Задача коммивояжера

Введем переменные $x_{ij} = \begin{cases} 1, & \text{если из города } i \text{ едем в город } j \\ 0 & \text{в противном случае} \end{cases}$

Математическая модель

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

при ограничениях

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j \in J,$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i \in J,$$

(исключение подциклов) $$\sum_{i \in S} \sum_{j \in J \setminus S} x_{ij} \geq 1, \quad \forall S \subset J, S \neq \varnothing,$$

или $$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset J, S \neq \varnothing,$$

или $$u_i - u_j + n x_{ij} \leq n - 1, \quad u_i, u_j \geq 0 \quad i, j = 2, \ldots, n$$

Нижние оценки в задаче коммивояжера

Примитивная оценка. Плата за выезд $a_i = \min\limits_{j \neq i} c_{ij}, \;\; i = 1,\ldots, n.$

Плата за въезд $b_j = \min\limits_{i \neq j} (c_{ij} - a_i) \;\; j = 1,\ldots, n.$

Теорема. $OPT(c_{ij}) \geq \sum\limits_{i=1}^{n} a_i + \sum\limits_{j=1}^{n} b_j.$

Доказательство. Положим $c'_{ij} = c_{ij} - a_i, \;\; 1 \leq i, j \leq n.$ Тогда

$OPT(c_{ij}) = OPT(c'_{ij}) + \sum\limits_{i=1}^{n} a_i.$ Аналогично, $c''_{ij} = c'_{ij} - b_j, \;\; 1 \leq i, j \leq n,$ и

$$OPT(c_{ij}) = OPT(c''_{ij}) + \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j \geq \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j. \quad \blacksquare$$

Оценка линейного программирования

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

при ограничениях

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j \in J,$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i \in J,$$

$$\sum_{i \in S} \sum_{j \in J \setminus S} x_{ij} \geq 1, \quad \forall S \subset J, S \neq \varnothing,$$

$$0 \leq x_{ij} \leq 1, \; i, j \in J.$$

задача линейного программирования, дает нижнюю оценку для оптимума, не хуже предыдущей.

1–Деревья для симметричных матриц

Хотим найти гамильтонов цикл минимального веса. Необходимо найти:

– ровно n ребер,

– которые покрывают все вершины,

– имеют минимальный суммарный вес и

– каждая вершина инцидентна ровно двум ребрам.

Заменим последнее условие на следующее:

– одна заданная вершина инцидентна ровно двум ребрам.

Ослабили условия, значит, получим нижнюю оценку.

Алгоритм построения 1-дерева

1. Удаляем заданную вершину и строим остовное дерево минимального веса (алгоритм Крускала, Прима).
2. Добавляем два ребра минимального веса, проходящих через заданную вершину, получаем 1-дерево.

Задача о назначениях

Дано: n рабочих, n станков,

c_{ij} — время выполнения работы i-рабочим на j-м станке.

Найти расстановку рабочих по станкам с минимальным суммарным рабочим временем.

Переменные задачи: $x_{ij} = \begin{cases} 1, & \text{если рабочий } i \text{ получил станок } j \\ 0 & \text{в противном случае} \end{cases}$.

Математическая модель:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

при ограничениях

$$\sum_{i=1}^{n} x_{ij} = 1, \quad 1 \le j \le n,$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad 1 \le i \le n,$$

$$x_{ij} \in \{0,1\}, \quad 1 \le i,j \le n.$$

Оптимальное решение задачи о назначениях дает нижнюю оценку для задачи коммивояжера, не хуже чем оценка 1–деревьев.

Определение. Пусть $\Delta = (\Delta_1,\ldots, \Delta_n)$ — некоторый вектор. Элемент c_{ij} называется *Δ-минимальным*, если $c_{ij} - \Delta_j \le c_{ik} - \Delta_k$ для всех $1 \le k \le n$.

Теорема. Пусть для некоторого Δ существует набор Δ-минимальных элементов $(c_{1\,j(1)},\ldots, c_{n\,j(n)})$ по одному в каждой строке и столбце. Тогда этот набор является оптимальным решением задачи.

Доказательство. Решение $(c_{1\,j(1)},\ldots, c_{n\,j(n)})$ является допустимым и

$$\sum_{i=1}^{n} c_{ij(i)} = \sum_{i=1}^{n} (c_{ij(i)} - \Delta_{j(i)}) + \sum_{j=1}^{n} \Delta_j .$$

В правой части равенства первая сумма является минимальной среди всех допустимых назначений, а вторая сумма является константой, то есть полученное решение является оптимальным. ■

Определение. Для вектора Δ выделим в каждой строке по одному Δ-минимальному элементу и назовем его *Δ-основой*. Другие Δ-минимальные элементы будем называть *альтернативными Δ-основами*. Число столбцов матрицы c_{ij} без Δ-основ назовем *дефектом*.

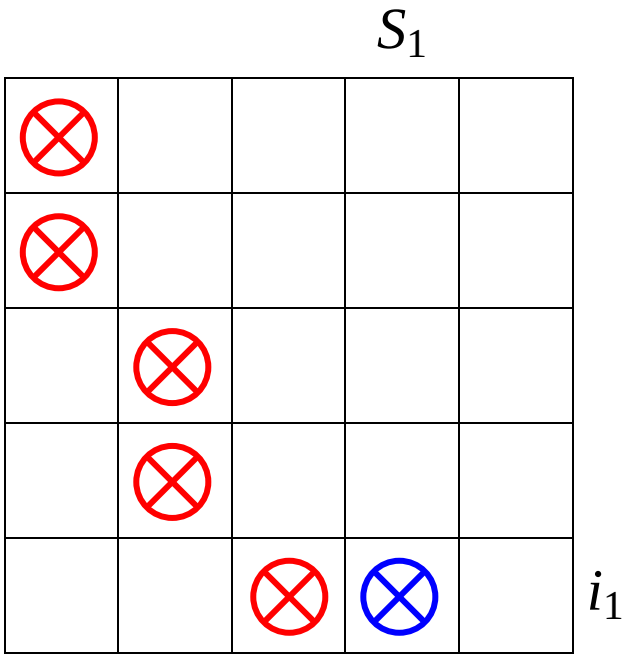
Общая идея алгоритма

Начинаем с $\Delta \equiv 0$. На каждом этапе алгоритма дефект уменьшается на 1, т.е. не более чем за n этапов найдем оптимальное решение задачи.

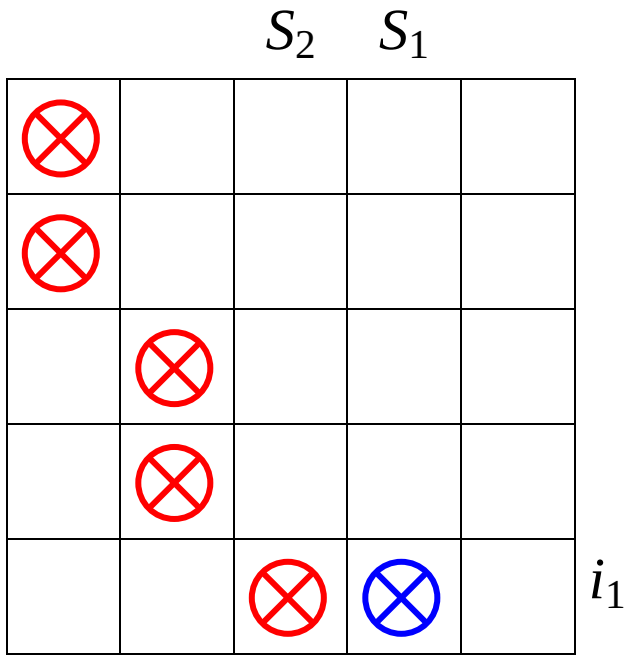
Описание одного этапа

1. Выберем столбец без Δ-основы и обозначим его S_1.

2. Увеличить Δ_{S_1} на максимальное δ так, чтобы все Δ-минимальные элементы остались Δ-минимальными (возможно $\delta = 0$). Получим для некоторой строки i_1 новый Δ-минимальный элемент $c_{i_1 S_1}$, назовем его альтернативной основой для строки i_1.

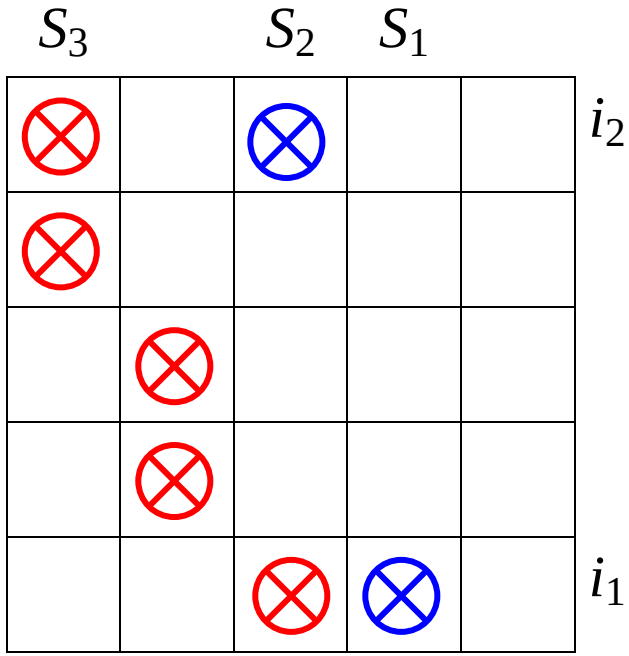


3. Для строки i_1 столбец $j(i_1)$ с Δ-основой пометим меткой S_2.



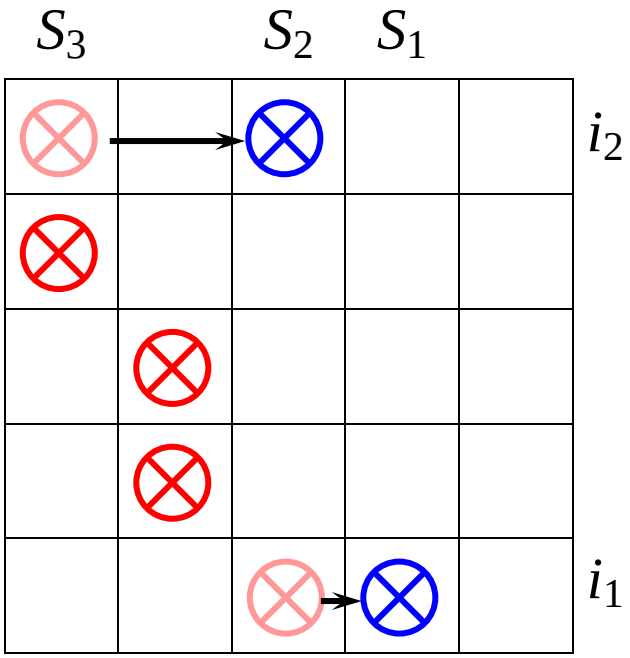
4. Увеличим Δ_{S_1} и Δ_{S_2} на максимальное δ так, чтобы все Δ-основы остались Δ-минимальными элементами.

Найдем новую альтернативную основу в одном из столбцов S_1 или S_2. Пусть она оказалась в строке i_2. Пометим столбец $j(i_2)$ меткой S_3 и будем продолжать этот процесс до тех пор пока не встретим столбец с двумя или более основами.

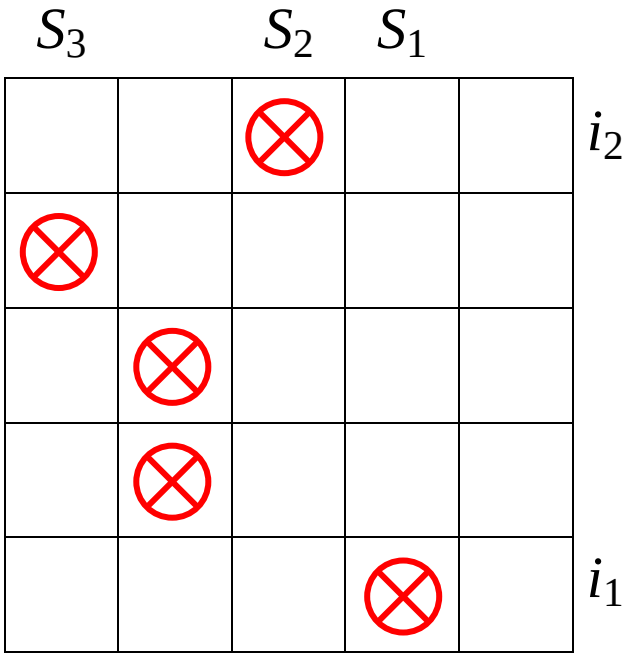


5. Строим новый набор из Δ-основ. Заменой основы в строке назовем следующую операцию: альтернативная основа становится основой, а старая перестает быть основой.

5.1. Произведем замену основ в строке, где лежит последняя альтернативная основа (строка i_k). Тогда в столбце $j(i_k)$ число основ уменьшится на 1, но останется положительным.



В столбце, где появилась новая основа, возьмем старую основу и в этой строке тоже проведем замену основ и т.д. до тех пор, пока не доберемся до столбца S_1. В итоге, столбец S_1 получит основу, а число основ в столбце $j(i_k)$ уменьшится на 1.



Упражнение. Оценить трудоемкость алгоритма решения задачи о назначениях.

Модификация алгоритма решения задачи о назначениях

Пусть S – множество номеров отмеченных столбцов,

R – множество номеров отмеченных строк без альтернативных основ

$$\delta = \min_{i \in R, k \in S} [(a_{ik} - \Delta_k) - (a_{ij(i)} - \Delta_{j(i)})]$$

Перепишем $\delta = \min_{k \in S} (\min_{i \in R} [(a_{ik} - \Delta_k) - (a_{ij(i)} - \Delta_{j(i)})])$

Модификация алгоритма, позволяющая быстро находить

$\min_{i \in R} [(a_{ik} - \Delta_k) - (a_{ij(i)} - \Delta_{j(i)})]$ и требующая $Cn^3 \lg n$ действий.

Пусть для записи упорядочения имеется массив из n ячеек.

В ячейке, соответствующей некоторому элементу массива, храним указатель на предыдущий и последующий (в смысле упорядочения) элементы, т.е. их номера.

При вычеркивании некоторого элемента массива меняется информация в ячейках для соседей этого элемента по упорядочению. Вместо указателей на вычеркиваемый элемент поставим указатели на его противоположного сосе-да.

Первоначальное упорядочение требует $Cn\lg n$ действий.

Для каждого $k \in S$ используем этот способ для быстрого нахождения $\min\limits_{i \in R}[(a_{ik} - \Delta_k) - (a_{ij(i)} - \Delta_{j(i)})]$.

Пусть $b_{ik} = [(a_{ik} - \Delta_k) - (a_{ij(i)} - \Delta_{j(i)})]$. Каждый из столбцов (b_{ik}) упорядочим и запишем это упорядочение по столбцам таблицы.

Отдельно выпишем строчку минимумов по столбцам равным $\min\limits_{i} b_{ik}$ в начале итерации.

В процессе проведения итерации должны вычеркивать из матрицы B строки, в которых появились альтернативные основы, с тем, чтобы минимум по столбцам матрицы был искомым $\min\limits_{i \in R} b_{ik}$.