Математические модели и методы принятия оптимальных решений

Береснев Владимир Леонидович

ФИТ НГУ

16 сентября 2025 г.

Оглавление

Глава 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.

Примерный алгоритм построения математических моделей

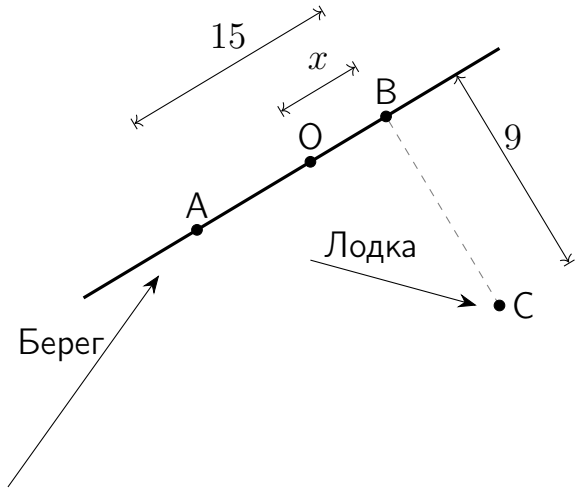
1. Нужно понять, *что будем оптимизировать?* По каким критериям будет оценивать решения? Например, если нас спрашивают, где построить магазин, то нам

нужно понять, как выбрать для этого место. Нужно, чтобы была максимальная прибыль или чтобы 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, значит нужно вернуться к самому началу...

Пример

Ваш друг плывёт на лодке, ему нужно попасть в определённую точку на берегу, и 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;
- $OB = x$;
- течения воды нет, то есть скорость течения равна нулю;
- цвет лодки не существенен.

Будем оптимизировать время t

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

Установим **ограничение**

$$0 \le x \le 15,$$

потому что иначе решение будет точно неоптимальным.

Решим задачу, найдя 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 $x_1 = -12$ не подходит ввиду ограничения выше, а вот $\boxed{x = 12}$ является ответом (упражнение).

1.2 Оптимизационные (экстремальные) задачи

Определение (экстремальная задача)

Экстремальная задача формулируется следующим образом

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

при этом *нужно найти*

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

Определение

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

Определение

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

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

или то же самое

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

Замечание

Ограничения $\{g_i\}$ могут быть какими угодно.

Пример

Пусть наши исходные данные это

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

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

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

Нужно найти x_1 и x_2.

Определение

Будем говорить, что есть $\mathcal{P}$ — *общая задача*, а $p \in \mathcal{P}$ — конкретная задача, в которой c_1, c_2, a, b, d — это конкретные числа. 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

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 будем оценивать трудоёмкость так

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

Определение

Если $f_A\big(|p|\big) = C \cdot |p|^k$, то такой алгоритм будем называть *«хорошим»* (*полиномиальным*).

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

Определение (задача линейного программирования)

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

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

$x_j \in \{0, 1, 2, \ldots\}$ или то же самое $x_j \geq 0, \quad j = 1 \ldots n.$

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

1.4 Свойства оптимизационных задач

Утверждение 1

$$\max_x f(x) = \max_x \Big(\big(-1\big) \cdot \big(-f(x)\big) \Big) = \underbrace{-\min_x \big(-f(x)\big)}_{\text{новая задача}}.$$

Утверждение 2 (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

$$\forall p \in \mathcal{P} \quad \forall q \in \mathcal{Q}$$

1. существует полиномиальный алгоритм A_1, который переводит входные данные задачи p во входные данные задачи q;

$$p \qquad \xrightarrow{A_1} \qquad q$$

2. существует полиномиальный алгоритм A_2, с помощью которого можно из оптимального решения y^0 задачи q построить оптимальное решение x^* задачи p.

$$x^* \qquad \xleftarrow{A_2} \qquad y^0$$

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

Замечание

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

Утверждение 3 (сведение к другой задаче)

Пусть есть 2 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 пункту для x^* существует некоторый y^0 такой, что

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

По условию y^* является оптимальным решением 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 $x^0 = x^*$, то есть x^0 является оптимальным.

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

$(\mathcal{Q})$

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

.

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

.

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

$$y_2 \leq d.$$

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

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

   (a) Проверим ограничение 2

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

   (b) Проверим ограничение 1

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

   Значит x^0 удовлетворяет всем ограничениям, а поэтому x^0 — допустимое решение.

2. Мы показали, что x^0 — допустимое решение задачи $\mathcal{P}$. Осталось показать, что оно оптимальное. Для этого будем использовать утверждение 3.

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

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

(b) Выполнимость условия

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

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 равенство

$$f(x) = g(y).$$

Таким образом мы доказали, что можно применить утверждение 3, значит $\mathcal{P} \mapsto \mathcal{Q}$.

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

1.5 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. Это переменные, которые могут принимать лишь два значения — 0 и 1.

Утверждение 4 (простые условия)

Пусть у нас есть два логических условия A и B, и нам нужно записать на языке алгебры логическое выражение «если верно A, то верно B». Для этого введём две булевы переменные x и y, смысл этих переменных будет следующий

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

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

Рассмотрим всевозможные комбинации событий A и B на языке булевых переменных.

1. «Если верно A, то верно B». На языке наших переменных это записывается как «если $x = 0$, то $y = 0$». Будем записывать это алгебраически следующим образом
$$x \geq y.$$
   - 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 $1 \geq 1$.

   Значит наше неравенство по смыслу совпадает с исходным логическим условием.
2. Условие «если $x = 0$, то $y = 1$» записывается как
$$x \geq 1 - y.$$

- 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 $\{0, 1\}$.

Записать через $x \geq y - 1$ было бы некорректно, поскольку при $y = 0$ в правой части получилось бы отрицательное число, а при использовании булевых переменных нужно оперировать лишь 0 и 1.

3. «Если $x = 1$, то $y = 0$». По аналогии с предыдущим пунктом это записывается как
$$1 - x \geq y.$$
4. «Если $x = 1$, то $y = 1$». По аналогии с предыдущими пунктами это записывается как
$$1 - x \geq 1 - y,$$
или то же самое
$$x \leq y.$$

Подытожим

1. «Если $x = 0$, то $y = 0$» $\quad x \geq y.$
2. «Если $x = 0$, то $y = 1$» $\quad x \geq 1 - y.$
3. «Если $x = 1$, то $y = 0$» $\quad 1 - x \geq y.$
4. «Если $x = 1$, то $y = 1$» $\quad 1 - x \geq 1 - y \iff x \leq y.$

Утверждение 5 (сложные условия)

Пусть теперь у нас есть множества условий $\mathcal{A} = \{A_i\}$ и $\mathcal{B} = \{B_k\}$, которым мы будем сопоставлять булевы переменные x и y.

1. Пусть нам хочется алгебраически записать следующее условие: «если некоторые условия из множества $\mathcal{A}$ истины (а остальные ложны), то все условия из множества $\mathcal{B}$ истины». Введём булев вектор $x = (x_i)$, соответствующий условиям из $\mathcal{A}$. Также введём не пересекающиеся множества индексов $I^0, I^1 \subset I$, соответствующие верным и неверным условиям из $\mathcal{A}$ следующим образом
$$x_i = \begin{cases} 0, & i \in I^0 \Leftrightarrow A_i \text{ — истина} \\ 1, & i \in I^1 \Leftrightarrow A_i \text{ — ложь} \end{cases}$$
Ещё введём булевы переменную y, которая обозначает следующее
$$y = \begin{cases} 0, & \text{все условия из } \mathcal{B} \text{ истины} \\ 1, & \text{не все условия из } \mathcal{B} \text{ истины} \end{cases}$$

Фактически нам бы хотелось записать следующее: «если вектор x имеет такой вид, то $y = 0$». Записать алгебраически это можно следующим образом

$$\sum_{i \in I^1} (1 - x_i) + \sum_{i \in I^0} x_i \geq y.$$

Действительно, если вектор x имеет нужный нам вид, то обе суммы равняются нулю, значит y ничего не остаётся кроме как быть равным нулю.

2. Пусть нам хочется алгебраически записать следующее условие: «если все события из $\mathcal{A}$ истины, то некоторые условия из множества $\mathcal{B}$ истины (а остальные ложны)». Введём булев вектор $y = (y_k)$, соответствующий условиям из $\mathcal{B}$. Также введём не пересекающиеся множества индексов $K^0, K^1 \subset K$, соответствующие верным и неверным условиям из $\mathcal{B}$ следующим образом

$$y_k = \begin{cases} 0, & k \in K^0 \Leftrightarrow B_k \text{ — истина} \\ 1, & k \in K^1 \Leftrightarrow B_k \text{ — ложь} \end{cases}$$

Ещё введём булевы переменную x, которая обозначает следующее

$$x = \begin{cases} 0, & \text{все условия из } \mathcal{A} \text{ истины} \\ 1, & \text{не все условия из } \mathcal{A} \text{ истины} \end{cases}$$

Фактически нам бы хотелось записать следующее: «если $x = 0$, то y имеет такой-то вид». Записать алгебраически это можно следующим образом

$$\left|K^0 \cup K^1\right| \cdot x \geq \sum_{k \in K^1} (1 - y_k) + \sum_{k \in K^0} y_k.$$

Зачем здесь нужен коэффициент $\left|K^0 \cup K^1\right|$?. Если $x = 0$, то наше условие соблюдается, поскольку в правой части тогда обе суммы должны равняться нулю, а значит y принимает нужным нам вид. Однако дело в том, что 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 правая часть без коэффициента может лишь не больше 1. Это означает, что наше алгебраическое выражение по смыслу не совпадает с изначальными логическими условиями. 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. Рассмотрим самый общий случай: «если некоторые условия из $\mathcal{A}$ истины (а остальные ложны), то некоторые условия из множества $\mathcal{B}$ истины (а остальные

ложны)». Для этого введём булевы векторы $x = (x_i)$ и $y = (y_k)$, соответствующие условиям из $\mathcal{A}$ и $\mathcal{B}$ соответственно. Также аналогично с двумя предыдущими пунктами введём множества $I^0, I^1 \subset I$ и $K^0, K^1 \subset K$ следующим образом

$$x_i = \begin{cases} 0, & i \in I^0 \Leftrightarrow A_i \text{ — истина} \\ 1, & i \in I^1 \Leftrightarrow A_i \text{ — ложь} \end{cases}$$

$$y_k = \begin{cases} 0, & k \in K^0 \Leftrightarrow B_k \text{ — истина} \\ 1, & k \in K^1 \Leftrightarrow B_k \text{ — ложь} \end{cases}$$

Фактически нам бы хотелось записать следующее: «если x имеет такой-то вид, то y имеет такой-то вид». Записать алгебраически это можно следующим образом

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

Коэффициент в левой части добавлен по аналогии с предыдущим пунктом (иначе при любых значениях вектора x вектор y не может быть любым).

Подытожим

1. «Если x имеет такой-то вид, то $y = 0$»

$$\sum_{i \in I^1} (1 - x_i) + \sum_{i \in I^0} x_i \geq y.$$

2. «Если $x = 0$, то y имеет такой-то вид»

$$\left|K^0 \cup K^1\right| \cdot x \geq \sum_{k \in K^1} (1 - y_k) + \sum_{k \in K^0} y_k.$$

3. «Если x имеет такой-то вид, то y имеет такой-то вид»

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

Утверждение 6 (альтернативные переменные)

Пусть в нашей задаче сформулированы 2 ограничения

$$f_1(x) \geq b_1,$$

$$f_2(x) \geq b_2,$$

однако нам достаточно, чтобы выполнялось хотя бы одно из них. Тогда такие условия называются *альтернативными*. Как это можно записать алгебраически?

Предположим, что мы реализуем алгоритм, который решает нашу задачу, при этом он находит решения, которые удовлетворяют всем условиям. Введём булеву переменную y, значения которой определяются следующим образом

$$y = \begin{cases} 0, & \text{выполнено первое ограничение} \\ 1, & \text{выполнено второе ограничение} \end{cases}$$

В идеальном случае наш алгоритм сам выберет значение этой переменной и на его основании построит оптимальное решение, однако какое бы значение он не выбрал, итоговое решение должно удовлетворять одному из ограничений. Алгебраически это можно записать так

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

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

где W — это какая-то большая величина. Какое конкретно значение эта величина 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.

- Если $y = 1$, то имеем

  $$f_1(x) \geq b_1,$$

  $$f_2(x) \geq b_2 - W \to -\infty.$$

  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.

Другой способ

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

$$f_1(x) \geq b_1 - W(1 - y_1),$$

$$f_2(x) \geq b_2 - W(1 - y_2),$$

$$y_1 + y_2 = 1.$$

Без условия $y_1 + y_2 = 1$ нет гарантий, что одно из неравенств будет верно.

Ограничения в другую сторону

Если у нас есть ограничения с другим знаком

$$f_1(x) \le b_1,$$

$$f_2(x) \le b_2,$$

то записать алгебраически их можно следующим образом

$$f_1(x) \le b_1 + W(1 - y_1),$$

$$f_2(x) \le b_2 + W y_2,$$

$$y_1 + y_2 = 1.$$

Аналогично можно было бы записать через y.

Утверждение 7 (замена нелинейностей)

Пусть при решение задачи в некотором выражении нам встретилась какая-то нелинейность, например $\hat{x} \cdot \hat{y}$ ($\hat{x}$ и $\hat{y}$ — булевы переменные). Работать с нелинейностями неудобно, поэтому хотелось бы заменить это на новую булеву переменную

$$\hat{z} = \hat{x} \cdot \hat{y}.$$

Однако просто заменить в выражении $\hat{x} \cdot \hat{y}$ на $\hat{z}$ нельзя, поскольку нужно изменить ограничения. Нам нужно ввести ограничение на новую переменную

$$\hat{z} = 1 \quad \Longleftrightarrow \quad \hat{x} = 1 \wedge \hat{y} = 1$$

Данное ограничение можно расписать через две импликации

1. Если $\hat{z} = 1$, то $\hat{x} = 1$ и $\hat{y} = 1$. Данное логическое условие можно записать алгебраически, используя утверждение 5. Будем использовать самый общий случай

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

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

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

$$\Updownarrow$$

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

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

2. Если $\hat{x} = 1$ и $\hat{y} = 1$, то $\hat{z} = 1$. Данное логическое условие можно записать алгебраически, используя утверждение 5. Будем использовать самый общий случай

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

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

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

$$\Updownarrow$$

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

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

Итого при замене $\hat{z} = \hat{x} \cdot \hat{y}$ нужно добавить два ограничения

$$\boxed{\begin{cases} 2\hat{z} \leq \hat{x} + \hat{y}, \\ \hat{z} + 1 \geq \hat{x} + \hat{y}. \end{cases}}$$

Глава 2

Динамическое программирование

Динамическое программирование эффективно, 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... **Главная идея динамического программирования** заключается в том, чтобы не пытаться решить задачу непосредственно, а поместить её в семейство аналогичных задач, среди которых есть просто решаемые.

2.1 Задача загрузки судна

Задача

Есть грузовое судно и набор различных контейнеров. Требуется загрузить судно контейнерами таким образом, чтобы при их дальнейшей продаже заработать как можно больше.

Формальное описание:

- есть N контейнеров;
- x_i — сколько контейнеров с номером i нужно взять на судно;
- h_i — место на грузовой площадке, занимаемое контейнером с номером i;
- c_i — ценность контейнера с номером i;
- P — размер грузовой площадки судна;

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

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

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

Решение

Для решения задачи рассмотрим семейство аналогичных задач. Задачи данного семейства будут охарактеризовываться парой (n,p), где

- p — место на площадке;
- n — минимальный номер для контейнеров, которые у нас есть.

$$n = 1 \dots N \qquad p = 0 \dots P.$$

Вместо рассмотрения общей 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, а также в которых нам «недоступна» вся грузовая площадка судна, а лишь её часть размером p.

Запишем целевую функцию и ограничения для элемента семейства

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

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

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

Мы формально записали формулировку задачи для некоторого элемента семейства, при этом этот элемент характеризуется парой (n,p).

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 контейнеры с номером N.

Введём обозначение. Пусть $\boxed{f_n(p)}$ — оптимальное значение целевой функции задачи (n,p). По своей суть $f_n(p)$ —это максимальный доход, который мы получим, если будем на площадку размера p грузить контейнеры с номерами $n, n+1, n+2, \dots, N$.

Легко заметить, что

$$f_N(p) = \left[\frac{p}{h_N}\right],$$

то есть решать задачу (N, p) для любого $p \leq P$ мы умеем. 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 накосячили и пошли неоптимально, стратегия точно не получится оптимальной.

Вернёмся к задаче. Предположим, что мы умеем считать

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

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 $h_n \cdot x = h_n$. Запишем целевую функцию

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

Мы получили **рекуррентное соотношение динамического программирования**. На данном этапе $f_n(p)$ можно получить перебором значение $x_n = x$, а $f_{n+1}(\ldots)$ нам уже известно.

Какой здесь будет алгоритм при реализации? Алгоритм будет включать в себя N шагов, на каждом из которых мы будем вычислять $f_n(p)$. По сути мы будем заполнять табличку

Для решения задачи нам важно лишь $f_1(P)$, считать $f_1(p)$ для всех $p \leq P$ не нужно.

Алгоритм имеет две стадии:

1. *обратный ход* $N \to 1$ — заполнение таблицы,
2. *прямой ход* $1 \to N$ — вычисление оптимального решения.

p	$f_1(p)$	$f_2(p)$	$\dots$	$f_n(p)$	$f_{N-1}(p)$	$f_N(p)$
0	$\times$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	$\times$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
2	$\times$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
3	$\times$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$P-1$	$\times$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
P	$\dots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Сначала мы будем идти с конца и для всех значений p считать $f_N(p)$. Потом на основании этого будем для всех значений p считать $f_{N-1}(p)$ и так далее. Когда мы заполним все столбцы, кроме $f_1(p)$, мы посчитаем $f_1(P)$ и на этом заполнение таблички закончим.

Как теперь по табличке получить само оптимальное решение — отдельный вопрос, который будет рассмотрен позже.

2.2 N-шаговый процесс принятия решений