

# Задача линейного программирования

Даня Меркулов

## 1 Примеры задач линейного программирования

### 1.1 Что такое линейное программирование?



В общем случае все задачи с линейной целевой функцией и линейными функциональными ограничениями можно считать задачами линейного программирования. Однако существует несколько стандартных формулировок.

$$\begin{aligned} & \min_{x \in \mathbb{R}^n} c^\top x \\ \text{s.t. } & Ax \leq b \end{aligned} \tag{LP.Basic}$$

для некоторых векторов  $c \in \mathbb{R}^n$ ,  $b \in \mathbb{R}^m$  и матрицы  $A \in \mathbb{R}^{m \times n}$ , где неравенства — покомпонентные. Мы будем часто использовать эту формулировку для построения интуиции.

Широко используется **стандартная форма** записи задачи линейного программирования. Пусть заданы векторы  $c \in \mathbb{R}^n$ ,  $b \in \mathbb{R}^m$  и матрица  $A \in \mathbb{R}^{m \times n}$ .

$$\begin{aligned} & \min_{x \in \mathbb{R}^n} c^\top x \\ \text{s.t. } & Ax = b \\ & x_i \geq 0, \quad i = 1, \dots, n \end{aligned} \tag{LP.Standard}$$

## 1.2 Задача о диете



Представьте, что вам нужно составить план диеты из некоторых продуктов: бананы, пироги, курица, яйца, рыба. Каждый из продуктов имеет свой вектор питательных веществ. Таким образом, все питательные вещества можно представить в виде матрицы  $W$ .

Предположим, что у нас есть вектор требований для каждого питательного вещества  $r \in \mathbb{R}^n$ . Нам нужно найти самую дешёвую диету, которая удовлетворяет всем требованиям:

$$\begin{aligned} & \min_{x \in \mathbb{R}^p} c^\top x \\ \text{s.t. } & Wx \succeq r \\ & x_i \geq 0, \quad i = 1, \dots, p \end{aligned}$$

Open In Colab

### 1.3 Минимизация выпуклой функции как задача линейного программирования



Рисунок 1: Как задача линейного программирования может помочь с общей задачей выпуклой оптимизации

- Функция выпукла, если она может быть представлена как поточечный максимум линейных функций.
- В пространствах большой размерности аппроксимация может потребовать огромного количества функций.
- Существуют более эффективные солверы для выпуклой оптимизации (не сводящиеся к LP).

### 1.4 Транспортная задача

Типичная транспортная задача заключается в распределении товара от производителей к потребителям. Цель состоит в минимизации общих затрат на транспортировку при соблюдении ограничений на количество товара на каждом источнике и удовлетворении требований к спросу на каждом пункте назначения.



Пункт назначения / Источник	Арнем [€/тонна]	Гауда [€/тонна]	Спрос [тонн]
Лондон	n/a	2.5	125
Берлин	2.5	n/a	175
Маастрихт	1.6	2.0	225
Амстердам	1.4	1.0	250
Уtrecht	0.8	1.0	225
Гаага	1.4	0.8	200
<b>Макс. производство [тонн]</b>	<b>550</b>	<b>700</b>	

$$\text{Минимизировать: Стоимость} = \sum_{c \in \text{Пункты назначения}} \sum_{s \in \text{Источники}} T[c, s]x[c, s]$$

$$\sum_{c \in \text{Пункты назначения}} x[c, s] \leq \text{Поставка}[s] \quad \forall s \in \text{Источники}$$

$$\sum_{s \in \text{Источники}} x[c, s] = \text{Спрос}[c] \quad \forall c \in \text{Пункты назначения}$$

Задачу можно представить в виде следующего графа:

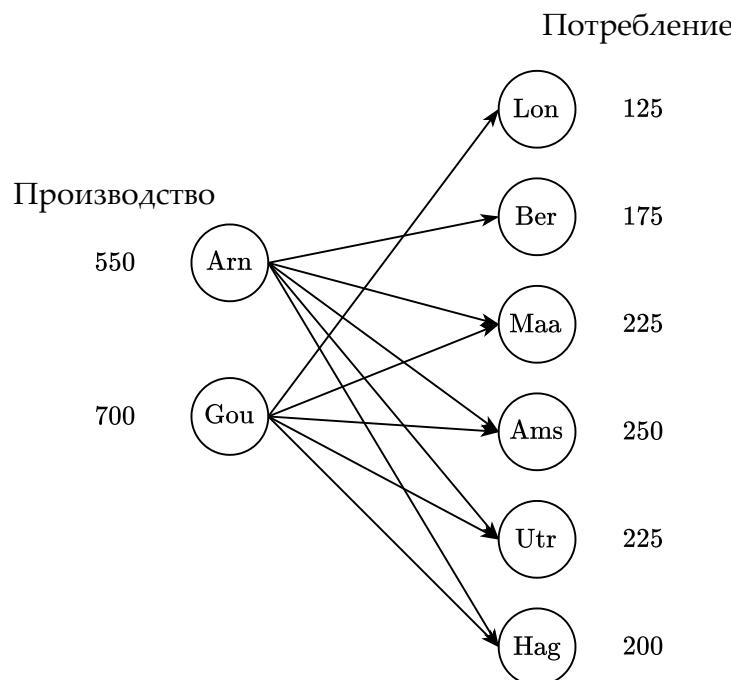


Рисунок 3: Граф, связанный с задачей

## 2 Как получить задачу линейного программирования?

### 2.1 Основные преобразования

- Максимум-минимум

$$\begin{array}{ll} \min_{x \in \mathbb{R}^n} c^\top x & \leftrightarrow \max_{x \in \mathbb{R}^n} -c^\top x \\ \text{s.t. } Ax \leq b & \text{s.t. } Ax \leq b \end{array}$$

- Равенство к неравенству

$$Ax = b \leftrightarrow \begin{cases} Ax \leq b \\ Ax \geq b \end{cases}$$

- Неравенство к равенству, увеличивая размерность задачи на  $m$ .

$$Ax \leq b \leftrightarrow \begin{cases} Ax + z = b \\ z \geq 0 \end{cases}$$

- Неотрицательные переменные

$$x \leftrightarrow \begin{cases} x = x_+ - x_- \\ x_+ \geq 0 \\ x_- \geq 0 \end{cases}$$

### 2.2 Задача аппроксимации Чебышева

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_\infty \leftrightarrow \min_{x \in \mathbb{R}^n} \max_i |a_i^\top x - b_i|$$

Можно записать эквивалентную задачу линейного программирования с заменой максимальной координаты вектора:

$$\begin{array}{ll} \min_{t \in \mathbb{R}, x \in \mathbb{R}^n} & t \\ \text{s.t. } & a_i^\top x - b_i \leq t, \quad i = 1, \dots, m \\ & -a_i^\top x + b_i \leq t, \quad i = 1, \dots, m \end{array}$$

### 2.3 Задача $\ell_1$ -аппроксимации

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_1 \leftrightarrow \min_{x \in \mathbb{R}^n} \sum_{i=1}^m |a_i^\top x - b_i|$$

Можно записать эквивалентную задачу линейного программирования с заменой суммы координат вектора:

$$\begin{aligned} & \min_{t \in \mathbb{R}^m, x \in \mathbb{R}^n} \mathbf{1}^T t \\ \text{s.t. } & a_i^T x - b_i \leq t_i, \quad i = 1, \dots, m \\ & -a_i^T x + b_i \leq t_i, \quad i = 1, \dots, m \end{aligned}$$

## 2.4 Задача смешивания: от нелинейных ограничений к ЛП<sup>1</sup>

Производственное предприятие получает заказ на 100 литров раствора с определённой концентрацией (например, 4% сахарного раствора). На складе есть:

Компонент	Сахар (%)	Стоимость (\$/л)
Концентрат А (Добрый кола)	10.6	1.25
Концентрат В (Север кола)	4.5	1.02
Вода (Псыж)	0.0	0.62

**Цель:** Найти смесь с минимальной стоимостью, которая удовлетворит заказ.

### 2.4.1 Целевая функция

Минимизировать стоимость:

$$\text{Cost} = \sum_{c \in C} x_c P_c$$

где  $x_c$  — объём используемого компонента  $c$ , и  $P_c$  — его цена.

### 2.4.2 Ограничение на объём

Убедитесь, что общий объём  $V$ :

$$V = \sum_{c \in C} x_c$$

#### 2.4.2.1 Ограничение на состав

Убедитесь, что содержание сахара — 4%:

$$\bar{A} = \frac{\sum_{c \in C} x_c A_c}{\sum_{c \in C} x_c}$$

Линеаризованная версия:

$$0 = \sum_{c \in C} x_c (A_c - \bar{A})$$

Это можно решить с помощью линейного программирования.

#### Код

<sup>1</sup>[Источник](#)

### 3 Симплекс-метод

#### 3.1 Геометрия симплекс-метода



Рассмотрим следующую простую формулировку задачи линейного программирования:

$$\begin{aligned} & \min_{x \in \mathbb{R}^n} c^\top x \\ \text{s.t. } & Ax \leq b \end{aligned} \tag{LP.Inequality}$$

- Определение: **базис**  $\mathcal{B}$  — это подмножество  $n$  (целых) чисел между 1 и  $m$ , такое что  $\text{rank } A_{\mathcal{B}} = n$ .
- Обратите внимание, что мы можем связать подматрицу  $A_{\mathcal{B}}$  и соответствующую правую часть  $b_{\mathcal{B}}$  с базисом  $\mathcal{B}$ .
- Также мы можем получить точку пересечения всех этих гиперплоскостей из базиса:  $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b_{\mathcal{B}}$ .
- Если  $Ax_{\mathcal{B}} \leq b$ , то базис  $\mathcal{B}$  является **допустимым**.
- Базис  $\mathcal{B}$  оптимальен, если  $x_{\mathcal{B}}$  является решением задачи LP.Inequality.
- $x_{\mathcal{B}}$  называют **базисной точкой** или базисным решением (иногда её тоже называют **базисом**).

### 3.2 Если решение задачи линейного программирования существует, то оно лежит в вершине



#### Theorem

- Если задача линейного программирования в стандартной форме имеет непустое бюджетное множество, то существует по крайней мере одна допустимая базисная точка.
- Если задача линейного программирования в стандартной форме имеет решения, то по крайней мере одно из таких решений является оптимальной базисной точкой.
- Если задача линейного программирования в стандартной форме допустима и ограничена, то она имеет оптимальное решение.

Для доказательства см. теорему 13.2 в [Numerical Optimization by Jorge Nocedal and Stephen J. Wright](#)

Верхнеуровневая идея симплекс-метода:

- Убедитесь, что вы находитесь в вершине.
- Проверьте оптимальность.
- Если необходимо, перейдите к другой вершине (измените базис).
- Повторяйте, пока не сойдёtesь.

### 3.3 Оптимальный базис



Поскольку у нас есть базис, мы можем разложить наш целевой вектор  $c$  в этом базисе и найти скалярные коэффициенты  $\lambda_{\mathcal{B}}$ :

$$\lambda_{\mathcal{B}}^T A_{\mathcal{B}} = c^T \leftrightarrow \lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$$

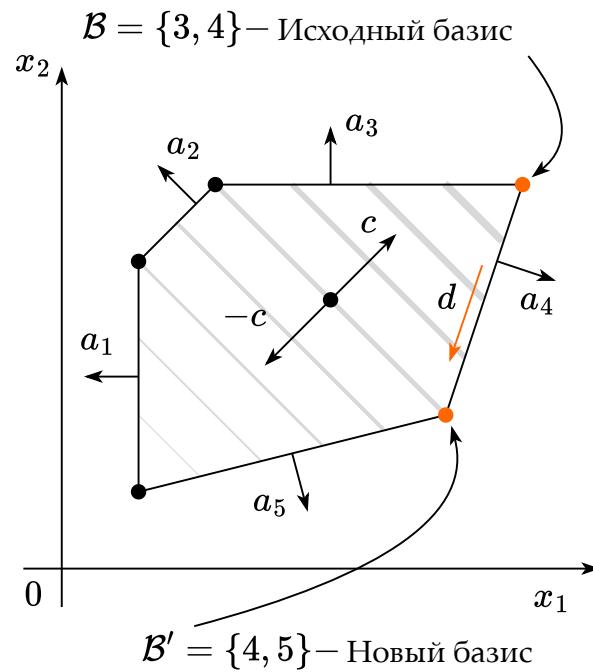
#### Theorem

Если все компоненты  $\lambda_{\mathcal{B}}$  неположительны и  $\mathcal{B}$  допустим, то  $\mathcal{B}$  оптимальен.

**Доказательство** Предположим противное, то есть  $\lambda_{\mathcal{B}} \leq 0$  и  $\mathcal{B}$  допустим, но не оптимальен.

$$\begin{aligned} \exists x^* : Ax^* &\leq b, c^T x^* < c^T x_{\mathcal{B}} \\ A_{\mathcal{B}} x^* &\leq b_{\mathcal{B}} \mid \lambda_{\mathcal{B}}^T \leq 0 \\ \lambda_{\mathcal{B}}^T A_{\mathcal{B}} x^* &\geq \lambda_{\mathcal{B}}^T b_{\mathcal{B}} \\ c^T x^* &\geq \lambda_{\mathcal{B}}^T A_{\mathcal{B}} x_{\mathcal{B}} \\ c^T x^* &\geq c^T x_{\mathcal{B}} \end{aligned}$$

### 3.4 Изменение базиса



Предположим, что некоторые из коэффициентов  $\lambda_{\mathcal{B}}$  положительны. В этом случае необходимо осуществить переход по ребру многогранника к новой вершине, то есть произвести замену базиса.

- Предположим, что у нас есть базис  $\mathcal{B}: \lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$
- Предположим, что  $\lambda_{\mathcal{B}}^k > 0$ . Мы хотим удалить  $k$  из базиса и сформировать новый:

$$\begin{cases} A_{\mathcal{B} \setminus \{k\}} d = 0 \\ a_k^T d = -1 \end{cases} \quad c^T d = \lambda_{\mathcal{B}}^T A_{\mathcal{B}} d = \sum_{i=1}^n \lambda_{\mathcal{B}}^i (A_{\mathcal{B}} d)^i = -\lambda_{\mathcal{B}}^k < 0$$

- Для всех  $j \notin \mathcal{B}$  рассчитаем размер шага проекции:

$$\mu_j = \frac{b_j - a_j^T x_{\mathcal{B}}}{a_j^T d}$$

- Определим новую вершину, которую мы добавим в новый базис:

$$\begin{aligned} t &= \arg \min_j \{\mu_j \mid \mu_j > 0\} \\ \mathcal{B}' &= \mathcal{B} \setminus \{k\} \cup \{t\} \\ x_{\mathcal{B}'} &= x_{\mathcal{B}} + \mu_t d = A_{\mathcal{B}'}^{-1} b_{\mathcal{B}'} \end{aligned}$$

- Обратите внимание, что изменение базиса приводит к уменьшению целевой функции:  $c^T x_{\mathcal{B}'} = c^T(x_{\mathcal{B}} + \mu_t d) = c^T x_{\mathcal{B}} + \mu_t c^T d$

### 3.5 Поиск начального допустимого базиса

Нам нужно решить следующую задачу:

$$\begin{aligned} & \min_{x \in \mathbb{R}^n} c^\top x \\ \text{s.t. } & Ax \leq b \end{aligned} \tag{1}$$

Предложенный алгоритм требует начального допустимого базиса.

Начнём с переформулировки задачи:

$$\begin{aligned} & \min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} c^\top (y - z) \\ \text{s.t. } & Ay - Az \leq b \\ & y \geq 0, z \geq 0 \end{aligned} \tag{2}$$

Зная решение задачи (2), можно восстановить решение задачи (1), и наоборот.

$$x = y - z \quad \Leftrightarrow \quad y_i = \max(x_i, 0), \quad z_i = \max(-x_i, 0)$$

Теперь мы попытаемся сформулировать новую задачу линейного программирования, решение которой будет допустимой базисной точкой для Задачи 2. Это означает, что мы сначала запускаем симплекс-метод для задачи Phase-1, а затем запускаем задачу Phase-2 с известным начальным решением. Обратите внимание, что допустимое базисное решение для Phase-1 должно быть легко вычислимо.

$$\begin{aligned} & \min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} c^\top (y - z) \\ \text{s.t. } & Ay - Az \leq b \\ & y \geq 0, z \geq 0 \end{aligned} \quad (\text{Фаза-2 (главная задача ЛП)})$$

$$\begin{aligned} & \min_{\xi \in \mathbb{R}^m, y \in \mathbb{R}^n, z \in \mathbb{R}^n} \sum_{i=1}^m \xi_i \\ \text{s.t. } & Ay - Az \leq b + \xi \\ & y \geq 0, z \geq 0, \xi \geq 0 \end{aligned} \quad (\text{Фаза-1})$$

- Если Фаза-2 (главная задача ЛП) имеет допустимое решение, то оптимум Фаза-1 равен нулю (т.е. все переменные  $\xi_i$  равны нулю).

**Доказательство:** тривиальная проверка.

- Если оптимум Фаза-1 равен нулю (т.е. все переменные  $\xi_i$  равны нулю), то мы получаем допустимый базис для Фаза-2.

**Доказательство:** тривиальная проверка.

- Теперь мы знаем, что если мы можем решить задачу Фаза-1, то мы либо найдём начальную точку для симплекс-метода в исходном методе (если переменные  $\xi_i$  равны нулю), либо проверим, что исходная задача не имеет допустимого решения (если переменные  $\xi_i$  не равны нулю).

- Но как решить задачу Фаза-1? Она имеет допустимое базисное решение (задача имеет  $2n + m$  переменных, и точка ниже гарантирует, что  $2n + m$  неравенств удовлетворяются как равенства (активны).)

$$z = 0 \quad y = 0 \quad \xi_i = \max(0, -b_i)$$

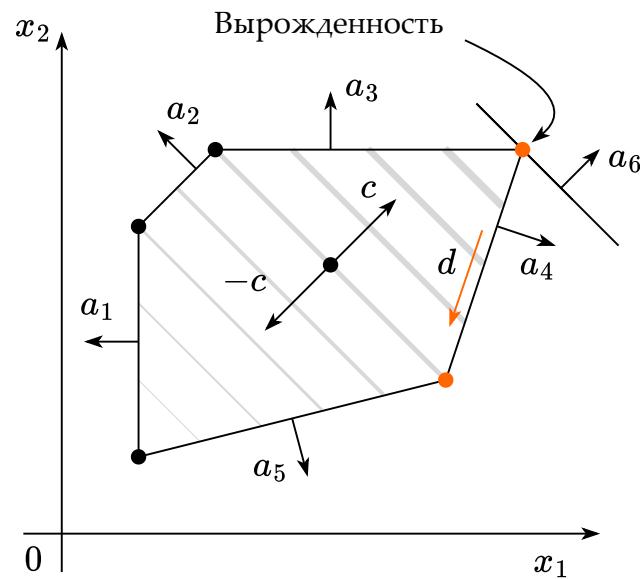
## 4 Сходимость симплекс-метода

### 4.1 Неограниченное бюджетное множество



В этом случае не найдётся ни одного положительного  $\mu_j$ .

## 4.2 Вырожденность вершин



Случаи вырожденности требуют особого рассмотрения. В отсутствие вырожденности на каждой итерации гарантируется монотонное убывание значения целевой функции.

#### 4.3 Экспоненциальная сходимость



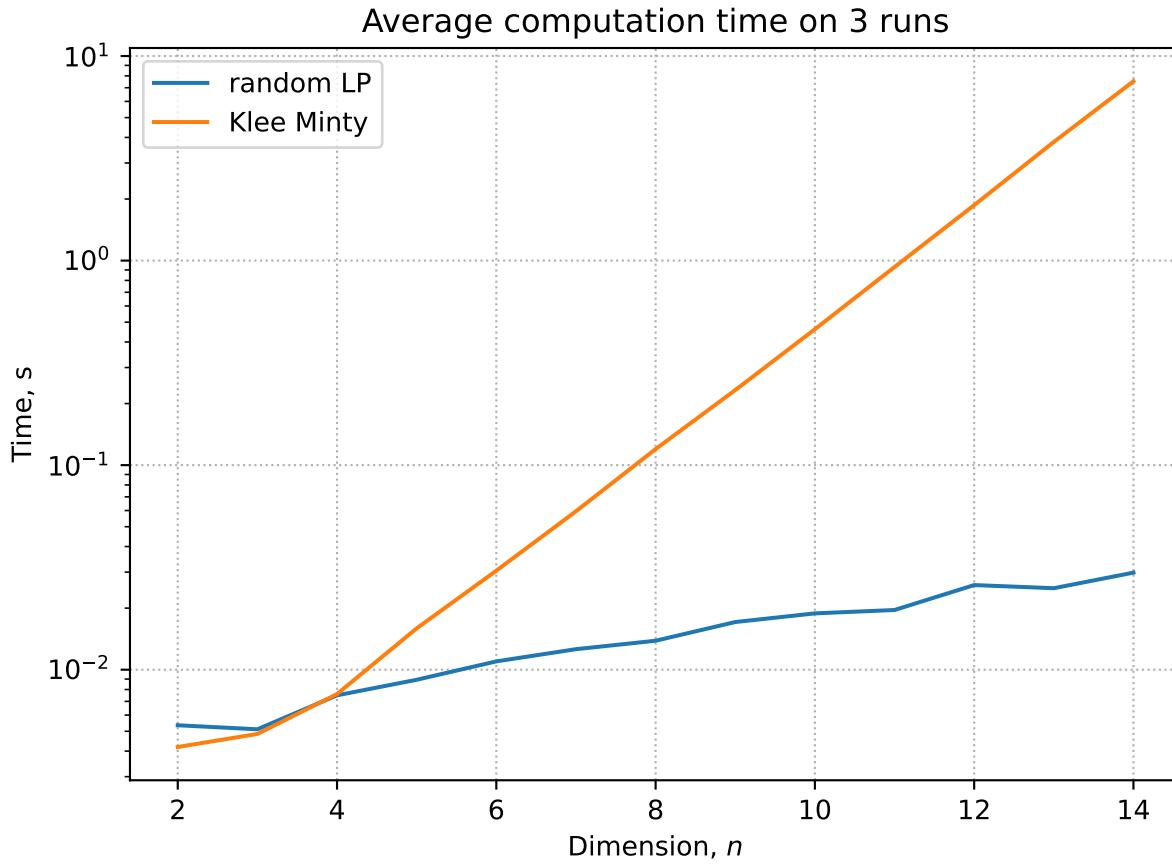
- Много прикладных задач может быть сформулировано в виде задач линейного программирования.
- Симплекс-метод прост в своей основе, но в худшем случае может работать экспоненциально долго.
- Метод эллипсоидов Хачияна (1979) стал первым алгоритмом с доказанной полиномиальной сложностью для задач ЛП. Однако он обычно работает медленнее, чем симплекс-метод в реальных небольших задачах.
- Основной прорыв — метод Кармаркара (1984) для решения задач ЛП с использованием метода внутренней точки.
- Методы внутренней точки являются последним словом в этой области. Тем не менее, для типовых задач ЛП качественные реализации симплекс-метода и методов внутренней точки показывают схожую производительность.

#### 4.4 Пример Klee Minty

Так как число вершин конечно, сходимость алгоритма гарантирована (за исключением вырожденных случаев, которые здесь не рассматриваются). Тем не менее, сходимость может быть экспоненциально медленной из-за потенциально большого числа вершин. Существует пример, в котором симплекс-метод вынужден пройти через все вершины многогранника.

В следующей задаче симплекс-метод должен проверить  $2^n - 1$  вершин с  $x_0 = 0$ .

$$\begin{aligned}
 & \max_{x \in \mathbb{R}^n} 2^{n-1}x_1 + 2^{n-2}x_2 + \dots + 2x_{n-1} + x_n \\
 \text{s.t. } & x_1 \leq 5 \\
 & 4x_1 + x_2 \leq 25 \\
 & 8x_1 + 4x_2 + x_3 \leq 125 \\
 & \dots \\
 & 2^n x_1 + 2^{n-1}x_2 + 2^{n-2}x_3 + \dots + x_n \leq 5^n \\
 & x \geq 0
 \end{aligned}$$



## 5 Смешанное целочисленное программирование (MIP)

### 5.1 Сложность MIP

Рассмотрим следующую задачу смешанного целочисленного программирования (MIP):

$$\begin{aligned}
 z = & 8x_1 + 11x_2 + 6x_3 + 4x_4 \rightarrow \max_{x_1, x_2, x_3, x_4} \\
 \text{s.t. } & 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
 & x_i \in \{0, 1\} \quad \forall i
 \end{aligned} \tag{3}$$

Упростим её до:

$$\begin{aligned} z &= 8x_1 + 11x_2 + 6x_3 + 4x_4 \rightarrow \max_{x_1, x_2, x_3, x_4} \\ \text{s.t. } &5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\ &x_i \in [0, 1] \quad \forall i \end{aligned} \tag{4}$$

Оптимальное решение

$$x_1 = 0, x_2 = x_3 = x_4 = 1, \text{ и } z = 21.$$

Оптимальное решение

$$x_1 = x_2 = 1, x_3 = 0.5, x_4 = 0, \text{ и } z = 22.$$

- Округление  $x_3 = 0$ : даёт  $z = 19$ .
- Округление  $x_3 = 1$ : недопустимо.

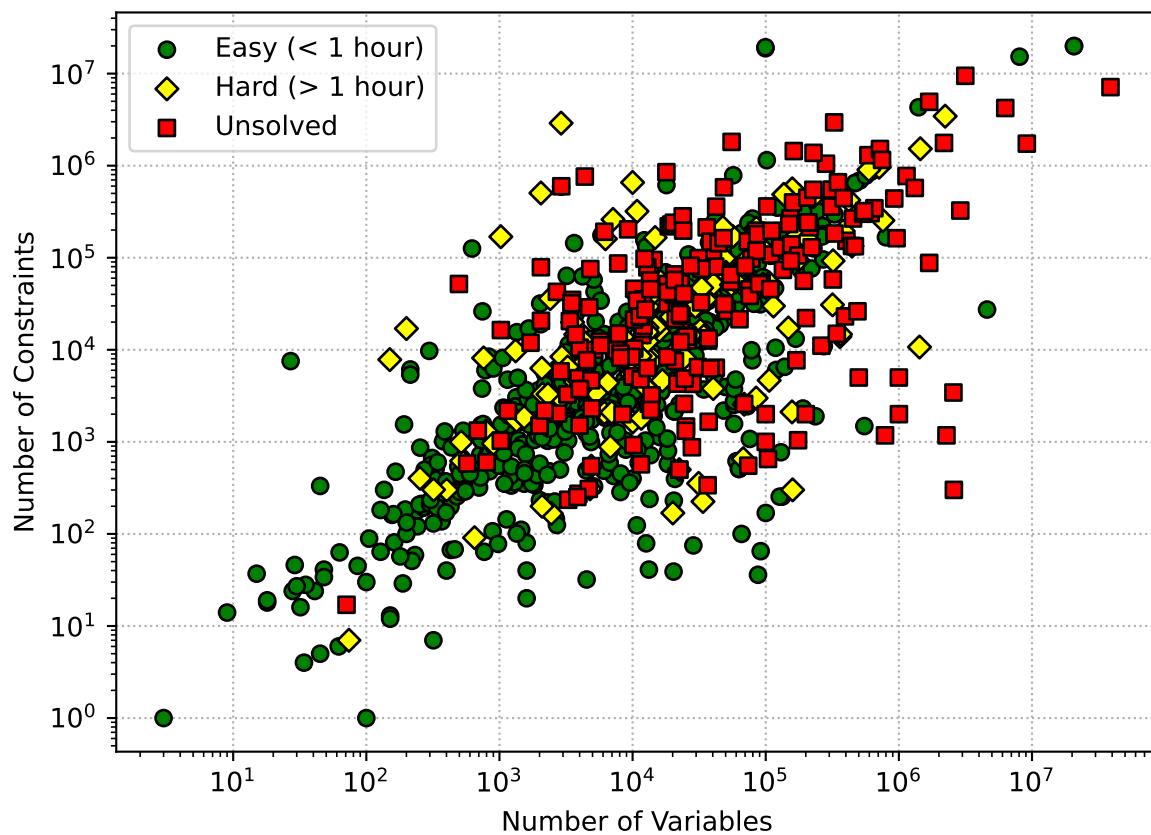
### ! МИР намного сложнее, чем ЛП

- Наивное округление решения, полученного для ЛП-релаксации исходной задачи МИР, может привести к недопустимому или неоптимальному решению.
- Общая задача МИР является NP-трудной задачей.
- Однако, если матрица коэффициентов МИР является *полностью унимодулярной матрицей*, то она может быть решена за полиномиальное время.

## 5.2 Непредсказуемая сложность МИР

- Трудно предсказать, что будет решено быстро, а что потребует много времени
-  Датасет
-  Код

### Running time to optimality for different MIPs MIPLIB 2017 Collection Set



### 5.3 Прогресс аппаратного vs программного обеспечения

Что бы вы выбрали, если предположить, что вопрос поставлен корректно (вы можете скомпилировать ПО для любого оборудования, и задача в обоих случаях одна и та же)? Мы рассмотрим период с 1992 по 2023 год.

#### 🔥 Аппаратное обеспечение

Решение МИР с использованием старого ПО на современном оборудовании

#### 🔥 Программное обеспечение

Решение МИР с использованием современного ПО на старом оборудовании

$\approx 1.664.510 \times$  ускорение

Закон Мура утверждает, что вычислительная мощность удваивается каждые 18 месяцев.

$\approx 2.349.000 \times$  ускорение

Р. Бикси провёл масштабный эксперимент по сравнению производительности всех версий CPLEX с 1992 по 2007 год и измерил общий прогресс ПО (29000 раз), позже (в 2009 году) он стал одним из основателей Gurobi Optimization, которое дало дополнительное  $\approx 81$  ускорение на МИР.

Оказывается, что если вам нужно решить МИР, лучше использовать старый компьютер и современные методы, чем наоборот, самый новый компьютер и методы начала 1990-х годов.<sup>2</sup>

## 5.4 Источники

- Теория оптимизации (МАТН4230) курс @ СУНК, профессор Тейюн Цень

## 6 Задачи на дом

1. **Производство чехлов.** [20 баллов] Lizard Corp производит чехлы для следующих продуктов:

- телефоны
- наушники
- ноутбуки

Производственные мощности компании таковы, что при полной загрузке производства выпускем чехлов для наушников, мы можем произвести 5000 штук в день. Если мы посвятим всю производственную мощность чехлам для телефонов или ноутбуков, мы можем произвести 4000 или 2000 штук в день.

Производственный цикл — одна неделя (6 рабочих дней), и недельную продукцию необходимо разместить на складе до отгрузки. Хранение 1000 чехлов для наушников (включая упаковку) занимает 30 кубических футов. Хранение 1000 чехлов для телефонов (включая упаковку) занимает 50 кубических футов, а 1000 чехлов для ноутбуков — 200 кубических футов. Доступный складской объём — 1500 кубических футов. В силу коммерческих соглашений Lizard Corp должна поставлять не менее 6000 чехлов для наушников и 4000 чехлов для ноутбуков в неделю для усиления распространения продукта. Отдел маркетинга оценивает, что недельный спрос на чехлы для наушников, телефонов и ноутбуков не превышает 15 000, 12 000 и 8000 единиц соответственно; следовательно, компания не хочет производить больше этих объёмов для указанных видов чехлов.

Наконец, чистая прибыль на один чехол для наушников, чехол для телефона и чехол для ноутбука составляет 5, 7 и 12 долларов соответственно.

Цель — определить производственный график, который максимизирует общий чистый доход.

1. Напишите формулировку задачи линейного программирования для этой задачи. Используйте следующие переменные:

- $y_1$  = количество чехлов для наушников, произведенных за неделю,
- $y_2$  = количество чехлов для телефонов, произведенных за неделю,
- $y_3$  = количество чехлов для ноутбуков, произведенных за неделю.

<sup>2</sup>R. Bixby report Recent study

## 2. Найдите решение задачи с помощью PyOMO

```
!pip install pyomo
! sudo apt-get install glpk-utils --quiet # GLPK
! sudo apt-get install coinor-cbc --quiet # CoinOR
```

3. Проведите анализ чувствительности. Какое ограничение можно ослабить, чтобы увеличить прибыль? Докажите это численно.

2. Докажите оптимальность решения [10 баллов]

$$x = \left( \frac{7}{3}, 0, \frac{1}{3} \right)^T$$

для следующей задачи линейного программирования:

$$\begin{aligned} 9x_1 + 3x_2 + 7x_3 &\rightarrow \max_{x \in \mathbb{R}^3} \\ \text{s.t. } 2x_1 + x_2 + 3x_3 &\leq 6 \\ 5x_1 + 4x_2 + x_3 &\leq 12 \\ 3x_3 &\leq 1, \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

В этой задаче нельзя использовать никакие численные алгоритмы.

3. [10 баллов] предположим, небольшая мастерская делает деревянные игрушки, где на каждый игрушечный поезд требуется одна заготовка древесины и 2 банки краски, а на каждую игрушечную лодку — одна заготовка древесины и 1 банка краски. Прибыль с одного поезда составляет 30 долларов, с одной лодки — 20 долларов. Имея запас 80 заготовок древесины и 100 банок краски, какое количество каждой из игрушек следует произвести, чтобы максимизировать прибыль?

1. Запишите оптимационную задачу в стандартной форме, записывая все ограничения в виде неравенств.
2. Нарисуйте допустимое множество и определите  $p^*$  и  $x^*$
3. Другая интерпретация множителей Лагранжа — анализ чувствительности к изменению ограничений. Предположим, что мастерская нашла больше заготовок древесины;  $\lambda_k$  ассоциированная с ограничением на древесину, будет равна частной производной  $-p^*$  по отношению к тому, насколько больше древесины стало доступно. Предположим, что запас увеличился на одну заготовку древесины. Используйте  $\lambda^*$  для оценки того, насколько увеличится прибыль, без решения обновленной оптимационной задачи. Как это согласуется с приведённой выше ценовой интерпретацией множителей Лагранжа?[source](#)