколько нужно нанять рабочих в начале i-ой недели. Может быть отрицательным (увольняем) или равно 0 (ничего не делаем) $q_i \in Q_i = \{\underbrace{\dots, -2, -1}_{\text{увольняем}}, 0, \underbrace{1, 2, \dots}_{\text{нанимаем}}\}$

3. Математическая формулировка задачи

Пусть G — затраты за 5 недель, а $p = \{p_i\}_{i=1}^5$ — количество рабочих в начале недели, тогда

$$G = \sum_{i=1}^{5} \begin{cases} e + cq_i + d(p_i + q_i - \lambda_i), & q_i > 0, \\ d(p_i + q_i - \lambda_i), & q_i = 0, \\ -uq_i + d(p_i + q_i - \lambda_i), & q_i < 0; \end{cases}$$

$$G \to \min_q,$$

$$\forall i \; p_i + q_i \geq \lambda_i. \qquad (*)$$

Решение

Решим задачу с помощью N-шагового процесса принятия решений с теми же данными, которые были определены в задаче и с учетом условий на q.

Исходные данные

- количество шагов процесса = количество недель $= 5$;
- на i-ом шаге будем определять сколько нужно нанять/уволить рабочих в начале недели (q_i);
- состояние — текущее количество рабочих;
- функция перехода — сколько рабочих будет в начале следующей недели, если в начале этой было p и мы наняли/уволили q

$$T_i(p, q) = p + q;$$

- множества возможных состояний; учтём, что на i-ой неделе подрядчику нужно минимум λ_i рабочих, а также то, что нет смысла нанимать свыше $\max(\lambda)$ рабочих, при этом в начале было 0 рабочих

$$P_1 = \{0\}, \quad P_2 = \{5, 6, 7, 8\}, \quad P_3 = \{7, 8\},$$

$$P_4 = \{8\}, \quad P_5 = \{4, 5, 6\}, \quad P_6 = \{6\};$$

  Множество P_6 фиктивное, оно необходимо, чтобы указать что на 5-ой неделе должно после выполнения шага остаться 6 рабочих.
- множества возможных решений

$$Q_i = \{\underbrace{\ldots, -2, -1,}_{\text{увольняем}} 0, \underbrace{1, 2, \ldots}_{\text{нанимаем}}\};$$

- множества допустимых решений; учтём (*)

$$Q_i(p) = \{q \in Q_i \mid \underbrace{T_n(p, q)}_{p+q} \in P_{i+1}\};$$

- функция расхода — сколько денег потратит подрядчик в начале i-ой недели, если при имеющихся p рабочих он наймет q

$$g_i(p, q) = \begin{cases} 4 + 2q + 3(p + q - \lambda_i), & q > 0, \\ 3(p + q - \lambda_i), & q = 0, \\ -2q + 3(p + q - \lambda), & q < 0. \end{cases}$$

Для решения будем рассматривать семейства задач, которые определяются парой (n, p). Фактически это означает, что

- рассматриваем расход за недели $n, n + 1, \ldots, 5$, $n \leq 5$;
- количество рабочих на начало n-ой недели составляет $p \geq \begin{cases} \lambda_{n-1}, & n > 1, \\ 0 & n = 1. \end{cases}$

База процесса

Пусть $f_n(p)$ — минимальные затраты в семействе задач (n, p). Заметим, что если $n = 5$, то задача легко решается: подрядчику нужно нанять/уволить столько рабочих, чтобы их стало ровно $\lambda_5 = 6$ т.е. $q_5(p) = 6 - p$

$$\forall p \in P_5 \quad f_5(p) = g_5(p, 6 - p).$$

Переход

Теперь когда у нас есть база для решения задачи, осуществим переход $f_{n+1}(p) \to f_n(p)$ в предположении, что $f_{n+1}(p)$ мы знаем $\forall p \geq \lambda_n$.

$$\boxed{f_n(p) = \min_{q \in Q_n(p)} \Big\{ g_n(p,q) + f_{n+1}(\underbrace{T_n(p,q)}_{p+q}) \Big\}}. \qquad (**)$$

Вспомним множества допустимых состояний P_i. В объединении всех P_i лежит множество $\{0,4,5,6,7,8\}$ — все значения, которые может принимать p. Также понятно, что считать $f_n(p)$ для $p \notin P_n$ не имеет смысла, так как эти состояния недопустимы. Исходя из этого составим следующую таблицу

p	(f_1,q_1)	(f_2,q_2)	(f_3,q_3)	(f_4,q_4)	(f_5,q_5)
0		$\times$	$\times$	$\times$	$\times$
4	$\times$	$\times$	$\times$	$\times$	
5	$\times$		$\times$	$\times$	
6	$\times$		$\times$	$\times$	
7	$\times$			$\times$	$\times$
8	$\times$				$\times$

Значение минимального расхода в исходной задаче равняется $\mathbf{f_1(0)}$. Заполним таблицу, чтобы найти его, а затем вычислить все q_i^*.

Заполнение таблицы

$\boxed{\text{Шаг 1}}$ На первом шаге $n = 5$, запишем выражение для $f_5(p)$ с учетом того что $q_5(p) = 6 - p$

$$f_5(p) = g_5(p, 6-p).$$

Посчитаем значение f_5 для всех возможных состояний

$$f_5(4) = g_5(4,2) = 4 + 2 \cdot 2 + 3(4 + 2 - 6) = 8;$$

$$f_5(5) = g_5(5,1) = 4 + 2 \cdot 1 + 3(5 + 1 - 6) = 6;$$

$$f_5(6) = g_5(6,0) = 3(6 + 0 - 6) = 0.$$

Занесём все эти данные в последний столбец таблицы.

p	(f_1,q_1)	(f_2,q_2)	(f_3,q_3)	(f_4,q_4)	(f_5,q_5)
0		$\times$	$\times$	$\times$	$\times$
4	$\times$	$\times$	$\times$	$\times$	$(8,2)$
5	$\times$		$\times$	$\times$	$(6,1)$
6	$\times$		$\times$	$\times$	$(0,0)$
7	$\times$			$\times$	$\times$
8	$\times$				$\times$

$\boxed{\text{Шаг 2}}$ На втором шаге $n = 4$. Выражение для этого и последующих шагов выглядит как рекуррентное соотношение (**).

$$f_4(8) = \begin{array}{r|l} -4 & g_4(8,-4) + f_5(4) = 2\cdot 4 + 3(8-4-4) + 8 = 16 \\ -3 & g_4(8,-3) + f_5(5) = 2\cdot 3 + 3(8-3-4) + 6 = 15 \\ -2 & g_4(8,-2) + f_5(6) = 2\cdot 2 + 3(8-2-4) + 0 = 10 \end{array}$$

Занесём все данные в таблицу.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
0		$\times$	$\times$	$\times$	$\times$
4	$\times$	$\times$	$\times$	$\times$	$(8, 2)$
5	$\times$		$\times$	$\times$	$(6, 1)$
6	$\times$		$\times$	$\times$	$(0, 0)$
7	$\times$			$\times$	$\times$
8	$\times$			$(10, -2)$	$\times$

$\boxed{\text{Шаг 3}}$ На третьем шаге $n = 3$. Считаем значения для $f_3(p)$

$$f_3(8) = \begin{array}{r|l} 0 & g_3(8,0) + f_4(8) = 0 + 10 = 10 \end{array}$$

$$f_3(7) = \begin{array}{r|l} 1 & g_3(7,1) + f_4(8) = 4 + 2 + 10 = 16 \end{array}$$

Занесём все данные в таблицу.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
0		$\times$	$\times$	$\times$	$\times$
4	$\times$	$\times$	$\times$	$\times$	$(8, 2)$
5	$\times$		$\times$	$\times$	$(6, 1)$
6	$\times$		$\times$	$\times$	$(0, 0)$
7	$\times$		$(16, 1)$	$\times$	$\times$
8	$\times$		$(10, 0)$	$(10, -2)$	$\times$

$\boxed{\text{Шаг 4}}$ На четвертом шаге $n = 2$. Считаем значения для $f_2(p)$

$$f_2(5) = \begin{array}{r|l} 2 & g_2(5,2) + f_3(7) = 4 + 2\cdot 2 + 3(5+2-7) + 16 = 24 \\ 3 & g_2(5,3) + f_5(8) = 4 + 2\cdot 2 + 3(5+3-7) + 10 = 23 \end{array}$$

$$f_2(6) = \begin{array}{r|l} 1 & g_2(6,1) + f_3(7) = 4 + 2\cdot 1 + 3(6+1-7) + 16 = 22 \\ 2 & g_2(6,2) + f_5(8) = 4 + 2\cdot 2 + 3(6+2-7) + 10 = 21 \end{array}$$

$$f_2(7) = \begin{array}{r|l} 0 & g_2(7,0) + f_3(7) = 3(7+0-7) + 16 = 16 \\ 1 & g_2(7,1) + f_5(8) = 4 + 2\cdot 1 + 3(7+1-7) + 10 = 19 \end{array}$$

$$f_2(8) = \begin{array}{r|l} 0 & g_2(8,0) + f_3(8) = 3(8+0-7) + 10 = 13 \\ -1 & g_2(8,-1) + f_5(7) = 2\cdot 1 + 3(8-1-7) + 16 = 18 \end{array}$$

Занесём все данные в таблицу.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
0		$\times$	$\times$	$\times$	$\times$
4	$\times$	$\times$	$\times$	$\times$	$(8, 2)$
5	$\times$	$(23, 3)$	$\times$	$\times$	$(6, 1)$
6	$\times$	$(21, 2)$	$\times$	$\times$	$(0, 0)$
7	$\times$	$(16, 0)$	$(16, 1)$	$\times$	$\times$
8	$\times$	$(13, 0)$	$(10, 0)$	$(10, -2)$	$\times$

$\boxed{\text{Шаг 5}}$ На пятом шагу $n = 1$. Считаем значения для $f_1(p)$

$$f_1(0) = \begin{array}{c|l} 5 & g_1(5,0) + f_2(5) = 4 + 2 \cdot 5 + 3(5 + 0 - 5) + 23 = 37 \\ 6 & g_1(5,1) + f_2(6) = 4 + 2 \cdot 6 + 3(6 + 0 - 5) + 21 = 40 \\ 7 & g_1(5,2) + f_2(7) = 4 + 2 \cdot 7 + 3(7 + 0 - 5) + 16 = 40 \\ 8 & g_1(5,3) + f_2(8) = 4 + 2 \cdot 8 + 3(8 + 0 - 5) + 13 = 42 \end{array}$$

Занесём все данные в таблицу.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)	(f_5, q_5)
0	$(37, 5)$	$\times$	$\times$	$\times$	$\times$
4	$\times$	$\times$	$\times$	$\times$	$(8, 2)$
5	$\times$	$(23, 3)$	$\times$	$\times$	$(6, 1)$
6	$\times$	$(21, 2)$	$\times$	$\times$	$(0, 0)$
7	$\times$	$(16, 0)$	$(16, 1)$	$\times$	$\times$
8	$\times$	$(13, 0)$	$(10, 0)$	$(10, -2)$	$\times$

Поиск оптимальной стратегии

Заметим, что наше начальное состояние $p_0 = 0$, поскольку в самом начале было 0 рабочих

$$\begin{aligned} p_1^* &= p_0 = 0, & q_1^* &= 5, \\ p_2^* &= p_1^* + q_1^* = 5, & q_2^* &= 3, \\ p_3^* &= p_2^* + q_2^* = 8, & q_3^* &= 0, \\ p_4^* &= p_3^* + q_3^* = 8, & q_4^* &= -2, \\ p_5^* &= p_4^* + q_4^* = 6, & q_5^* &= 0. \end{aligned}$$

Итоговая стратегия — это $q^* = \{5, 3, 0, 0, -2\}$, то есть в начале 1-ой недели нанимаем 5 рабочих, на следующей еще 3, на третьей ничего не делаем, а на четвертой увольняем 2-их рабочих, на последней неделе ничего не делаем. При данной стратегии расход составит $f_1(0) = 37$.

2.6 Распределительная задача

Задача (распределительная)

Распределительная задача обобщает задачу о машине и задачу загрузки судна. В общем случае задача формулируется следующим образом: имеется ограниченное количество некоторого ресурса; ресурс можно использовать различными способами для получения дохода/расхода; требуется максимизировать доходы/минимизировать расходы от использования ресурса.

Математическая модель

- b — количество ресурса;
- N — количество способов использования ресурса;
- $x = 0, 1, 2, \ldots$ — интенсивность использования ресурса;
- $h_i(x)$ — затраты ресурса при использовании его i-ым способом с интенсивностью x;
- $c_i(x)$ — прибыль/убыток при использовании ресурса i-ым способом с интенсивностью x;
- $h_i(x)$ и $c_i(x)$ — возрастающие функции, при этом $c_i(0) = h_i(0) = 0$;

$$\sum_{i=1}^{N} c_i(x) \to \max_{(x_i)} / \min,$$

$$\sum_{i=1}^{N} h_i(x_i) \leq b,$$

$$x_i \leq a_i \quad i = 1 \ldots N.$$

Утверждение

Распределительная задача сводится к задаче выбора оптимальной стратегии N-шагового процесса.

Доказательство

Покажем, что исходная задача $\mathcal{P}$ сводится к задаче $\mathcal{Q}$. Для этого построим исходные данные для осуществления N-шагового процесса.

Исходные данные

- количество шагов процесса $= N$;
- состояние на каждом шаге — сколько ресурса можно израсходовать;
- решение на каждом шаге — какая будет интенсивность использования ресурса;
- функция перехода — сколько останется свободного ресурса после шага i, если в начале шага было p, а на шаге i ресурс использовался с интенсивностью q

$$T_i(p, q) = p - h_i(q), \quad i = 1 \ldots N;$$

- множества возможных состояний

$$P_1 = \{b\},$$

$$P_i = \{0, \ldots, b\}, \quad i = 2 \ldots N + 1;$$

- множества возможных решений

$$Q_i = \{0, \ldots, a_i\}, \quad i = 1 \ldots N;$$

- множества допустимых решений

$$Q_i(p) = \{q \in Q_i \mid T_i(p, q) \in P_{i+1}\}, \quad i = 1 \dots N;$$

- функция дохода/расхода

$$g_i(p, q) = c_i(q), \quad i = 1 \dots N.$$

После того, как исходные построены, можно непосредственно заняться сведением задачи $\mathcal{P}$ к задаче $\mathcal{Q}$.

Сведение к задаче выбора оптимальной стратегии

Для сведения исходной задачи $\mathcal{P}$ к задаче $\mathcal{Q}$ нужно

1. по исходным данным задачи $\mathcal{P}$ построить исходные данные задачи $\mathcal{Q}$;
2. по оптимальной стратегии задачи $\mathcal{Q}$ построить оптимальное решение задачи $\mathcal{P}$.

Пусть $q^* = \{q_1^*, q_2^*, \dots, q_N^*\}$ — оптимальная стратегии задачи $\mathcal{Q}$, тогда решение для исходной задачи $\mathcal{P}$ выглядит следующим образом

$$x^0 = (x_i^0), \quad x_i^0 = q_i^*.$$

Таким образом, первое уже сделано, а для второго нужно доказать, что x^0 — допустимое и оптимальное решение задачи $\mathcal{P}$.

Допустимость решения

- $q_i^* \le a_i$, поэтому из определения x_i^0 следует, что $\boxed{x_i^0 \le a_i}$.
- Пусть p_i^* — состояние на i-ом шаге, если следовать оптимальной стратегии q^*. Из выражения для множеств допустимых состояний следует, что $p_{N+1}^* \ge 0$, из выражения для функции перехода следует, что $p_{i+1}^* = p_i^* - h_i(q_i^*)$. Распишем имеющие неравенства

$$\begin{aligned}
0 \le p_{N+1}^* &= p_N^* - h_N(q_N^*) \\
&= \big(p_{N-1}^* - h_{N-1}(q_{N-1}^*)\big) - h_N(q_N^*) \\
&= \dots \\
&= -\sum_{i=1}^{N} h_i(q_i^*) + p_1^* \\
&\overset{P_1=\{b\}}{=} -\sum_{i=1}^{N} h_i(q_i^*) + b \\
&= b - \sum_{i=1}^{N} h_i(q_i^*),
\end{aligned}$$

$$\Downarrow$$

$$0 \le b - \sum_{i=1}^{N} h_i(q_i^*),$$

$$\boxed{\sum_{i=1}^{N} h_i(q_i^*) \le b}.$$

Таким образом, мы доказали, что x^0 — допустимое решение задачи $\mathcal{P}$, так как оно удовлетворяет всем ограничениям задачи $\mathcal{P}$.

Оптимальность решения

Заметим, что целевые функции в задачах $\mathcal{P}$ и $\mathcal{Q}$ одни и те же, поэтому нетрудно показать, что по сведению к другой задаче решение исходной задачи x^0, построенное на оптимальной стратегии q^*, будет оптимальным.

Таким образом, задача $\mathcal{P}$ действительно сводится к задаче $\mathcal{Q}$.

Решение (распределительная задача)

Благодаря тому, что распределительная задача сводится к задаче выбора оптимальной стратегии N-шагового процесса, её можно решить соответствующим образом.

Вспомним, что $q = 0, 1, \dots a$, то есть $\boxed{0 \le q \le a_i}$. Распишем функцию перехода

$$T_i(p, q) \in P_{i+1},$$

$$\Updownarrow$$

$$0 \le p - h_i(q) \le b,$$

$$\Downarrow$$

$$\boxed{h_i(q) \le p}.$$

Таким образом, решение q на i-ом шаге в состоянии p является допустимым тогда и только когда, когда

$$\begin{cases} 0 \le q \le a_i, \\ h_i(q) \le p. \end{cases}$$

База процесса

Пусть $p \in P_N$, запишем выражение для $f_N(p)$

$$f_N(p) = \max_{q \in Q_N(p)} \{c_N(q)\}.$$

Вспомним, что $\{c_i(x)\}$ — возрастающие функции, поэтому максимум $c_N(q)$ достигается при $q_{\max} \in (p)$. Таким образом

$$f_N(p) = c_N(q_{\max}),$$

при $q_{\max}$, которое удовлетворяет ограничениям

$$\begin{cases} 0 \le q \le a_i, \\ h_i(q) \le p. \end{cases}$$

Переход

Пусть $n = 1 \dots N - 1$, $p \in P_n$, запишем выражение для $f_n(p)$

$$f_n(p) = \max_{q \in Q_n(p)} \left\{c_n(q) + f_{n+1}\big(\underbrace{T_i(p, q)}_{p - h_n(q)}\big)\right\},$$

$$q \in Q_n(p) \quad \Longleftrightarrow \quad \begin{cases} 0 \le q \le a_i, \\ h_i(q) \le p. \end{cases}$$

С помощью данных рекуррентных соотношений можно осуществить алгоритм выбора оптимальной стратегии N-шагового процесса и найти оптимальную стратегию распределительной задачи.

Утверждение

Оценка трудоёмкости алгоритма A решения распределительной задачи с помощью N-шагового процесса принятия решений имеет вид

$$T_A(p) \le b \cdot |p|.$$

Доказательство

Напишем выражение трудоёмкости алгоритма выбора оптимальной стратегии N-шагового процесса

$$T_A = N \cdot b \cdot a,$$

- N — число шагов алгоритма;
- b — количество возможных состояний на каждом шаге;
- $a = \max\{a_1, a_2, \ldots, a_N\}$ — количество перебираемых решений на каждом шаге для каждого состояния.

Напишем выражение оценки трудоёмкости через длину входа задачи. Для хранения условия задачи в памяти нужно хранить значения всех констант и всех функций во всех возможных точках. Будем учитывать лишь хранение значений функций, поскольку занимаемый объём памяти для всего остального можно оценить как $\mathcal{O}(1)$. В данной задаче у нас есть функции $\{c_i(x)\}_{i=1}^N$ и $\{h_i(x)\}_{i=1}^N$, при этом каждая функция определена в a точках. Занимаемый объём памяти функциями составляет $2 \cdot N \cdot a$, запишем это так

$$|p| = \mathcal{O}(N \cdot a),$$

$$\Downarrow$$

$$T_A = N \cdot b \cdot a = b \cdot N \cdot a \le b \cdot |p|,$$

$$\Downarrow$$

$$\boxed{T_A(p) \le b \cdot |p|}.$$

Замечание

Нельзя написать, что

$$T_A(p) \le C \cdot |p|,$$

так как нет такой константы C, чтобы неравенство выполнялась для любой задачи, относящейся к распределительной, поскольку какое бы C мы не взяли, всегда можно написать распределительную задачу, в которой $b = C + 1$. Таким образом, алгоритм решения распределительной задачи с помощью выбора оптимальной стратегии N-шагового процесса не является полиномиальным (он является *псевдополиномиальным*).

2.6.1 Задача о рюкзаке

Задача (о рюкзаке)

Задача о рюкзаке является частным случаем распределительной задачи. Условие формулируется следующим образом: имеется рюкзак ограниченной вместимости и некоторые предметы, которые нужно сложить в рюкзак так, чтобы общая стоимость предметов в рюкзаке была максимальной.

Математическая модель

- b — вместимость рюкзака;
- N — количество предметов;
- a_i — место, занимаемое i-ым предметом;
- c_i — ценность i-го предмета;
- x_i — сколько предметов типа i класть в рюкзак, $x_i \in \{0, 1\}$ или $x_i \in \{0, 1, 2, \dots\}$;

$$\sum_{i=1}^{N} c_i x_i \to \max_{(x_i)},$$

$$\sum_{i=1}^{N} a_i x_i \leq b.$$

Решение (задача о рюкзаке)

Поскольку задача является частным случаем распределительной задачи, её тоже можно решить с помощью N-шагового процесса принятия решений.

База процесса

$$f_N(p) = \begin{cases} c_N, & p \geq a_N, \\ 0, & p < a_N; \end{cases}$$

$$q_N(p) = \begin{cases} 1, & p \geq a_N, \\ 0, & p < a_N. \end{cases}$$

Переход

$$f_n(p) = \begin{cases} \max\limits_{x \in \{0,1\}} \{c_n x + f_{n+1}(p - a_n x)\}, & p \geq a_n, \\ f_{n+1}(p), & p < a_n. \end{cases}$$

Утверждение

Оценка трудоёмкости алгоритма A решения задачи о рюкзаке с помощью N-шагового процесса принятия решений имеет вид

$$\boxed{T_A(p) \leq b \cdot |p|}.$$

Замечание

Полиномиального решения задачи о рюкзаке на данный момент не существует.

2.7 Задача о ближайшем соседе

Задача (о ближайшем соседе)

Есть некоторая трасса, представленная набором точек, требуется найти точки разбиения трассы на участки так, чтобы сумма расстояний между точками разбиения была минимальной.

Математическая модель

- $Z = \{a, a+1, a+2, \ldots, b-1, b\}$ — множество точек трассы;
- $N+1$ — количество точек разбиения;
- $f(x, y)$ — функция расстояния между x и y; $x \leq y$, $x \in Z$, $y \in Z$. Это может быть, например, функция затрат на содержанию телевышек, а не просто расстояние между точками.

$$\min_{(x_1, x_2, \ldots, x_N, x_N+1)} \left\{ \sum_{i=1}^{N} f(x_i, x_{i+1}) \right\},$$

$$\begin{cases} x_1 = a, \\ x_1 \leq x_2, \\ x_2 \leq x_3, \\ \ldots \\ x_N \leq x_{N+1}, \\ x_{N+1} = b; \end{cases}$$

$$x_i \in Z.$$

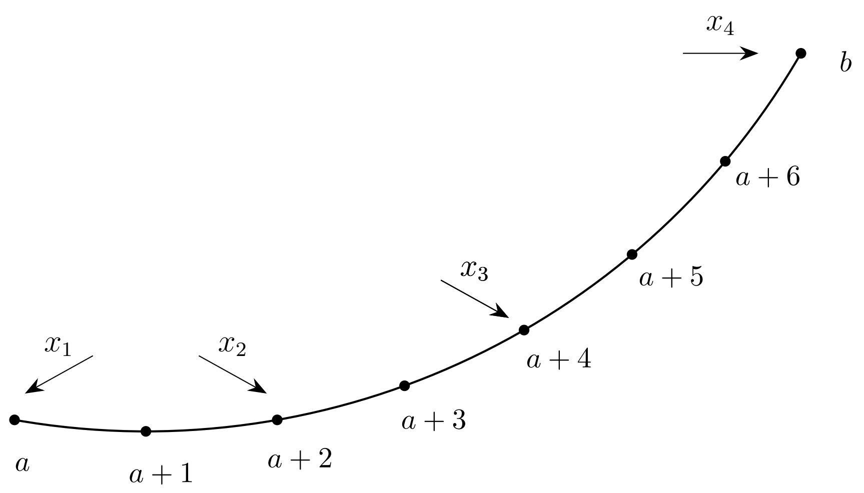


Рис. 2.2: Пример разбиения трассы на 3 сегмента $(x_1 \to x_2)$, $(x_2 \to x_3)$, $(x_3 \to x_4)$. $N = 3$, $b = a + 7$

Утверждение

Задача о ближайшем соседе сводится к задаче выбора оптимальной стратегии N-шагового процесса.

Доказательство

Покажем, что исходная задача $\mathcal{P}$ сводится к задаче $\mathcal{Q}$. Для этого построим исходные данные для осуществления N-шагового процесса.

Исходные данные

- количество шагов процесса $= N$;
- состояние на каждом шаге — координата текущей точки;
- решение на каждом шаге — насколько выдвинуться вперёд;
- функция перехода
$$T_i(p,q) = p+q;$$
- множества возможных состояний
$$P_1 = \{a\},$$
$$P_i = Z, \quad i = 2 \dots N,$$
$$P_{N+1} = \{b\};$$
- множества возможных решений
$$Q_i = \{0, \dots, b-a\} \quad i = 1 \dots N;$$
- множества допустимых решений
$$Q_i(p) = \{q \in Q_i \mid \underbrace{T_i(p,q)}_{p+q} \leq b\};$$
- функция расхода
$$g_i(p,q) = f(p, p+q).$$

После того, как исходные данные построены, можно непосредственно заняться сведение задачи $\mathcal{P}$ к задаче $\mathcal{Q}$.

Сведение к задаче выбора оптимальной стратегии

Пусть для задачи $\mathcal{Q}$ мы построили оптимальную стратегию $q^* = \{q_1^*, q_2^*, \dots, q_N^*\}$, однако по построению для решения исходной задачи нам нужны не значения $\{q_i^*\}_{i=1}^N$, а значения $\{p_i^*\}_{i=1}^N$
$$x_i^0 = p_i^*, \quad i = 1 \dots N.$$

Действительно, состояние — координата текущей точки на каждом шаге, а по условию исходной задачи нам и нужны координаты точек. В отличие от распределительной задачи, в которой для построения оптимального решения нам нужны были не состояния $\{p_i^*\}_{i=1}^N$, а решения $\{q_i^*\}_{i=1}^N$.

Допустимость и оптимальность решения

$$\underbrace{p_{i+1}^*}_{x_{i+1}^0} = \underbrace{p_i^*}_{x_i^0} + \underbrace{q_i^*}_{\geq 0},$$

поэтому $x_i^0 \leq x_{i+1}^0$, значит решение исходной задачи x^0 является оптимальным.

Решение (задача о ближайшем соседе)

Благодаря тому, что задача о ближайшем соседе сводится к задаче выбора оптимальной стратегии N-шагового процесса, её можно решить соответствующим образом.

Будем рассматривать семейства задач (n, p), $f_n(p)$ — минимальная сумма расстояний между точками, если текущая координата равняется p, и мы можем добавить в маршрут лишь $(N - n)$ точек.

База процесса

Пусть $n = N$, тогда нам необходимо продвинуться вперёд ровно на $b - p$, поскольку конечная точка маршрута по условию обязана быть b, а на данном шаге мы находимся в точке p. Также ясно, что расстояние между текущей и следующей точкой равняется $f(p, b)$. То есть

$$f_N(p) = f(p, b),$$

$$q_N(p) = b - p.$$

Переход

Пусть $n = 1 \dots N - 1$, $p \in P_n$. Запишем выражение для $f_n(p)$

$$\begin{aligned} f_n(p) &= \min_{q \in Q_i(p)} \{f(p, p+q) + f_{n+1}(p+q)\} \\ &= \min_{0 \le q \le b-p} \{f(p, p+q) + f_{n+1}(p+q)\} \\ &\overset{p'=p+q}{=} \min_{p \le p' \le b} \{f(p, p') + f_{n+1}(p')\}. \end{aligned}$$

С помощью данных рекуррентных соотношений можно осуществить алгоритм выбора оптимальной стратегии N-шагового процесса и найти оптимальную стратегию задачи о ближайшем соседе.

Утверждение

Оценка трудоёмкости алгоритма A решения распределительной задачи с помощью N-шагового процесса принятия решений имеет вид

$$\boxed{T_A(p) \le C \cdot |p|^2}.$$

Доказательство

Напишем выражение трудоёмкости алгоритма выбора оптимальной стратегии N-шагового процесса

$$T = N \cdot (b - a) \cdot (b - a),$$

- N — число шагов алгоритма;
- $(b - a)$ — количество возможных состояний на каждом шаге;
- $(b - a)$ — количество перебираемых значений p' на каждом шаге для каждого состояния.

Заметим, что если $N \ge b - a + 1$, то для некоторых i $p_i = p_{i+1}$, поэтому есть смысл рассматривать лишь $N \ge b - a + 1$. Поэтому можно сказать, что

$$N = \mathcal{O}(b - a),$$

$$\Downarrow$$

$$T = \mathcal{O}\big((b-a)^3\big).$$

Для хранения условия задачи нужно хранить значения функции $f(x,y)$ на множестве Z^2 — это $\mathcal{O}(N^2)$, плюс другие исходные данные задачи, которые можно оценить как $\mathcal{O}(1)$. Таким образом

$$|p| = \mathcal{O}(N^2) = \mathcal{O}((b-a)^2),$$

$$\Downarrow$$

$$T_A(p) = \mathcal{O}(|p|^{1.5}),$$

$$\Downarrow$$

$$T_A(p) \leq C \cdot |p|^2.$$

Следствие

Алгоритм решения задачи о ближайшем соседе с помощью выбора оптимальной стратегии N-шагового процесса является полиномиальным.

2.8 Задача о ракетах

Задача (о ракетах)

Имеются 7 ракет для поражения четырех целей. Известны вероятности λ_i поражения i-ой цели одной ракетой:

Цель (i)	1	2	3	4
Вероятность поражения (λ_i)	0.5	0.6	0.9	0.8

Сколько ракет нужно направить на каждую цель, чтобы вероятность поражения всей совокупности целей была наибольшей?

Математическая модель

Пусть $i = 1 \ldots 4$ — номер цели.

Параметры

- $\lambda = \{0.5, 0.6, 0.9, 0.8\}$ — вероятности поражения целей.

Переменные

- $q = \{q_i\}_{i=1}^4$ — сколько нужно направить ракет на i-ую цель.

Использование теории вероятности

Допустим мы выпускаем одну ракету во вторую цель и поражаем её — это событие X, отметим $P(X)$ как вероятность события X. Из условия $P(X) = \lambda_2 = 0.6$. Отметим $\bar{X}$ — событие противоположное X, в текущем контексте это событие того, что мы выпустили ракету во вторую цель, но не поразили её. Вероятность такого события равна $1 - P(X)$, т.е $P(\bar{X}) = 1 - P(X)$.

Пусть у нас есть 2 события X, Y, они происходят одновременно и друг от друга не зависят (мы запустили ракету в 1 и 3 цель, например), тогда вероятность того, что эти события произойдут одновременно, равна $P(AB) = P(A) \cdot P(B)$.

Пусть мы запустили c ракет в один объект: каждый запуск ракеты это событие X_i $i \in \{1, 2, \cdots, c\}$, при этом эти события независимы друг от друга и имеют одинаковую вероятность $P(X)$. Тогда вероятность того, что при запуске всех c ракет по одной цели, мы ни разу не поразили цель, равна $\prod\limits_{i=1}^{c}\big(1 - P(X)\big) = \big(1 - P(X)\big)^c$. То есть вероятность того, что мы поразим цель i, выпустив по ней c ракет, равна

$$1 - \big(1 - P(X)\big)^c. \tag{*}$$

Математическая формулировка задачи

У нас имеется 4 цели, в каждую из которых мы запускаем q_i ракет. Фактически это 4 независимых события, описанные выше, и мы знаем вероятность поражения каждой цели хотя бы 1 ракетой при запуске q_i ракет (*), из этого получаем итоговую вероятность поражения всех целей в совокупности

$$G = \prod_{i=1}^{4}\big(1 - (1-\lambda_i)^{q_i}\big) \to \max_q,$$

$$\sum_{i=1}^{4} q_i \leq 7,$$

$$\forall i \quad q_i \in \{1, 2, 3, 4\}.$$

Ограничение на q_i следует из двух фактов

1. если мы по некоторой цели i не запустим ни одной ракеты ($q_i = 0$), то $G = 0$, поэтому решение точно будет неоптимальным;
2. если на каком-нибудь шаге мы запустим по некоторой цели более 4 ракет, то значит на оставшиеся 3 цели останется не более 2 ракет, значит поразить все цели точно не получится ($G = 0$).

Решение

Решим задачу с помощью N-шагового процесса принятия решений с теми же данными, которые были определены в задаче и с учетом условий на q.

Исходные данные

- количество шагов процесса = количество целей = 4;
- на i-ом шаге будем определять сколько нужно запустить ракет в i-ую цель (q_i);
- состояние — оставшееся количество ракет;
- функция перехода — сколько ракет останется, если из имеющихся p будет запущено q

$$T_i(p, q) = p - q;$$

- множества возможных состояний; учтём, что в каждую цель надо выпустить минимум 1 ракету, а так же то, что для последующих целей надо оставить минимум по 1 ракете и того, что изначально было 7 ракет

$$P_1 = \{7\}, \quad P_2 = \{3, 4, 5, 6\}, \quad P_3 = \{2, 3, 4, 5\},$$

$$P_4 = \{1, 2, 3, 4\}, \quad P_5 = \{0\}.$$

Множество P_5 фиктивное, оно необходимо, чтобы указать что в четвертую цель мы запускаем все оставшиеся ракеты, это объясняется формулой (*), указывающей на то, что чем больше запускается ракет в цель, тем выше вероятность поразить её хотя бы 1 ракетой.

- множества возможных решений;

$$Q_i = \{1, 2, 3, 4\};$$

- множества допустимых решений

$$Q_i(p) = \{q \in \{1, 2, 3, 4\} \mid \underbrace{T_i(p, q)}_{p-q} \in P_{i+1}\};$$

- функция вероятности — вероятность поразить i-ую цель, если в неё запустить q ракет

$$g_i(q) = 1 - (1 - \lambda_i)^q.$$

Для решения будем рассматривать семейства задач, которые определяются парой (n, p). Фактически это означает, что

- рассматриваем цели $n, n+1, \ldots, 4$, $n \le 4$;
- количество ракет, которое имеется в наличии $p \in P_n$.

База процесса

Пусть $f_n(p)$ — наибольшая вероятность поразить совокупность целей в семействе задач (n, p). Заметим, что если $n = 4$, то задача легко решается: нужно запустить в цель все ракеты, то есть $\forall p \in P_4$

$$\begin{cases} f_4(p) = g_4(p), \\ q_4(p) = p. \end{cases}$$

Переход

Теперь когда у нас есть база для решения задачи, осуществим переход $f_{n+1}(p) \to f_n(p)$ в предположении, что $f_{n+1}(p)$ мы знаем $\forall p \in P_{n+1}$.

$$\boxed{f_n(p) = \max_{q \in Q_n(p)} \left\{ g_n(q) \cdot f_{n+1}(\underbrace{T_n(p, q)}_{p-q}) \right\}}. \qquad (**)$$

Вспомним множества допустимых состояний P_i. В объединении всех P_i лежит множество $\{0, 1, 2, 3, 4, 5, 6, 7\}$ — все значения, которые может принимать p. Также понятно, что считать $f_n(p)$ для $p \notin P_n$ не имеет смысла, так как эти состояния недопустимы. Исходя из этого составим следующую таблицу

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)
1	$\times$	$\times$	$\times$	
2	$\times$	$\times$		
3	$\times$			
4	$\times$			
5	$\times$			$\times$
6	$\times$		$\times$	$\times$
7		$\times$	$\times$	$\times$

Значение максимальной вероятности в исходной задаче равняется $\mathbf{f_1(7)}$. Заполним таблицу, чтобы найти его, а затем вычислить все q_i^*.

Заполнение таблицы

$\boxed{\text{Шаг 1}}$ На первом шаге $n = 4$, запишем выражение для $f_4(p)$

$$f_4(p) = g_4(p) = 1 - (1 - \lambda_4)^p = 1 - 0.2^p.$$

Посчитаем значение f_4 для всех возможных состояний

$$f_4(1) = g_4(1) = 1 - 0.2^1 = 0.8;$$

$$f_4(2) = g_4(2) = 1 - 0.2^2 = 0.96;$$

$$f_4(3) = g_4(3) = 1 - 0.2^3 = 0.992;$$

$$f_4(4) = g_4(4) = 1 - 0.2^4 = 0.9984;$$

Занесём все эти данные в последний столбец таблицы.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)
1	$\times$	$\times$	$\times$	$(0.8, 1)$
2	$\times$	$\times$		$(0.96, 2)$
3	$\times$			$(0.992, 3)$
4	$\times$			$(0.9984, 4)$
5	$\times$			$\times$
6	$\times$		$\times$	$\times$
7		$\times$	$\times$	$\times$

$\boxed{\text{Шаг 2}}$ На втором шаге $n = 3$. Выражение для этого и последующих шагов выглядит как рекуррентное соотношение (**).

$$f_3(2) = \begin{array}{c|l} 1 & g_3(1) \cdot f_4(1) = (1 - 0.1) \cdot 0.8 = 0.72 \end{array}$$

$$f_3(3) = \begin{array}{c|l} 1 & g_3(1) \cdot f_4(2) = (1 - 0.1) \cdot 0.96 = 0.864 \\ 2 & g_3(2) \cdot f_4(1) = (1 - 0.1^2) \cdot 0.8 = 0.792 \end{array}$$

$$f_3(4) = \begin{array}{c|l} 1 & g_3(1) \cdot f_4(3) = (1 - 0.1) \cdot 0.992 = 0.8928 \\ 2 & g_3(2) \cdot f_4(2) = (1 - 0.1^2) \cdot 0.96 = 0.9504 \\ 3 & g_3(3) \cdot f_4(1) = (1 - 0.1^3) \cdot 0.8 = 0.7992 \end{array}$$

$$f_3(5) = \begin{array}{c|l} 1 & g_3(1) \cdot f_4(4) = (1 - 0.1) \cdot 0.9984 = 0.89856 \\ 2 & g_3(2) \cdot f_4(3) = (1 - 0.1^2) \cdot 0.992 = 0.98208 \\ 3 & g_3(3) \cdot f_4(2) = (1 - 0.1^3) \cdot 0.96 = 0.95904 \\ 4 & g_3(4) \cdot f_4(1) = (1 - 0.1^4) \cdot 0.8 = 0.79992 \end{array}$$

Занесём все данные в таблицу.

p	(f_1,q_1)	(f_2,q_2)	(f_3,q_3)	(f_4,q_4)
1	$\times$	$\times$	$\times$	$(0.8, 1)$
2	$\times$	$\times$	$(0.72, 1)$	$(0.96, 2)$
3	$\times$		$(0.864, 1)$	$(0.992, 3)$
4	$\times$		$(0.9504, 2)$	$(0.9984, 4)$
5	$\times$		$(0.98208, 2)$	$\times$
6	$\times$		$\times$	$\times$
7		$\times$	$\times$	$\times$

$\boxed{\text{Шаг 3}}$ На третьем шаге $n = 2$. Считаем значения для $f_2(p)$

$$f_2(3) = \begin{array}{c|l} 1 & g_2(1)\cdot f_3(2) = (1-0.4)\cdot 0.72 = 0.432 \end{array}$$

$$f_2(4) = \begin{array}{c|l} 1 & g_2(1)\cdot f_3(3) = (1-0.4)\cdot 0.864 = 0.5184 \\ 2 & g_2(2)\cdot f_3(2) = (1-0.4^2)\cdot 0.72 = 0.6048 \end{array}$$

$$f_2(5) = \begin{array}{c|l} 1 & g_2(1)\cdot f_3(4) = (1-0.4)\cdot 0.9504 = 0.57204 \\ 2 & g_2(2)\cdot f_3(3) = (1-0.4^2)\cdot 0.864 = 0.72576 \\ 3 & g_2(3)\cdot f_3(2) = (1-0.4^3)\cdot 0.72 = 0.67392 \end{array}$$

$$f_2(6) = \begin{array}{c|l} 1 & g_2(1)\cdot f_3(5) = (1-0.4)\cdot 0.98208 = 0.589248 \\ 2 & g_2(2)\cdot f_3(4) = (1-0.4^2)\cdot 0.9504 = 0.798336 \\ 3 & g_2(3)\cdot f_3(3) = (1-0.4^3)\cdot 0.864 = 0.808704 \\ 4 & g_2(4)\cdot f_3(2) = (1-0.4^4)\cdot 0.72 = 0.701568 \end{array}$$

Занесём все данные в таблицу.

p	(f_1,q_1)	(f_2,q_2)	(f_3,q_3)	(f_4,q_4)
1	$\times$	$\times$	$\times$	$(0.8, 1)$
2	$\times$	$\times$	$(0.72, 1)$	$(0.96, 2)$
3	$\times$	$(0.432, 1)$	$(0.864, 1)$	$(0.992, 3)$
4	$\times$	$(0.6048, 2)$	$(0.9504, 2)$	$(0.9984, 4)$
5	$\times$	$(0.72576, 2)$	$(0.98208, 2)$	$\times$
6	$\times$	$(0.808704, 3)$	$\times$	$\times$
7		$\times$	$\times$	$\times$

$\boxed{\text{Шаг 4}}$ На четвертом шаге $n = 1$. Считаем значения для $f_1(p)$

$$f_1(7) = \begin{array}{c|l} 1 & g_1(1)\cdot f_2(6) = (1-0.5)\cdot 0.808704 = 0.404352 \\ 2 & g_1(2)\cdot f_2(5) = (1-0.5^2)\cdot 0.72576 = 0.54432 \\ 3 & g_1(3)\cdot f_2(4) = (1-0.5^3)\cdot 0.6048 = 0.5292 \\ 4 & g_1(4)\cdot f_2(3) = (1-0.5^4)\cdot 0.432 = 0.405 \end{array}$$

Занесём все данные в таблицу.

p	(f_1, q_1)	(f_2, q_2)	(f_3, q_3)	(f_4, q_4)
1	$\times$	$\times$	$\times$	$(0.8, 1)$
2	$\times$	$\times$	$(0.72, 1)$	$(0.96, 2)$
3	$\times$	$(0.432, 1)$	$(0.864, 1)$	$(0.992, 3)$
4	$\times$	$(0.6048, 2)$	$(0.9504, 2)$	$(0.9984, 4)$
5	$\times$	$(0.72576, 2)$	$(0.98208, 2)$	$\times$
6	$\times$	$(0.808704, 3)$	$\times$	$\times$
7	$(0.54432, 2)$	$\times$	$\times$	$\times$

Поиск оптимальной стратегии

Заметим, что наше начальное состояние $p_0 = 7$, поскольку в самом начале было 7 ракет

$$\begin{aligned} p_1^* &= p_0 = 7, & q_1^* &= 2, \\ p_2^* &= p_1^* - q_1^* = 5, & q_2^* &= 2, \\ p_3^* &= p_2^* - q_2^* = 3, & q_3^* &= 1, \\ p_4^* &= p_3^* - q_3^* = 2, & q_4^* &= 2. \end{aligned}$$

Итоговая стратегия — это $q^* = \{2, 2, 1, 2\}$, т.е. в первую, вторую, четвертую цель запускаем по 2 ракеты, в третью одну. При данной стратегии вероятность поражения совокупности целей составит $f_1(7) = 0.54432$.

При заполнении таблицы для удобства можно было округлять значения $f_n(p)$ до второго знака после запятой.

2.9 Многошаговый процесс принятия решений

Мы рассмотрели один из подходов динамического программирования — N-шаговый процесс принятия решений, который позволяет решать задачи, если число шагов процесса заранее известно. Однако встречаются задачи, в которых число шагов для получения оптимальной стратегии может быть заранее неизвестно, а значит к таким задачам нужен новый подход.

Определение

Бинарное отношение $\succ$ на множестве X будем называть *отношением частичного строгого порядка*, если оно обладает следующими свойствами

1. антирефлексивность: $\forall x \in X : (x \not\succ x)$;
2. антисимметричность: $\forall x, y \in X : (x \succ y) \;\; \& \;\; (y \succ x) \Rightarrow (x = y)$;
3. транзитивность: $\forall x, y, z \in X : (x \succ y) \;\&\; (y \succ z) \Rightarrow (x \succ z)$.

Если $x \succ y$, то будем говорить, что «x *следует за* y», а «y *предшествует* x».

Замечание

Слово «частичный» говорит о том, что отношение определено не для всех пар элементов множества X (есть *несравнимые элементы*).

Определение

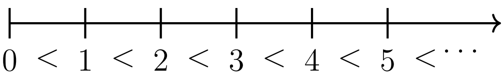
Элемент $x \in X$ называется *максимальным*, если

$$\forall y \in X \quad (y \succ x) \Rightarrow (y = x).$$

Это означает, что во множестве X нет элементов, которые бы следовали за максимальными.

Пример

Примером отношения строгого порядка является числовая прямая $\langle \mathbb{R}, > \rangle$, поскольку можно написать, что $1 > 0$, $2 > 1$, $2.5 > 2.4$ и так далее. Однако здесь порядок не частичный, а линейный, поскольку написать $x > y$ можно для любых двух чисел x и y, если $x \neq y$. Также можно взять не всё множество $\mathbb{R}$, а, например, интервал $[a, b]$. Максимальным элементом на интервале $[a, b]$ является элемент b.



Пример

Пусть X — множество аудиторий корпуса некоторого университета, $x, y \in X$. Пусть $\boxed{x \succ y}$, если аудитория x находится на более высоком этаже, чем аудитория y.

Это отношение является отношением частичного строгого порядка, потому что сравнить между собой можно лишь аудитории на разных этажах. Значит на множестве X есть несравнимые элементы — аудитории на одном этаже. Максимальными аудиториями будут те, которые находятся на самом высоком этаже.

Алгоритм (многошаговый процесс принятия решений)

Общие соображения

В N-шаговом процессе принятия решений у нас были состояния и решения на каждом шаге, начальное состояние, фиксированное число шагов, функция перехода и целевая функция, значение которой нужно оптимизировать. Теперь у нас всё то же самое, но количество шагов заранее неизвестно. Возникает вопрос: а как понять, сколько шагов совершить, то есть в какой момент остановить алгоритм?

Пусть P — множество возможных состояний, на котором определено отношение частичного строгого порядка $\succ$. Определим условие остановки следующим образом: алгоритм останавливается, если текущее состояние p принадлежит множеству конечных состояний $\bar{P}$. Определим $\bar{P}$ как множество максимальных элементов множества P в смысле отношения $\succ$, то есть

$$\bar{P} = \left\{ x \in P \;\middle|\; \forall y \in P : x \not\succ y \right\}.$$

Итого, нам известно множество возможных состояний P, начальное состояние p_0 и множество конечных состояний $\bar{P} \subseteq P$, тогда мы получаем задачу поиска пути $(p_1^* = p_0, p_2^*, p_3^*, \cdots, p_k^*)$, который даст оптимальное значение целевой функции исходной задачи (например, минимальные затраты), при этом $p_k^* \in \bar{P}$ и все переходы осуществимы, то есть $p_{i+1}^* = T(p_i^*, q)$. Заметим, что в таком случае количество шагов, которое нужно будет совершить, равняется k.

Для корректности алгоритма необходимо на множестве P определить правила того, как можно перемещаться между состояниями в соответствии с задачей, иначе мы сможем ходить циклами, строя неоптимальные маршруты, а оптимальный так и не получим. Для этого необходимо, чтобы функция перехода $T(p, q)$ была «согласована» с отношением $\succ$ на множестве P, то есть чтобы

$$\forall p \in P \quad T(p, q) \succ p.$$

Выполнимость этого гарантирует, что какие бы решения мы не принимали, мы сможем попасть в конечное состояние.

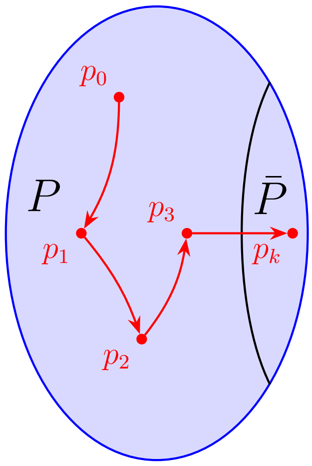


Рис. 2.3: Пример пути из p_0 в p_k $(k = 4)$

Подход к выбору оптимальной стратегии

В N-шаговом процессе принятия решений мы рассматривали семейства задач (n, p) и строили оптимальную стратегию, идя от $n = N$ к $n = 1$ и минимизируя значение функции дохода $g_n(p, q)$. При этом мы шли от состояний, которые были на шаге $n = N$, к начальному состоянию на шаге $n = 1$ («обратный ход»). Здесь мы также будем на каждом шаге оптимизировать функцию дохода $g(p, q)$, но идти будем от конечных состояний к начальному.

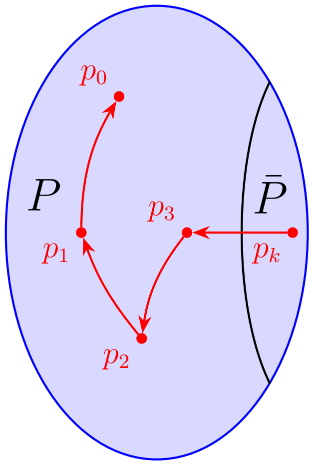


Рис. 2.4: Пример «обратного хода» из p_k в p_0 $(k = 4)$

Для этого на каждом шаге будем определять текущее множество конечных состояний. На самом первом шаге это множество $\bar{P}_0 = \bar{P}$ (из условия), на втором шаге это $\bar{P}_1 = \max(P \setminus \bar{P}_0)$, на третьем шаге это $\bar{P}_2 = \max(P \setminus \bar{P}_0 \setminus \bar{P}_1)$ и так далее, где $\max(A)$ — множество максимальных элементов множества A. Данный выбор множеств конечных состояний обусловлен следующим: для завершения процесса нам нужно оказаться в состоянии из множества $\bar{P}$, оказаться в состоянии из этого множества за один шаг можно, если находимся в состоянии из $\bar{P}_1$; оказаться в состоянии из $\bar{P}_1$ можно за один шаг, если находимся в состоянии из $\bar{P}_2$ и так далее. Таким образом, получается цепочка переходов между множествами конечных состояний

$$\bar{P}_k \to \bar{P}_{k-1} \to \cdots \to \bar{P}_3 \to \bar{P}_2 \to \bar{P}_1 \to \bar{P}_0 = \bar{P},$$

при этом $p_o \in \bar{P}_k$ и каждый переход осуществим за шаг.

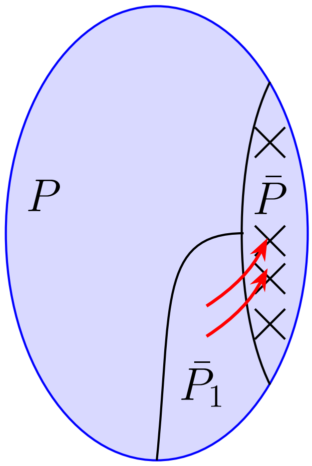


Рис. 2.5: Иллюстрация множеств конечных состояний

По аналогии с алгоритмом выбора оптимальной стратегии N-шагового процесса введём функцию $f(p)$ (теперь она не имеет индекса, поскольку нет конкретным шагов). Эта функция будет обозначать оптимальное значение целевой функции исходной задачи, если мы находимся в состоянии p. Фактически эта функция показывает оптимальное значение целевой функции при некотором оптимальном пути из состояния p в множество конечных состояний

$\bar{P}$. Как и в алгоритме выбора оптимальной стратегии N-шагового процесса, для каждого состояния p будем рассчитывать и запоминать значение $f(p)$ вместе с оптимальным решением.

Алгоритм

Для осуществления алгоритма нужны следующие исходные данные

- *состояние* на каждом шаге. Обозначение: $\boxed{p}$;
- *множество возможных состояний*, на котором определено отношение частичного строгого порядка $\succ$. Обозначение: $\boxed{P} = \{p_0, p_1, p_2, \dots\}$;
- *начальное состояние*. Обозначение: $\boxed{p_0}$;
- *множества конечных состояний*

$$\bar{P}_0 = \bar{P},$$

$$\forall k > 0 \quad \bar{P}_k = \max(P \setminus \bar{P}_0 \setminus \bar{P}_1 \dots \setminus \bar{P}_{k-1});$$

- *решение* на каждом шаге. Обозначение: $\boxed{q}$;
- *множество возможных решений*. Обозначение $\boxed{Q}$;
- *функция перехода* — функция перехода из текущего состояния p в новое состояние p', если принимается решение q. Обозначение: $\boxed{T(p,q)}$.

$$p' = T(p,q),$$

$$\forall p \in P \qquad \underbrace{T(p,q) \succ p}_{\text{«согласованность»}}$$

- *множество допустимых решений* в состоянии p. Обозначение: $\boxed{Q(p)}$;
- *функция дохода* — оптимальное значение целевой функции на одном шаге, если в состоянии p принимается решение q. Обозначение: $\boxed{g(p,q)}$;
- $\boxed{f(p)}$ — оптимальное значение целевой функции, если вначале было состояние p.

Рекуррентные соотношения

$$\forall p \in \bar{P} \quad f(p) = 0,$$
$$\forall p \notin \bar{P} \quad f(p) = \min_{q \in Q(p)} \Big\{ g(p,q) + f\big(T(p,q)\big) \Big\}.$$

Значение $f(p)$ для конечного состоянии p определяется конкретной задачей. Так, если в задаче на всех шагах предполагается суммирование (как в задаче о машине), то $f(p) = 0$, а если в задаче предполагается умножение (как в задаче о ракетах), то $f(p) = 1$ и так далее.

В общем случае $f(p)$ может записываться не только с помощью $\min$, но и с помощью $\max$. Также операция между $g(p,q)$ и $f\big(T(p,q)\big)$ может быть как сложение, так и умножение, взятие минимума/максимума и так далее.

Алгоритм

Все промежуточные результаты по аналогии с алгоритмом выбора оптимальной стратегии принятия решений будут фиксироваться в следующей таблице

p	(f, q)
p_0	
p_1	
p_2	
$\dots$	

Сначала на каждом шаге считаем значения $f(p)$ для p из текущего множества конечных состояний и фиксируем в таблице значение $f(p)$ и решение, на котором достигается оптимальное значение целевой функции — «обратный ход».

После того, как таблица заполнена, будем выбирать оптимальную стратегию — «прямой ход». Делать это будем следующим образом

$$\begin{aligned} p_1^* &= p_0, & q_1^* &= q(p_1^*), \\ p_2^* &= T(p_1^*, q_1^*), & q_2^* &= q(p_2^*), \\ p_3^* &= T(p_2^*, q_2^*), & q_3^* &= q(p_3^*), \\ &\dots & &\dots \\ p_k^* &= T(p_{k-1}^*, q_{k-1}^*), & q_k^* &= q(p_k^*), \\ p_{k+1}^* &= T(p_k^*, q_k^*) \in \bar{P}. \end{aligned}$$

Заметим, что мы закончили в тот момент, когда очередное состояние p_{k+1}^* оказалось конечным.

По итогам «прямого хода» получим стратегию $q^* = \{q_1^*, q_2^*, \dots, q_{k-1}^*, q_k^*\}$, при этом количество шагов k определяется именно в этот момент.

2.10 Задача о дороге

Задача (о дороге)

Дорога состоит из 8 последовательных участков. Для поддержания её в нормальном состоянии решено разместить вдоль дороги дорожно-ремонтные станции. Известно что затраты на обслуживание участка от x до y одной станцией составляет величину $f(x, y)$, задаваемую таблицей. Известно также, что одна станция не может обслуживать отрезок дороги, содержащий более 4 участков. Каким образом разделить дорогу между станциями, чтобы затраты на обслуживание дороги были минимальными?

x/y	0	1	2	3	4	5	6	7	8
0	2	3	5	6	15	–	–	–	–
1	–	2	4	6	10	12	–	–	–
2	–	–	2	3	7	8	10	–	–
3	–	–	–	2	4	7	8	15	–
4	–	–	–	–	2	3	6	12	15
5	–	–	–	–	–	2	5	10	12
6	–	–	–	–	–	–	2	7	10
7	–	–	–	–	–	–	–	2	6
8	–	–	–	–	–	–	–	–	2

Математическая модель

Эта задача является частным случаем задачи о ближайшем соседе, за исключением того, что в данной задаче количество сегментов разбиения трассы не задано.

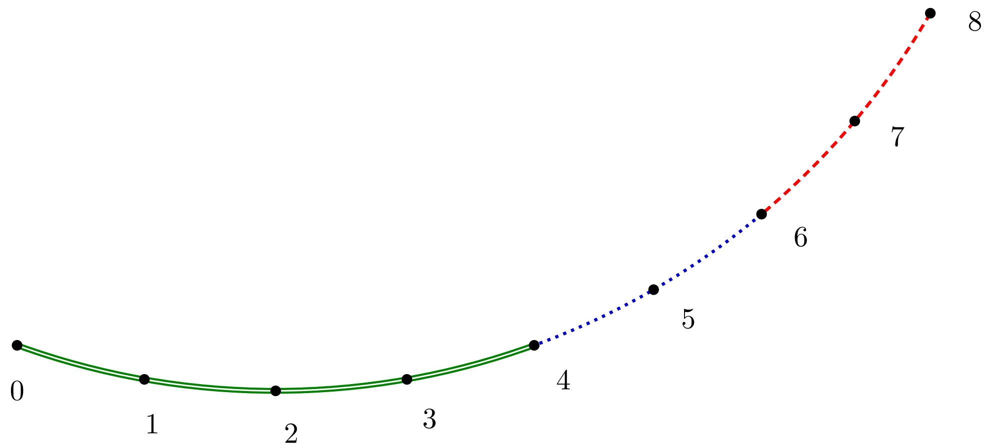


Рис. 2.6: Пример разбиения трассы на 3 сегмента $(0 \to 4)$, $(4 \to 6)$, $(6 \to 8)$.

Пусть $i = 1 \dots 8$ — точки, где можно ставить станции (в точке $i = 0$ станция устанавливается по умолчанию).

Параметры

- $f(x, y)$ — затраты на обслуживания участка $(x \to y)$.

Переменные

- $x = \{x_i\}$ — точки, где нужно строить станции.

Решение

Будем решать задачу с помощью динамического программирования, однако поскольку нам неизвестно количество шагов, для решения будем использовать многошаговый процесс принятия решений.

Исходные данные

- на каждом шаге будем определять в какой точке нужно ставить станцию обслуживания;

- <u>состояние</u> на каждом шаге (p) — точка начала участка, в конце которого будет установлена станция;
- <u>решение</u> на каждом шаге (q) — насколько нужно продвинуться вперёд от начала участка для установки станции;
- <u>множество возможных состояний</u>

$$P = \{0, 1, 2, \cdots, 8\},$$

на этом множестве установлен линейный строгий порядок, поскольку участки дороги следуют друг за другом;
- <u>функция перехода</u> — точка конца текущего и начала следующего участков

$$p' = T(p, q) = p + q,$$

функция перехода «согласована» с линейным строгим порядком на множестве P;
- <u>множества конечных состояний</u>;

$$\bar{P} = \{8\}, \quad \bar{P}_1 = \{7\}, \quad \bar{P}_2 = \{6\},$$

$$\bar{P}_3 = \{5\}, \quad \bar{P}_4 = \{4\}, \quad \bar{P}_5 = \{3\},$$

$$\bar{P}_6 = \{2\}, \quad \bar{P}_7 = \{1\}, \quad \bar{P}_8 = \{0\};$$

- в рамках задачи будет удобнее работать не с q, а с $p' = p + q$, в связи с этим определим <u>множество возможных значений для p'</u>

$$P' = \{1, 2, 3, 4, 5, 6, 7, 8\};$$

- <u>множества допустимых значений для p'</u>; учтём, что одна станция может обслуживать не более 4 точек дороги, а также что в одной точке дороги не может быть больше одной станции

$$P'(p) = \{p + 1, p + 2, \cdots, \min(p + 4, 8)\}.$$

Для решения данной задачи будем рассматривать различные подзадачи, в каждой из которых своё множество конечных состояний $\bar{P}_k$. В рамках данной задачи это будет означать следующее: мы будем рассматривать подзадачи, в которых нам доступна не вся дорога, а лишь её часть, которая начинается в точке $p \in \bar{P}_k$ и кончается в точке 8.

База процесса

Пусть $f(p)$ — это минимальные затраты на содержание дороги, если она начинается в точке p (в таком случае считается, что в точке p станция установлена), $f(0)$ — исходная задача.

По теории $f(8) = 0$ — база процесса, которую мы будем использовать для перехода.

> По таблице $f(8, 8) = 2$ — затраты на содержание двух станций, которые будут установлены в точке 8. В то время как $f(8)$ — содержание лишь одной станции, установленной в точке 8, поэтому $f(8) = 0 \neq f(8, 8) = 2$.

Переход

Пусть на некотором шаге у нас есть множество конечных состояний $\bar{P}_k$, тогда будем считать $f(p)$ для $p \in \bar{P}_k$ в предположении, что $f(p')$ мы знаем $\forall p' \in P'(p)$. Напишем рекуррентное соотношение для $f(p)$

$$\boxed{f(p) = \min_{p' \in P'(p)} \Big\{ f(p,p') + f(p') \Big\}}, \tag{*}$$

- $f(p,p')$ — затраты на первый сегмент дороги $(p \to p')$, при этом дорога начинается в точке p;
- $f(p')$ — минимальные затраты на содержание оставшейся части дороги $(p' \to 8)$; по предположению значение уже известно.

Исходя из множества возможных состояний P, составим таблицу.

p	(f, p')
0	
1	
2	
3	
4	
5	
6	
7	
8	

Искомое значение — это $\mathbf{f(p_0) = f(0)}$. Заполним таблицу, чтобы найти его, и определить точки где нужно ставить станции (оптимальное разбиение дороги).

Заполнение таблицы

$\boxed{\bar{P}}$ Рассматриваем первое множество конечных состояний $\bar{P} = \{8\}$, то есть нам доступна лишь части дороги $(8 \to 8)$, таким образом мы можем поставить станцию лишь в точке 8.

$$f(p) = f(8) = 0.$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	
4	
5	
6	
7	
8	$(0, 0)$

Значение p' в при $p = 8$ в таблице не имеет никакого значения, поэтому можно просто оставить его нулём.

$\boxed{\bar{P}_1}$ Рассматриваем второе множество конечных состояний $\bar{P}_1 = \{7\}$, то есть нам доступна лишь часть дороги $(7 \to 8)$. Выражение для этого и последующих множеств выглядит как рекуррентное соотношение (*)

$$f(p) = f(7) = \underbrace{8}_{p'} \;\Big|\; f(7,8) + f(8) = 6 + 0 = 6$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	
4	
5	
6	
7	$(6, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_2}$ Рассматриваем множество конечных состояний $\bar{P}_2 = \{6\}$, то есть нам доступна лишь часть дороги $(6 \to 8)$.

$$f(p) = f(6) = \begin{array}{c|l} 7 & f(6,7) + f(7) = 7 + 6 = 13 \\ \underbrace{8}_{p'} & f(6,8) + f(8) = 10 + 0 = 10 \end{array}$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	
4	
5	
6	$(10, 8)$
7	$(6, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_3}$ Рассматриваем множество конечных состояний $\bar{P}_3 = \{5\}$.

$$f(5) = \begin{array}{c|l} 6 & f(5,6)+f(6)=5+10=15 \\ 7 & f(5,7)+f(7)=10+6=16 \\ 8 & f(5,8)+f(8)=12+0=12 \end{array}$$

Занесём данные в таблицу.

p	(f,p')
0	
1	
2	
3	
4	
5	$(12,8)$
6	$(10,8)$
7	$(6,8)$
8	$(0,0)$

$\boxed{\bar{P}_4}$ Рассматриваем множество конечных состояний $\bar{P}_4 = \{4\}$.

$$f(4) = \begin{array}{c|l} 5 & f(4,5)+f(5)=3+12=15 \\ 6 & f(4,6)+f(6)=6+10=16 \\ 7 & f(4,7)+f(7)=12+6=18 \\ \underbrace{8}_{p'} & f(4,8)+f(8)=15+0=15 \end{array}$$

Занесём данные в таблицу. Учтём, что оптимальное значение целевой функции $f(p) = 15$ достигается при $p' = 5$ и $p' = 8$. В таблице это будет записано как $(15, 5/8)$.

p	(f,p')
0	
1	
2	
3	
4	$(15,5/8)$
5	$(12,8)$
6	$(10,8)$
7	$(6,8)$
8	$(0,0)$

$\boxed{\bar{P}_5}$ Рассматриваем множество конечных состояний $\bar{P}_5 = \{3\}$.

$$f(3) = \begin{array}{c|l} 4 & f(3,4)+f(4)=4+15=19 \\ 5 & f(3,5)+f(5)=7+12=19 \\ 6 & f(3,6)+f(6)=8+10=18 \\ \underbrace{7}_{p'} & f(3,7)+f(7)=15+6=21 \end{array}$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	$(18, 6)$
4	$(15, 5/8)$
5	$(12, 8)$
6	$(10, 8)$
7	$(6, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_6}$ Рассматриваем множество конечных состояний $\bar{P}_6 = \{2\}$.

$$f(2) = \underbrace{\begin{array}{c} 3 \\ 4 \\ 5 \\ 6 \end{array}}_{p'} \left| \begin{array}{l} f(2,3) + f(3) = 3 + 18 = 21 \\ f(2,4) + f(4) = 7 + 15 = 22 \\ f(2,5) + f(5) = 8 + 12 = 20 \\ f(2,6) + f(6) = 10 + 10 = 20 \end{array} \right.$$

Занесём данные в таблицу. Учтём, что оптимальное значение целевой функции $f(p) = 20$ достигается при $p' = 5$ и $p' = 6$. В таблице это будет записано как $(20, 5/6)$.

p	(f, p')
0	
1	
2	$(20, 5/6)$
3	$(18, 6)$
4	$(15, 5/8)$
5	$(12, 8)$
6	$(10, 8)$
7	$(6, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_7}$ Рассматриваем множество конечных состояний $\bar{P}_7 = \{1\}$.

$$f(1) = \underbrace{\begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \end{array}}_{p'} \left| \begin{array}{l} f(1,2) + f(2) = 4 + 20 = 24 \\ f(1,3) + f(3) = 6 + 18 = 24 \\ f(1,4) + f(4) = 10 + 15 = 31 \\ f(1,5) + f(5) = 12 + 12 = 24 \end{array} \right.$$

Занесём данные в таблицу. Учтём, что оптимальное значение целевой функции $f(p) = 24$ достигается при $p' = 2$, $p' = 3$ и $p' = 5$. В таблице это будет записано как $(24, 2/3/5)$.

p	(f, p')
0	
1	$(24, 2/3/5)$
2	$(20, 5/6)$
3	$(18, 6)$
4	$(15, 5/8)$
5	$(12, 8)$
6	$(10, 8)$
7	$(6, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_8}$ Рассматриваем множество конечных состояний $\bar{P}_8 = \{8\}$, оно содержит p_0, значит сейчас нам доступна вся дорога $(0 \to 8)$, таким образом это последнее множество конечных состояний, которое нам нужно рассмотреть.
значит оно последнее, которое нам нужно рассмотреть.

$$f(0) = \underbrace{\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array}}_{p'} \left| \begin{array}{l} f(0,1) + f(1) = 3 + 24 = 27 \\ f(0,2) + f(2) = 5 + 20 = 25 \\ f(0,3) + f(3) = 8 + 18 = 26 \\ f(0,4) + f(4) = 15 + 15 = 30 \end{array} \right.$$

Занесём данные в таблицу.

p	(f, p')
0	$(25, 2)$
1	$(24, 2/3/5)$
2	$(20, 5/6)$
3	$(18, 6)$
4	$(15, 5/8)$
5	$(12, 8)$
6	$(10, 8)$
7	$(6, 8)$
8	$(0, 0)$

Поиск оптимальной стратегии
Наше начальное состояние $p_0 = 0$, поскольку первый участок начинается в точке 0.

$$\begin{aligned} p_1^* &= p_0 = 0, & p_1^{'*} &= 2, \\ p_2^* &= p_1^{'*} = 2, & p_2^{'*} &= 5, \\ p_3^* &= p_2^{'*} = 5, & p_3^{'*} &= 8 \in \bar{P}. \end{aligned}$$

Итоговая стратегия — это $p^{'*} = \{2, 5, 8\}$, то есть станции нужно ставить в точках 2, 5 и 8, в таком случае затраты на содержание дороги составят $f(0) = 25$. Таким образом, в ходе решения задачи выяснилось, что для минимизации расходов нужно строить 3 станции.

$p_2'^*$ можно было выбрать равным 6, это бы привело к стратегии $p'^* = \{2, 6, 8\}$, но затраты остались бы такими же.

2.11 Задача о моторах

Задача (о моторах)

Имеется потребность в моторах 8 типов. Моторы i-го типа требуются в количестве a_i штук. Известно, что мотор i-го типа можно заменить мотором k-го типа, если $i < k$. Завод может производить не более трёх типов моторов. Стоимость производства одного мотора i-го типа равняется $30u_i$, а затраты на организацию производства моторов этого типа составляет $600 + 50u_i$. Какие типы моторов и в каких количествах следует производить на заводе, чтобы полностью удовлетворить потребность в моторах всех типов. Числовые значения величин a_i и u_i содержатся в таблице.

i	1	2	3	4	5	6	7	8
a_i	50	20	40	80	20	50	10	20
u_i	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1

Математическая модель

Сведём задачу к задаче о ближайшем соседе. Для этого нужно определить точки нашей трассы, количество точек разбиения и функцию расстояния. Пусть точки нашей трассы — это типы моторов, а функция расстояния — это затраты на производство моторов в нужных количествах.

В задаче о дороге мы разбивали трассу на участки, при этом в конце и в начале каждого участка устанавливалась станция. В данной задаче мы также будем разбивать дорогу на участки, но смысл будет следующий: если в разбиении дороги присутствует участок $(x \to y)$, то заводу нужно производить моторы типов x и y, а моторы промежуточных типов — не нужно (потребность в них будет закрыта моторами типа y). Таким образом, нужно найти разбиение трассы, состоящей из типов моторов, на участки.

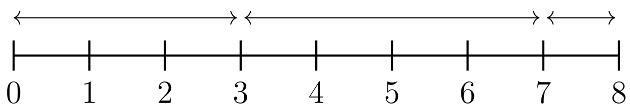


Рис. 2.7: Пример разбиение трассы в данной задаче на три участка: $(0 \to 3)$, $(3 \to 7)$, $(7 \to 8)$. Это означает, что заводу нужно производить моторы 3-го, 7-го и 8-го видов.

Итак, $\boxed{Z = \{0, 1, 2, \ldots, 7, 8\}}$ — множество точек трассы. Моторов 0-го типа нет, поэтому точка 0 в Z является фиктивной (смысл её добавления в трассу будет объяснён далее).

Теперь на Z нужно определить функцию расстояния $f(x, y)$. Эта функция будет иметь следующий смысл: это затраты на производство моторов типа y «в нужных количествах». Фраза «в нужных количествах» означает, что сюда вкладывается удовлетворение потребностей как непосредственно в моторах типа y, так и в моторах меньшего типа. Например, если у нас есть участок $(3 \to 8)$, то в $f(3,8)$ будут учтены затраты на производство моторов 8-го типа так, чтобы закрыть потребности в моторах 4-го, 5-го, 6-го, 7-го и 8-го типов. Запишем выражение для $f(x,y)$, исходя из условий задачи

$$\boxed{f(x,y) = 600 + 50 \cdot u_y + 30 \cdot u_y \cdot (a_{x+1} + a_{x+2} + \cdots + a_{y-1} + a_y)}, \quad x < y \in Z,$$

- $600 + 50 \cdot u_y$ — затраты на настройку оборудования для производства моторов типа y;
- $30 \cdot u_y \cdot (a_{x+1} + a_{x+2} + \cdots + a_{y-1} + a_y)$ — затраты на производство моторов типа y для закрытия потребностей в моторах типа $x+1$, $x+2$, $\ldots$, $y-1$, y.

Заметим, что если бы во множество точек трассы не входил ноль, то пришлось бы отдельно расписывать $f(1,y)$, поэтому добавление нуля во множество точек трассы позволяет обобщить выражение для $f(x,y)$ на все случаи. Можно считать, что моторы 0-го типа есть и по умолчанию производятся в нужных количествах, однако закрыть потребность в других с их помощью нельзя.

В условии задачи сказано, что завод может производить моторы не более трёх типов, то есть завод может производить моторы одного типа, двух и трёх. Если бы было сказано, что завод может производить ровно N типов моторов, то мы бы решали задачу с помощью N-шагового процесса принятия решений, однако поскольку этого не сказано, и оптимальная стратегия завода может включать производство моторов одного типа, двух и трёх, то будем решать задачу с помощью многошагового процесса принятия решений.

Решение

Исходные данные

- на каждом шаге будем определять моторы каких типов нужно производить, если уже в нужных количествах производятся моторы таких-то типов;
- состояние на каждом шаге (p) — точка начала участка (моторы в начале участка уже производятся в нужных количествах, а на производство моторов в конце участка нужно потратиться);
- решение на каждом шаге (q) — насколько нужно продвинуться вперёд от начала участка до его конца;
- множество возможных состояний

  $$P = \{0, 1, 2, \cdots, 8\},$$

  на этом множестве установлен линейный строгий порядок, поскольку участки дороги следуют друг за другом;
- функция перехода — точка конца текущего и начала следующего участков

  $$p' = T(p,q) = p + q,$$

  функция перехода «согласована» с линейным строгим порядком на множестве P;
- множества конечных состояний;

$$\bar{P} = \{8\}, \quad \bar{P}_1 = \{7\}, \quad \bar{P}_2 = \{6\},$$

$$\bar{P}_3 = \{5\}, \quad \bar{P}_4 = \{4\}, \quad \bar{P}_5 = \{3\},$$

$$\bar{P}_6 = \{2\}, \quad \bar{P}_7 = \{1\}, \quad \bar{P}_8 = \{0\};$$

- в рамках задачи будет удобнее работать не с q, а с $p' = p + q$, в связи с этим определим множество возможных значений для p'

$$P' = \{1, 2, 3, 4, 5, 6, 7, 8\};$$

- множества допустимых значений для p'

$$P'(p) = \{p + 1, p + 2, \cdots, 8\},$$

смысл этого множества следующий: моторы каких типов можно производить, если уже производятся моторы типа p.

Для решения данной задачи будем рассматривать различные подзадачи, в каждой из которых своё множество конечных состояний $\bar{P}_k$. В рамках данной задачи это будет означать следующее: мы будем рассматривать подзадачи, в которых у нас уже производятся моторы с типами $p \in \bar{P}_k$, а нам нужно определить, какие типы более мощных моторов нужно производить (чем больше тип мотора, тем он мощнее).

База процесса

Пусть $f(p)$ — минимальные затраты для закрытия потребностей во всех типах моторов, если завод может производить моторы типов $p + 1$, $p + 2$, ..., 8; $f(0)$ — ответ к исходной задаче.

По теории $f(8) = 0$ — база процесса, которую мы будем использовать для перехода.

Переход

Пусть на некотором шаге у нас есть множество конечных состояний $\bar{P}_k$, тогда будем считать $f(p)$ для $p \in \bar{P}_k$ в предположении, что $f(p')$ мы знаем $\forall p' \in P'(p)$. Напишем рекуррентное соотношение для $f(p)$

$$\boxed{f(p) = \min_{p' \in P'(p)} \Big\{ f(p, p') + f(p') \Big\}}, \qquad (*)$$

- $f(p, p')$ — затраты на производство моторов типа p';
- $f(p')$ — минимальные затраты на производство более мощных моторов, чем моторы типа p'; по предположению значение уже известно.

Исходя из множества возможных состояний P, составим таблицу.

p	(f, p')
0	
1	
2	
3	
4	
5	
6	
7	
8	

Искомое значение — это $\mathbf{f(p_0) = f(0)}$. Заполним таблицу, чтобы найти его, и определить типы моторов, которые нужно производить (оптимальное разбиение трассы).

Заполнение таблицы

$\boxed{\bar{P}_0}$ Рассматриваем первое множество конечных состояний $\bar{P} = \{8\}$, то есть моторы всех типов уже производятся, нам не нужно ничего производить.

$$f(p) = f(8) = 0.$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	
4	
5	
6	
7	
8	$(0, 0)$

> Значение p' в при $p = 8$ в таблице не имеет никакого значения, поэтому можно просто оставить его нулём.

$\boxed{\bar{P}_1}$ Рассматриваем второе множество конечных состояний $\bar{P}_1 = \{7\}$, то есть моторы с типами 1, 2, ..., 6, 7 уже производятся в нужных количествах, а мы можем добавить производство лишь моторов 8-го типа. Выражение для этого и последующих множеств выглядит как рекуррентное соотношение (*)

$$f(7) = \underbrace{8}_{p'} \left| \; f(7,8) + f(8) = 1250 + 0 = 1250 \right.$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	
4	
5	
6	
7	$(1250, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_2}$ Рассматриваем множество конечных состояний $\bar{P}_2 = \{6\}$, то есть моторы с типами 1, 2, ..., 5, 6 уже производятся в нужных количествах, а мы можем добавить производство лишь моторов 7-го и 8-го типов.

$$f(6) = \begin{array}{c|l} 7 & f(6,7) + f(7) = 915 + 1250 = 2165 \\ \underbrace{8}_{p'} & f(6,8) + f(8) = 1550 + 0 = 1550 \end{array}$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	
4	
5	
6	$(1550, 8)$
7	$(1250, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_3}$ Рассматриваем множество конечных состояний $\bar{P}_3 = \{5\}$.

$$f(5) = \begin{array}{c|l} 6 & f(5,6) + f(6) = 1840 + 1550 = 3390 \\ 7 & f(5,7) + f(7) = 2265 + 1250 = 3515 \\ \underbrace{8}_{p'} & f(5,8) + f(8) = 3050 + 0 = 3050 \end{array}$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	
4	
5	$(3050, 8)$
6	$(1550, 8)$
7	$(1250, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_4}$ Рассматриваем множество конечных состояний $\bar{P}_4 = \{4\}$.

$$f(4) = \begin{array}{c|l} 5 & f(4,5) + f(5) = 1055 + 3050 = 4105 \\ 6 & f(4,6) + f(6) = 2320 + 1550 = 3870 \\ 7 & f(4,7) + f(7) = 2805 + 1250 = 4055 \\ \underbrace{8}_{p'} & f(4,8) + f(8) = 3650 + 0 = 3650 \end{array}$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	
4	$(3650, 8)$
5	$(3050, 8)$
6	$(1550, 8)$
7	$(1250, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_5}$ Рассматриваем множество конечных состояний $\bar{P}_5 = \{3\}$.

$$f(3) = \left. \begin{array}{c} 4 \\ 5 \\ 6 \\ 7 \\ \underbrace{8}_{p'} \end{array} \right| \begin{array}{l} f(3,4) + f(4) = 2070 + 3650 = 5720 \\ f(3,5) + f(5) = 2735 + 3050 = 5785 \\ f(3,6) + f(6) = 4240 + 1550 = 5790 \\ f(3,7) + f(7) = 4265 + 1250 = 6215 \\ f(3,8) + f(8) = 6050 + 0 = 6050 \end{array}$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	
3	$(5720, 4)$
4	$(3650, 8)$
5	$(3050, 8)$
6	$(1550, 8)$
7	$(1250, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_6}$ Рассматриваем множество конечных состояний $\bar{P}_6 = \{2\}$.

$$f(2) = \left. \begin{array}{c} 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ \underbrace{8}_{p'} \end{array} \right| \begin{array}{l} f(2,3) + f(3) = 1225 + 5720 = 6945 \\ f(2,4) + f(4) = 2790 + 3650 = 6440 \\ f(2,5) + f(5) = 3575 + 3050 = 6625 \\ f(2,6) + f(6) = 5200 + 1550 = 6750 \\ f(2,7) + f(7) = 6045 + 1250 = 7295 \\ f(2,8) + f(8) = 7250 + 0 = 7250 \end{array}$$

Занесём данные в таблицу.

p	(f, p')
0	
1	
2	$(6440, 4)$
3	$(5720, 4)$
4	$(3650, 8)$
5	$(3050, 8)$
6	$(1550, 8)$
7	$(1250, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_7}$ Рассматриваем множество конечных состояний $\bar{P}_7 = \{1\}$.

$$f(1) = \begin{array}{c|l} 2 & f(1,2) + f(2) = 860 + 6440 = 7300 \\ 3 & f(2,3) + f(3) = 1525 + 5720 = 7245 \\ 4 & f(2,4) + f(4) = 3150 + 3650 = 6800 \\ 5 & f(2,5) + f(5) = 3995 + 3050 = 7045 \\ 6 & f(2,6) + f(6) = 5680 + 1550 = 7230 \\ 7 & f(2,7) + f(7) = 6585 + 1250 = 7835 \\ \underbrace{8}_{p'} & f(1,8) + f(8) = 7850 + 0 = 7850 \end{array}$$

Занесём данные в таблицу.

p	(f, p')
0	
1	$(6800, 4)$
2	$(6440, 4)$
3	$(5720, 4)$
4	$(3650, 8)$
5	$(3050, 8)$
6	$(1550, 8)$
7	$(1250, 8)$
8	$(0, 0)$

$\boxed{\bar{P}_8}$ Рассматриваем множество конечных состояний $\bar{P}_8 = \{0\}$. На данном шаге у нас в нужных количествах уже производятся лишь моторы 0-го (фиктивного типа), поэтому нам нужно закрыть потребность в моторах всех типов.

$$
f(0) = \begin{array}{c|l}
1 & f(0,1)+f(1) = 1065+6800 = 7865 \\
2 & f(0,2)+f(2) = 1460+6440 = 7900 \\
3 & f(0,3)+f(3) = 2275+5720 = 7995 \\
4 & f(0,4)+f(4) = 4050+3650 = 7700 \\
5 & f(0,5)+f(5) = 5045+3050 = 8095 \\
6 & f(0,6)+f(6) = 6800+1550 = 8450 \\
7 & f(0,7)+f(7) = 7935+1250 = 9185 \\
\underbrace{8}_{p'} & f(0,8)+f(8) = 9350+0 = 9350
\end{array}
$$

Занесём данные в таблицу.

p	(f, p')
0	$(7700, 4)$
1	$(6800, 4)$
2	$(6440, 4)$
3	$(5720, 4)$
4	$(3650, 8)$
5	$(3050, 8)$
6	$(1550, 8)$
7	$(1250, 8)$
8	$(0, 0)$

Поиск оптимальной стратегии

Наше начальное состояние $p_0 = 0$, поскольку заводу для производства доступны моторы всех видов.

$$
\begin{aligned}
p_1^* &= p_0 = 0, & p_1^{'*} &= 4, \\
p_2^* &= p_1^{'*} = 4, & p_2^{'*} &= 8 \in \bar{P}.
\end{aligned}
$$

Полученная стратегия — это $p^{'*} = \{4, 8\}$, то есть нужно производить моторы лишь 4-го и 8-го типов. Заметим, что это удовлетворяет условию задачи про то, что завод может производить не более трёх типов моторов. Осталось выяснить, сколько моторов каждого типа нужно производить.

Ясно, что для получения минимальных затрат нужно произвести столько моторов 4-го типа, чтобы закрыть ими потребности в моторах 1-го, 2-го, 3-го и 4-го типов, а потребность в моторах остальных типов закрыть моторами 8-го типа. Чтобы с помощью моторов 4-го типа закрыть потребность в моторах 1-го, 2-го, 3-го и 4-го типов, нужно производить их в количестве $a_1 + a_2 + a_3 + a_4 = 190$. Поскольку потребность в моторах от 1-го до 4-го типов включительно закрыта, осталось закрыть потребность в моторах 5-го, 6-го, 7-го и 8-го видов с помощью моторов 8-го вида. Для этого нужно производить моторы 8-го вида в количестве $a_5 + a_6 + a_7 + a_8 = 100$.

Таким образом, итоговая стратегия звучит так: нужно производить 190 моторов 4-го типа и 100 моторов 8-го типа, в таком случае затраты составят 7700.

Если бы полученная стратегия не удовлетворяла условию о том, что завод может производить не более трёх типов моторов, то нужно было бы решить задачу с условием «завод может производить ровно один тип моторов», «ровно два типа моторов» и «ровно три типа моторов». То есть фактически нужно было бы решить три задачи с помощью N-шагового процесса принятия решений: $N = 1$, $N = 2$ и $N = 3$ соответственно. Далее нужно было бы выбрать из полученных трёх стратегий самую оптимальную, это и было бы решением исходной задачи с условием «не более трёх типов моторов».

2.12 Модель замены оборудования

Определение (модель замены оборудования)

Исходные данные

Модель замены оборудования обобщает задачи, когда в распоряжении имеется определенное оборудование, которым необходимо пользоваться на протяжении N интервалов (дней, неделей, месяцев и тому подобному).

В начале каждого интервала нам нужно принять одно из возможных решений по оперированию с оборудованием:

- полностью его заменить (купить новое, старое можно продать),
- отремонтировать,
- ничего не делать.

Также оборудование необходимо обслуживать. Стоимость обслуживания зависит от «возраста» оборудования (времени, прошедшего после покупки оборудования), а также срока последнего ремонта (времени, прошедшего с последнего ремонта). Таким образом, состояние оборудования p определяется парой (t, r), то есть «возрастом» и временем, прошедшем с последнего ремонта.

Множество допустимых решений

Определим множество допустимых решений как $Q = \{0, 1, 2\}$, где

$$q = \begin{cases} 0, & \text{покупка нового и продажа старого,} \\ 1, & \text{ремонт,} \\ 2, & \text{ничего не делать.} \end{cases}$$

Функция перехода

Определим функцию перехода для данной модели. Обратим внимание, что операции происходят в начале интервала, поэтому

- в случае ремонта r становится равным 0,
- в случае покупки нового t становится равным 0.

Далее оборудование (будь то старое или новое) используется целый интервал, поэтому к началу следующего интервала оба показателя увеличиваются на 1.

В итоге, функция перехода имеет следующий вид

$$T_i(t,r,q) = \begin{cases} (1,1), & q = 0, \\ (t+1,1), & q = 1, \\ (t+1,r+1), & q = 2. \end{cases}$$

Параметры

Определим следующие параметры:

- C_i^0 – стоимость покупки нового оборудование на интервале i;
- $C_i^0(t)$ – стоимость продажи оборудования возраста t на интервале i;
- $C_i^1(t,r)$ – стоимость ремонта оборудования возраста t, который ремонтировался r шагов назад, на интервале i;
- $C_i^э(t,r)$ – затраты на эксплуатацию оборудования возраста t, который ремонтировался r шагов назад, на интервале i.

Функция затрат

Определим функцию затрат для интервала i:

$$g_i(t,r,q) = \begin{cases} C_i^0 - C_i^0(t) + C_i^э(0,0), & q = 0, \\ C_i^1(t,r) + C_i^э(t,0), & q = 1, \\ C_i^э(t,r), & q = 2. \end{cases}$$

Выбор оптимальной стратегии

<u>База процесса</u>

Пусть $f_n(t,r)$ — минимальные затраты на обеспечение работы оборудования (покупка, обслуживание, продажа) от интервала n до интервала N, при условии что на начало интервала n возраст оборудования равен t и последний раз оборудование ремонтировалось r интервалов назад.

Определим $f_N(t,r)$ для последнего интервала N:

$$\boxed{f_N(t,r) = \min_{q \in Q} \Big\{ g_N(t,r,q) \Big\}}$$

Можно так же рассмотреть случай когда после интервала N мы продадим оборудование. Для этого определим функцию

$$h(t,q) = \begin{cases} C_{N+1}^0(1) & q = 0, \\ C_{N+1}^0(t+1), & q \neq 0. \end{cases}$$

Тогда $f_N(t,r)$ будет иметь вид

$$\boxed{f_N(t,r) = \min_{q \in Q} \Big\{ g_N(t,r,q) - h(t,q) \Big\}}$$

<u>Переход</u>

Теперь, когда у нас есть база для решения, можно осуществить переход $f_{n+1}(t,r) \to f_n(t,r)$

$$\boxed{f_n(t,r) = \min_{q\in Q}\Big\{g_N(t,r,q) + f_{n+1}\big(T_n(t,r,q)\big)\Big\}}$$

Используя эти данные, можно решать задачи подобной модели с помощью N-шагового процесса принятия решений. Однако у данной модели есть существенный недостаток. Ввиду того, что фактически состояние p представляет собой пару (t,r), решение задачи займёт существенно больше времени, поскольку будет много возможных значений для $p=(t,r)$. В связи с этим, рассмотрим упрощённую версию этой модели.

2.12.1 Модель замены оборудования без учёта ремонта

Определение (модель замены оборудования без учёта ремонта)

В модели замены оборудования мы рассматривали ремонт как один из вариантов того, что мы можем сделать с оборудованием в начале интервала. В текущей модели предлагается убрать из рассмотрения этот вариант, иначе говоря, мы можем принять лишь одно из двух решений: купить новое оборудование или ничего не делать.

В таком сценарии больше не нужно учитывать вариант с ремонтом оборудования и не нужно следить за тем, сколько прошло времени с момента последнего ремонта. Таким образом, теперь в состоянии нет r, то есть $p=t$. Это существенно упрощает решение задачи.

Подход к решению

Можно заметить, что задача по сути свелась к тому, что нам нужно указать те точки (начала интервалов), где нужно покупать новое оборудования так, чтобы в итоге мы затратили как можно меньше денег. Это похоже на уже решённую задачу о дорогах. Чтобы воспользоваться решением, нужно всего лишь определить функцию расстояния $f(x,y)$, поскольку ясно, что множество точек дороги $Z=\{1,2,\ldots,N\}$.

Параметры

Оставим из модели замены оборудования следующие параметры:

- C_i^0 – стоимость покупки нового оборудование на интервале i;
- $C_i^{э}(t)$ – затраты на эксплуатацию оборудования возраста t на интервале i.

Функция расстояния

$f(x,y)$ — функция расстояния, она определяет сколько будет потрачено денег при покупке оборудования на начале интервала x и эксплуатацию на интервалах от x до y. Заметим, что во время эксплуатации оборудования на интервале x его возраст будет 0, на интервале $x+1$ уже 1 и так далее, а так же то, что не нужно учитывать эксплуатацию на интервале y, поскольку эксплуатация идет именно <u>до</u> интервала y. Тогда функция будет определяться как

$$\boxed{f(x,y) = C_x^0 + \sum_{i=x}^{y-1} C_i^{э}(i-x)} \qquad (*)$$

Допустим, изначально у нас уже было оборудование возраста t. Для удобства введём следующие функции

$$f_1(x,y) \overset{*}{=} C_x^0 + \sum_{i=x}^{y-1} C_i^{э}(i-x),$$

$$f_2(y) = \sum_{i=1}^{y-1} C_i^{э}(t+i-1),$$

- $f_1(x, y)$ обозначает затраты на эксплуатацию оборудования от интервала x до интервала y, если в начале интервала x было куплено новое оборудование;
- $f_2(y)$ обозначает затраты на эксплуатацию уже имеющегося оборудования возраста t от интервала 1 до интервала y.

Напишем выражение для $f(x, y)$ с учётом введённых функций

$$\boxed{f(x,y) = \begin{cases} f_1(x,y) & x \neq 1, \\ \min\left\{f_1(1,y), f_2(y)\right\}, & x = 1. \end{cases}}$$

Если $x = 1$ мы рассматриваем сразу 2 случая: либо в начале самого первого интервала продаем имеющееся оборудование возраста t, покупаем новое и далее эксплуатируем новое, либо оставляем имеющееся оборудование и эксплуатируем его (заранее неизвестно, что будет выгоднее по деньгам).

Используя эти данные, задачу можно решить как задачу о дорогах.

3 Сетевые модели

3.1 Теория графов

Определение

Графом называется пара множеств $G = (V, E)$, где

- V — *множество вершин*,
- $E \subseteq V \times V$ — *множество рёбер*.

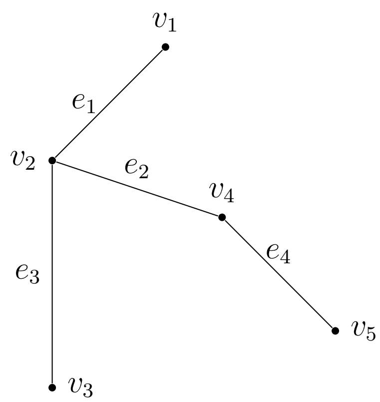


Рис. 3.1: Пример графа

Определение

Вершины называются *смежными*, если они соединены ребром.

Определение

Ориентированным графом называется пара множеств $G = (V, E)$, где

- V — *множество вершин*,
- $E \subseteq V \times V$ — *множество дуг*.

Будем обозначать дугу от вершины x до вершины y как $\vec{xy}$.

Замечание

В неориентированном графе если множество рёбер E содержит ребро (v_i, v_j), то оно также содержит ребро (v_j, v_i). Гарантировать то же самое в ориентированном графе нельзя.

Замечание

В дальнейшем для простоты будем считать, что

- $V = \{1, 2, \ldots, n\}$,
- $E = \{1, 2, \ldots, m\}$.

Определение

Пусть каждая дуга j ориентированного графа имеет некоторый вес $c(j)$, тогда *табличным видом ориентированного графа* будем называть следующую таблицу

j	$i(j)$	$k(j)$	$c(j)$
1			
1			
$\dots$			
m			

- j — дуга,
- $i(j)$ — начальная точка дуги,
- $k(j)$ — конечная точка дуги,
- $c(j)$ — вес дуги.

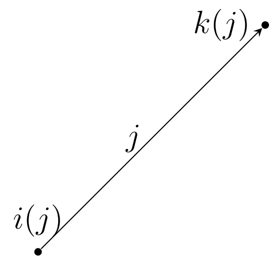


Рис. 3.2: Пример дуги j

Определение

Путь в графе от вершины i_1 до i_{L+1} — последовательность неповторяющихся вершин. Будем обозначать это как $P(i_1, \dots, i_{L+1})$ или $\{i_1, \dots, i_{L+1}\}$, где $(i_l, i_{l+1}) \in E$.

Замечание

В рамках данного курса не будут рассматриваться пути, являющиеся циклами, то есть всегда $i_1 \neq i_k$.

Определение

Сеть — ориентированных граф $G = (I, J)$, в котором

- есть 2 выделенные вершины s (источник/вход) и t (сток/выход);
- на множествах вершин и рёбер определён строгий порядок.

Вершины s и t определяется так: s — наименьшая вершина, в которую не входит дуга; а t — наибольшая вершина, из которой не выходит дуга.

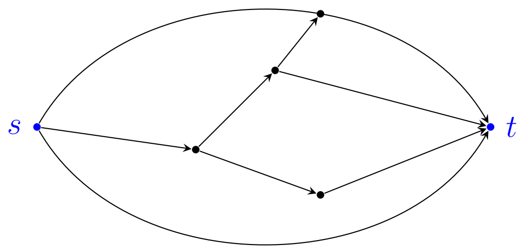


Рис. 3.3: Пример сети

Определение

Сеть называется *взвешенной*, если у каждой дуги j есть некоторой вес τ_j. Тогда такую сеть можно задать с помощью следующей таблицы

j	$i(j)$	$k(j)$	$c(j)$	τ_j
1				
1				
$\ldots$				
m				

3.2 Сетевая модель проекта

Определение

Проект — совокупность работ для достижения определённой цели. *Модель проекта* определяется следующим образом:

- $J = \{1, \ldots, n\}$ — множество работ проекта,
- $\tau_j \geq 0$ — длительность работы j;
- $C = \{(i, j) \mid i, j \in J\}$ — частичный порядок, работа j не может начаться раньше окончанию работы i.

Модель проекта можно записать с помощью следующей таблицы

j	следующие работы	τ_j
j_0	j_1, j_2, $\ldots$	τ_{j_0}
$\ldots$	$\ldots$	$\ldots$

При этом, «следующие работы» для работы j — это работы, которые зависят от j, то есть которые не могут начаться раньше, чем завершится работа j.

Пример

Рассмотрим следующую модель проекта:

- $J = \{1, 2, \ldots, 9, 10\}$ — множество работ проекта,
- 1 — выбрать место для офиса,
- 2 — создать финансовый и организационный план,
- 3 — определить обязанности персонала,

- 4 — разработать план офиса,
- 5 — ремонт помещений,
- 6 — отобрать кандидатов на увольнение,
- 7 — нанять новых служащих,
- 8 — назначить ключевых руководителей,
- 9 — распределить обязанности руководителей,
- 10 — обучить персонал;

j	следующие работы	τ_j
1	$\{4\}$	3
2	$\{3, 9\}$	5
3	$\{4, 6\}$	3
4	$\{5\}$	5
5	$\{10\}$	10
6	$\{7, 8\}$	2
7	$\{10\}$	5
8	$\{10\}$	2
9	$\varnothing$	5
10	$\varnothing$	3

Определение

Сетевая модель проекта — наглядное представление работ в виде сети, в которой для каждой работы есть дуга, а отношение следования работ в проекте имеет место и в сети.

Сетевая модель проекта — ориентированный взвешенный граф без циклов $G = (V, E)$ с выделенными вершинами s и t, при этом каждой дуге $j = \vec{ik}$ приписан вес $\tau_j \geq 0$. Вершинами сети будут *события* (результаты выполнения работ), а дугами сами работы.

Обозначение: ① — вершина с номером 1, $\boxed{1}$ — дуга с номером 1.

Пример

Изобразим сетевую модель проекта из примера.

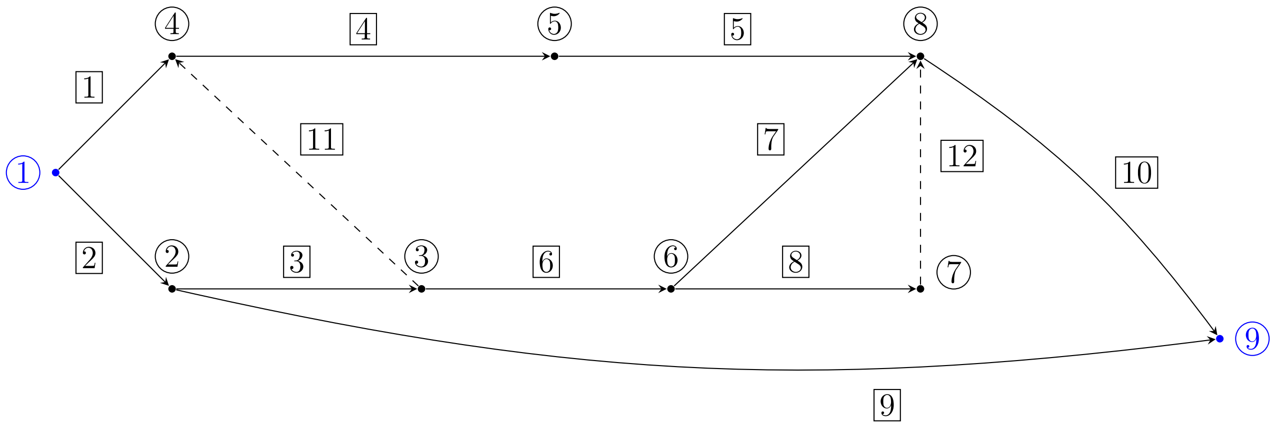


Рис. 3.4: Сеть для проекта из примера. Дуги 11 и 12 является фиктивными, то есть не соответствуют работам из модели проекта, они нужны для целостности сети.

Алгоритм (построения сетевой модели)

1. Для каждой работы строим вершины и дуги.
2. Обозначаем подчинённость, рисуя фиктивные дуги из конца текущей работы в следующую в соответствии с проектом.
3. Добавляем начальную и конечную вершины s и t.
4. Убираем фиктивные дуги так, чтобы граф не перестал быть сетью, а также чтобы не появились параллельные дуги. Фиктивную дугу можно убрать, если она единственная, которая входит в вершину или выходит из вершины.
5. Пронумеровываем вершины сети так, чтобы для каждой дуги j выполнялось $i(j) < k(j)$.

3.2.1 Алгоритм Форда

Определение

Рангом $r(x)$ вершины $x \in V$ называется число дуг в максимальном (по числу дуг) пути из начальной вершины s в вершину x. *Рангом проекта* называется $r(t)$ (ранг стока/выхода/конечной вершины).

Алгоритм (Форда)

Алгоритм позволяет устанавливать ранги в сети так, чтобы нумерация вершин оказалось корректной с точки зрения условия $i(j) < k(j)$.

Суть алгоритма в следующем:

- в каждый момент времени будем для каждой вершины k хранить её текущий ранг r_k (изначально все $r_k = 0$);
- на каждой итерации будем последовательно просматривать все работы из множества J от наименьшей к наибольшей и менять ранг вершин следующим образом: пусть текущая работа — это $j = (i, k)$, тогда

$$r_k = \max\{r_k, r_i + 1\};$$

- алгоритм будет повторяться до тех пор, пока ранги вершин не перестанут меняться.

Для удобства ранги вершин можно записывать в таблицу

k	r_k^0	r_k^1	r_k^2	$\dots$
1	0	$\dots$	$\dots$	$\dots$
2	0	$\dots$	$\dots$	$\dots$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
n	0	$\dots$	$\dots$	$\dots$

где r_k^i — ранг вершины k на шаге i.

Замечание

Значение r_k для одной и той же вершины k может меняться несколько раз в течение одного шага.

Замечание

Если сеть относительно небольшая, то быстрее будет «вручную» пронумеровать вершины, а не пользоваться алгоритмом Форда.

Утверждение

Алгоритм Форда создаёт корректную нумерацию вершин в сети.

3.3 Поиск максимальных путей

Определение

Длиной пути $P(i_1, \dots, i_{L+1})$ называется

$$|P(i_1, \dots, i_{L+1})| = \sum_{l=1}^{L} \tau_{(i_l, i_{l+1})}$$

Замечание

Вместо суммы может стоять произведение, взятие максимума/минимума и так далее (всё зависит от конкретной задачи и сети).

Задача (поиска максимальных путей)

Есть некоторая сеть $G = (I, J)$, нужно найти все пути максимальной длины от первой до последней вершины $(1 \to n)$.

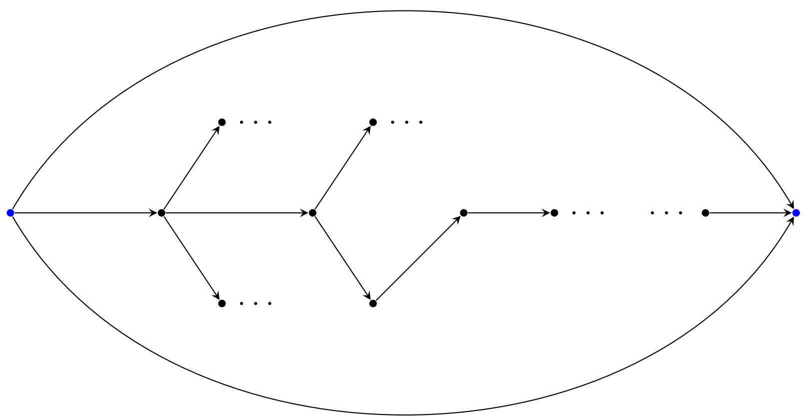


Рис. 3.5: Задача поиска максимальных путей в сети

Решение

Для решения задачи будем использовать динамическое программирование. Поскольку нам не задано количество шагов (длины путей зависят от конкретной сети), то будем использовать многошаговый процесс принятия решений. Вспомним, что по определению сети на множестве вершин задан строгий частичный порядок — это соответствует требования многошагового процесса («согласованность» функции перехода).

Исходные данные

Формализуем задачу поиска максимальных путей в терминах динамического программирования:

- <u>состояние</u>: $i \in I$ (текущая вершина);
- <u>решение</u>: $j \in J$ (дуга, по которой мы пойдём);
- <u>множество возможных состояний</u>: $P = I$;
- <u>множество конечных состояний</u>: $\bar{P} = \{n\}$;
- <u>начальное состояние</u>: $p_0 = 1$;
- <u>множество возможных решений</u>: $Q = J$;
- <u>множество допустимых решений</u>: $Q(i) = \{j \mid i(j) = i\}$ (все дуги, по которым можно пойти из текущей вершины);
- <u>функция перехода</u>: $T(i, j) = k(j)$ (находимся в вершине i и двигаемся по дуге j);
- <u>функция дохода</u>: $g(i, j) = \tau_j$.

База процесса

Пусть $f(i)$ — длина максимального пути из вершины i в вершину n, $f(1)$ — ответ на вопрос исходной задачи. Так как длину пути можно представить как сумму длин входящих в него путей, по теории $\boxed{f(n) = 0}$.

Переход

Пусть мы находимся в вершине i, тогда нам нужно рассмотреть все дуги, начинающиеся в данной вершине.

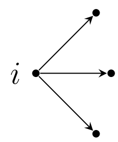


Рис. 3.6: Пример дуг, начинающихся в вершине i

Таким образом

$$\boxed{f(i) = \max_{j:\, i(j)=i} \Big\{\tau_j + f\big(k(j)\big)\Big\}}. \qquad (*)$$

Так как $k(j) > 1$, то значение $f\big(k(j)\big)$ нам известно.

Выбор оптимальной стратегии

Будем записывать все данные в следующую таблицу

i	$f(i)$	$j(i)$
1		
2		
$\dots$		
$n-1$		
n		

$j(i)$ — дуга, на которой достигается максимум в (*).

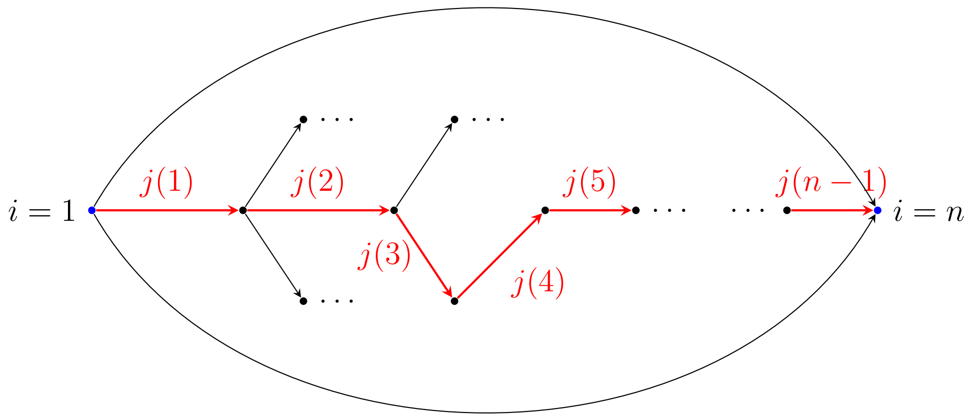


Рис. 3.7: Пример оптимальной стратегии

Замечание

По ходу процесса, мы узнаем максимальные пути из каждой вершины в n-ую. Однако на будущее хотелось бы знать про максимальные пути из 1-ой вершины в любую. Для этого можно проделать тот же самый процесс, но идти не $1 \to n$, а $n \to 1$.

База процесса

Пусть $g(i)$ — длина максимального пути из вершины i в вершину 1. По теории $\boxed{g(1) = 0}$.

Переход

$$\boxed{g(i) = \max_{j' \colon k(j') = i} \Big\{ \tau_{j'} + g\big(i(j')\big) \Big\}}.$$

Так как $i(j') < i$, то значение $g\big(i(j')\big)$ нам известно.

Выбор оптимальной стратегии

Итоговая таблица имеет следующий вид

i	$f(i)$	$j(i)$	$g(i)$	$j'(i)$
1				
2				
$\dots$				
$n-1$				
n				

Замечание

$f(1) = g(n)$.

Замечание

Пусть исходная сеть соответствовала некоторому проекту с набором работ J и продолжительностями работ $\{\tau_j\}$. Проект будет реализован, если будут реализованы самые длинны пути $1 \to n$.

Обозначение

- $T_i^{\text{р}}$ (*самое ранее время наступления события* i). Это равняется длине максимального пути из 1-ой вершины в i-ую.
- $T = T_n^{\text{р}} = g(n) = f(1)$ (*критическое время проекта*).
- $T_i^{\text{П}} = T - f(i)$ (*наиболее позднее время наступления события* i).

Определение

Путь будем называть *критическим*, если его длина равняется T.

Определение

Работу будем называться *критической*, если дуга, соответствующая этой работе, входит в состав критического пути.

Замечание

Если увеличить время критической работы, то время проекта увеличится.

Замечание

Мы уже говорили, что если работа критическая, то увеличивать её время нежелательно, так как увеличится время всего проекта. А если работа не критическая, то можно ли увеличить её время? И если можно, то насколько?

Утверждение

Пусть Δ_j — увеличение времени работы j, тогда максимальное значение Δ_j, которое не приведёт к увеличению времени проекта, вычисляется по формуле

$$\Delta_j^{\max} = T_{k(j)}^{\text{П}} - T_{i(j)}^{\text{р}} - \tau_j.$$

Доказательство

Заметим, что

$$g\big(i(j)\big) + \tau_j + f\big(k(j)\big) \leq T. \qquad (*)$$

Действительно, $g\big(i(j)\big)$ — максимальное время выполнения всех работ до работы j, а $f\big(k(j)\big)$ — после работы j, поэтому выражение слева уж никак не может быть больше времени всего проекта T.

Из (*) следует, что к левой части можно добавить $\Delta_j \geq 0$ так, чтобы неравенство всё ещё выполнялось. Максимальное Δ_j, которое можно прибавить, это значение, при котором будет достигаться равенство, то есть

$$g\big(i(j)\big) + \tau_j + \Delta_j^{\max} + f\big(k(j)\big) = T,$$

$$\Downarrow$$

$$\begin{aligned}\Delta_j^{\max} &= T - f\big(k(j)\big) - g\big(i(j)\big) - \tau_j \\ &= T_{k(j)}^{\text{П}} - T_{i(j)}^{\text{р}} - \tau_j.\end{aligned}$$

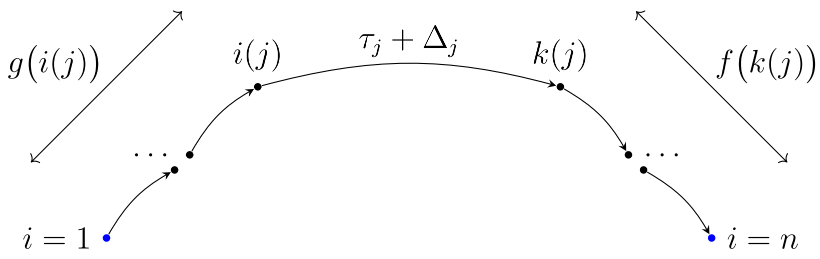


Рис. 3.8: Увеличение времени работы в сети

Следствие

- Если $\Delta_j > \Delta_j^{\max}$, то время проекта увеличится;
- Если $\Delta_j \leq \Delta_j^{\max}$, то время проекта не изменится.

3.4 Задача о максимальном потоке

Определение

Пусть

- $G = (I, J)$ — взвешенная сеть;
- $i = 1$ — вход, $i = n$ — выход;
- b_{ik} — вес дуги $\vec{ik}$;

тогда $x = (x_{ik})$ будем называть *потоком в сети* с величиной входного потока v, если выполняется следующие ограничения

$$\forall l \in I \quad \sum_{k\colon (l,k) \in J} x_{lk} - \sum_{i\colon (i,l) \in J} x_{il} = \begin{cases} v, & l = 1, \\ 0, & l = 2 \dots n-1, \\ -v, & l = n; \end{cases}$$

(закон сохранения энергии)

$$\forall j = (i, k) \quad x_{ik} \leq b_{ik}.$$

Определение

Величину потока x в сети будем обозначать $v(x)$.

Определение

Пару $(S, \bar{S})$ будем называть *разрезом сети*, если S и $\bar{S}$ разбивают множество вершин сети, при этом обязательно $1 \in S$, $n \in \bar{S}$.

Определение

Пропускной способностью разреза будем называть

$$B(S, \bar{S}) = \sum_{\substack{(i,k) \in J \\ i \in S, k \in \bar{S}}} b_{ik}$$

По сути, это сумма пропускным способностей всех дуг, идущих из S в $\bar{S}$.

Утверждение (свойство разреза)

Какой бы ни был поток $x = (x_{ik})$ и разрез $(S, \bar{S})$

$$v(x) \leq B(S, \bar{S}).$$

Доказательство

Напишем следующие выражение

$$Z = \sum_{l \in S} \Bigg(\sum_{k\colon (l,k) \in J} x_{lk} - \sum_{i\colon (i,l) \in J} x_{il} \Bigg).$$

Вспомним, что по определению разреза $1 \in S$, а значит $Z = v + \ldots$. Также вспомним определение потока и условия оттуда, из них следует, что внутреннее выражение для всех $l \neq 1$ равняется 0. Из всего этого следует, что

$$Z = \underbrace{0}_{v=1} + \underbrace{0 + \cdots + 0}_{1 \neq l \in S} = v$$

Посмотрим на Z с другой стороны. Рассмотрим возможные случаи взаимного расположения вершин i и k для вершины l

- $i \in S, k \in S$;
- $i \in S, k \in \bar{S}$;
- $k \in S, k \in S$.

$$\begin{aligned}
Z &= \sum_{l \in S} \Bigg(\sum_{k\colon (l,k) \in J} x_{lk} - \sum_{i\colon (i,l) \in J} x_{il} \Bigg) \\
&= \sum_{\substack{(i,k) \in J \\ i \in S, k \in \bar{S}}} x_{ik} - \sum_{\substack{(i,k) \in J \\ i \in \bar{S}, k \in S}} x_{ik} \\
&\leq \sum_{\substack{(i,k) \in J \\ i \in S, k \in \bar{S}}} x_{ik} \\
&\leq \sum_{\substack{(i,k) \in J \\ i \in S, k \in \bar{S}}} b_{ik} \\
&= B(S, \bar{S}).
\end{aligned}$$

Для перехода $\sum_{\ldots} x_{ik} \leq \sum_{\ldots} b_{ik}$ использовалось условие из определения потока в сети.

Таким образом $Z = v$ и $Z \leq B(S, \bar{S})$, значит $v \leq B(S, \bar{S})$.

TODO: есть путаница, где-то просто v, а где-то $v(x)$.

Утверждение (критерий максимальности потока)