

# Нижние оценки в задаче коммивояжера

**Примитивная оценка.** Плата за выезд  $a_i = \min_{j \neq i} c_{ij}$ ,  $i = 1, \dots, n$ .

Плата за въезд  $b_j = \min_{i \neq j} (c_{ij} - a_i)$   $j = 1, \dots, n$ .

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

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

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

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

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

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

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

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

Заменяя  $x_{ij} \in \{0, 1\}$  на  $0 \leq x_{ij} \leq 1$ , получаем задачу линейного программирования, которая 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 
$$\sum_{i=1}^n x_{ij} = 1, \quad 1 \leq j \leq n,$$

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

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

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

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

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

**Доказательство.** Решение  $(c_{1j(1)}, \dots, c_{nj(n)})$  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  $\Delta$  выделим в каждой строке по одному  $\Delta$ -минимальному элементу и назовем его  $\Delta$ -основой. Другие  $\Delta$ -минимальные элементы будем называть *альтернативными  $\Delta$ -основами*. Число столбцов матрицы  $c_{ij}$  без  $\Delta$ -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  $\Delta$ -основы и обозначим его  $S_1$ .







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

$S_1$

$i_1$

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








		$S_2$	$S_1$		
					
					
					
					
					$i_1$

4. Увеличим  $\Delta_{S_1}$  и  $\Delta_{S_2}$  на максимальное  $\delta$  так, чтобы все  $\Delta$ -основы остались  $\Delta$ -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 продол-

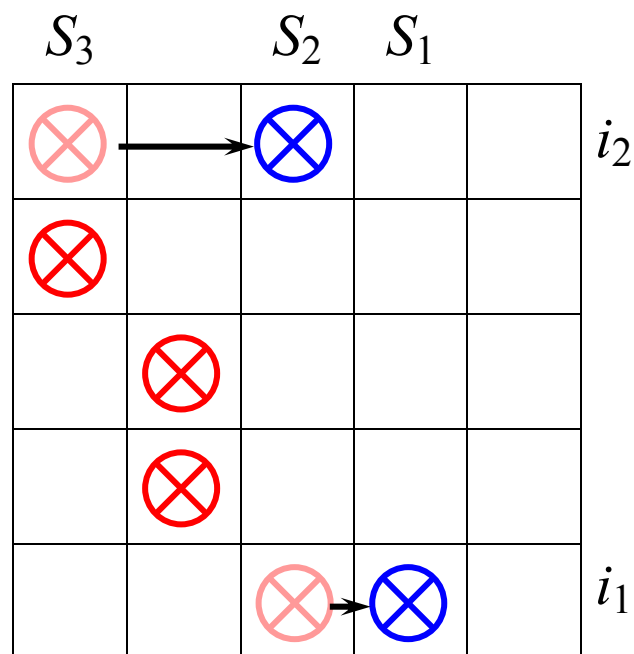


жать этот процесс до тех пор пока не встретим столбец с двумя или более основами.

	$S_3$		$S_2$	$S_1$	
					$i_2$
					
					
					
					$i_1$

5. Строим новый набор из  $\Delta$ -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 столб-

ца  $S_1$ . В итоге, столбец  $S_1$  получит основу, а число основ в столбце  $j(i_k)$  уменьшится на 1.

	$S_3$		$S_2$	$S_1$	
			⊗		$i_2$
⊗					
		⊗			
		⊗			
				⊗	$i_1$

**Упражнение.** Оценить трудоемкость алгоритма решения задачи о назначении. Придумать алгоритм решения задачи с трудоемкостью  $O(n^3)$ .

## Метод ветвей и границ

В основе метода лежит принцип «разделяй и властвуй».

Пусть  $D$  — множество допустимых решений задачи

$$\min \{f(x) \mid x \in D\},$$

и для любого подмножества  $d \subseteq D$  умеем вычислять:

$LB(d)$  — нижнюю оценку для минимума  $f(x)$ ,  $x \in d$ ,

$UB(d)$  — верхнюю оценку для минимума  $f(x)$ ,  $x \in d$ ,

т. е.

$$LB(d) \leq \min \{f(x) \mid x \in d\} \leq UB(d), \text{ для любого } d \subseteq D.$$

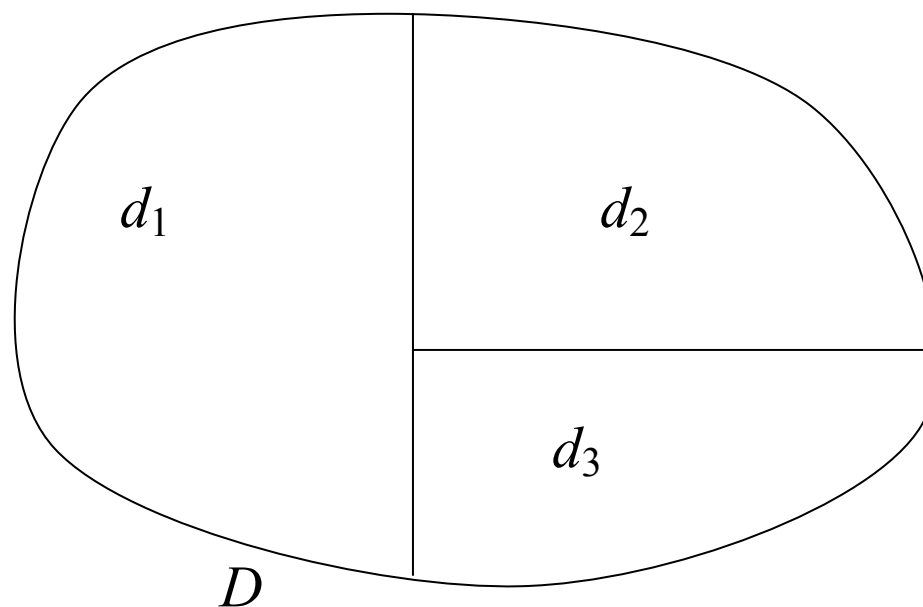
## Основная идея

Пусть  $x^*$  — текущий рекорд и сначала  $f(x^*) = UB(D)$ . Вычисляем  $LB(D)$  и, если  $LB(D) = UB(D)$ , то STOP,  $x^*$  — оптимальное решение задачи. В противном случае разбиваем  $D$  на подмножества  $D = d_1 \cup \dots \cup d_k$ . Для каждого подмножества вычисляем  $UB(d_i)$ ,  $LB(d_i)$ ,  $i = 1, \dots, k$ .

Если  $f(x^*) > UB(d_i)$ , то меняем рекорд.

Если  $LB(d_i) \geq f(x^*)$ , то выбрасываем  $d_i$ , иначе дробим  $d_i$  на подмножества.

Так как  $D$  — конечное множество, то процесс конечен и дает точное решение задачи.



## Описание метода

На каждом шаге имеется

- рекорд  $x^*$  ;
- просмотренная часть  $P \subset D$ , для которой  $f(x) \geq f(x^*)$ ,  $x \in P$ ;
- разбиение множества  $D \setminus P$  на подмножества  $d_{i_1}, d_{i_2}, \dots, d_{i_k}$ .

**Шаг состоит в следующем.**

1. Выбрать элемент разбиения, например,  $d_{i_k}$  ;
2. Вычислить  $UB(d_{i_k})$ . Если  $f(x^*) > UB(d_{i_k})$ , то сменить рекорд  $x^*$ .
3. Вычислить  $LB(d_{i_k})$ .

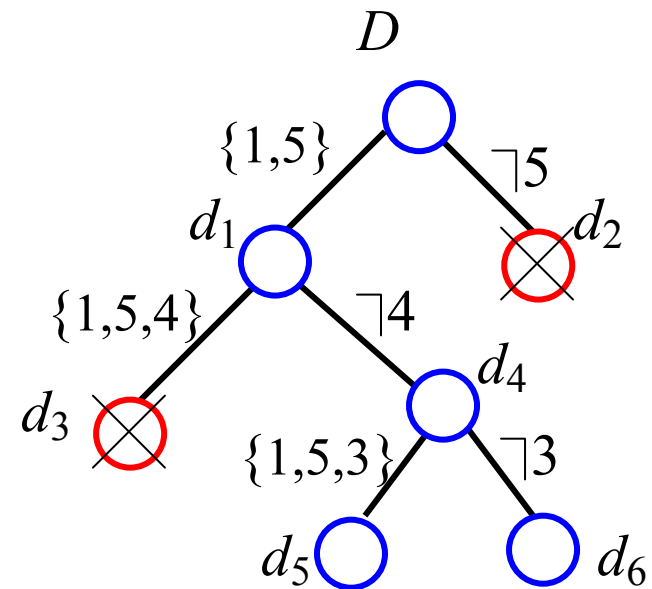
- 3.1. Если  $LB(d_{i_k}) \geq f(x^*)$ , то добавить  $d_{i_k}$  к  $P$  и перейти к следующему шагу.
- 3.2. Если  $LB(d_{i_k}) \leq f(x^*)$ , но в множестве  $d_{i_k}$  удалось найти наилучший элемент  $\tilde{x}$ :  $f(\tilde{x}) = \min\{f(x) \mid x \in d_{i_k}\}$ , то добавить  $d_{i_k}$  к  $P$ ; если  $f(x^*) > f(\tilde{x})$ , то положить  $x^* := \tilde{x}$ .
- 3.3. Если  $LB(d_{i_k}) \leq f(x^*)$ , но элемент  $\tilde{x}$  найти не удалось, то разбиваем  $d_{i_k}$  на подмножества  $d_{i_k} = d_{i_{k+1}} \cup \dots \cup d_{i_{k+m}}$  и переходим к следующему шагу, имея новое разбиение для  $D \setminus P$ .

# Метод В&Г для задачи коммивояжера

Разбиение множества  $D$  представляется в виде бинарного дерева.

Каждой вершине дерева соответствует частичный тур и список запретов.

Например, вершине  $d_6$  соответствует частичный тур 1,5 и запреты  $\{4,3\}$  на выход из города 5.





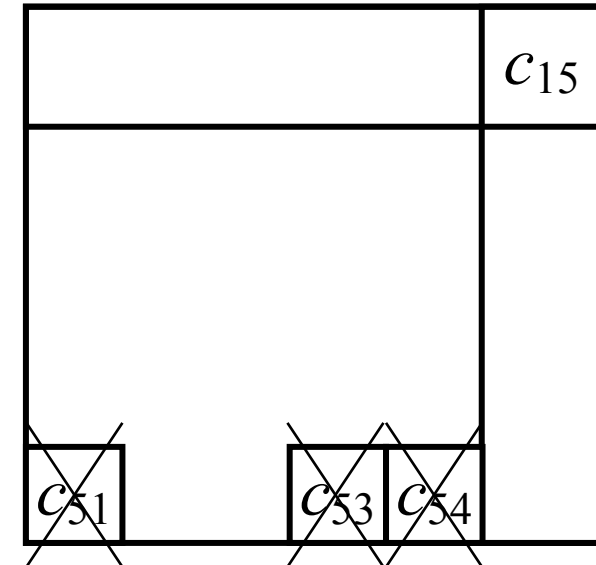
## Метод В&Г для задачи коммивояжера

Примитивная нижняя оценка для вершины дерева,  
например,  $d_6$  при  $n = 5$ :

$$LB(d_6) = c_{15} + \sum_{i=2}^5 a_i + \sum_{j=1}^4 b_j.$$

Задача о назначениях:

$$LB(d_6) = c_{15} + \sum_{i=2}^5 c_{ij(i)}, \text{ при } c_{53} = c_{54} = c_{51} = +\infty.$$

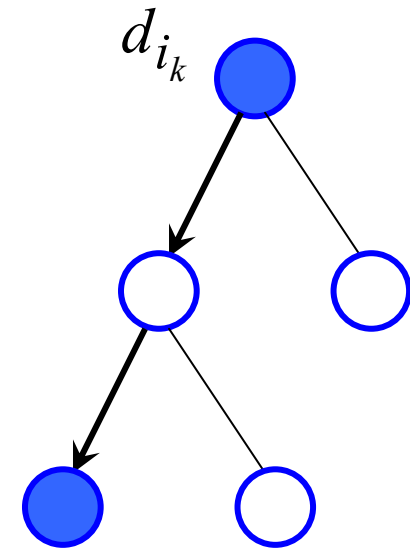


Верхняя оценка — алгоритм «Иди в ближайший».

## Выбор переменной для ветвления

Основная идея — угадать оптимальное решение на подмножестве  $d_{i_k}$  и ветвиться по дугам этого тура:

- для частичного тура  $i_1, \dots, i_k$  выбираем минимальный элемент в строке  $i_k$  матрицы  $c''_{ij} = c_{ij} - a_i - b_j, j \neq i_1, \dots, i_k$
- для частичного тура  $i_1, \dots, i_k$  строим верхнюю оценку и ветвимся по дуге  $(i_1, \dots, i_{k+1})$ .
- для частичного тура  $i_1, \dots, i_k$  решаем задачу о назначениях и ветвимся вдоль цикла, проходящего через вершину  $i_k$ .



# Выбор подмножества из разбиения $D \setminus P$

Две основные схемы:

- **многосторонняя схема ветвления**, когда выбирается подмножество  $d'$  такое, что

$$LB(d') = \min \{LB(d_i) \mid i = i_1, \dots, i_k\}.$$

Среди элементов разбиения  $D \setminus P = d_{i_1} \cup \dots \cup d_{i_k}$  выбирается подмножество с наименьшей нижней границей.

- **односторонняя схема ветвления**, когда всегда выбираем последний элемент  $d' = d_{i_k}$ .

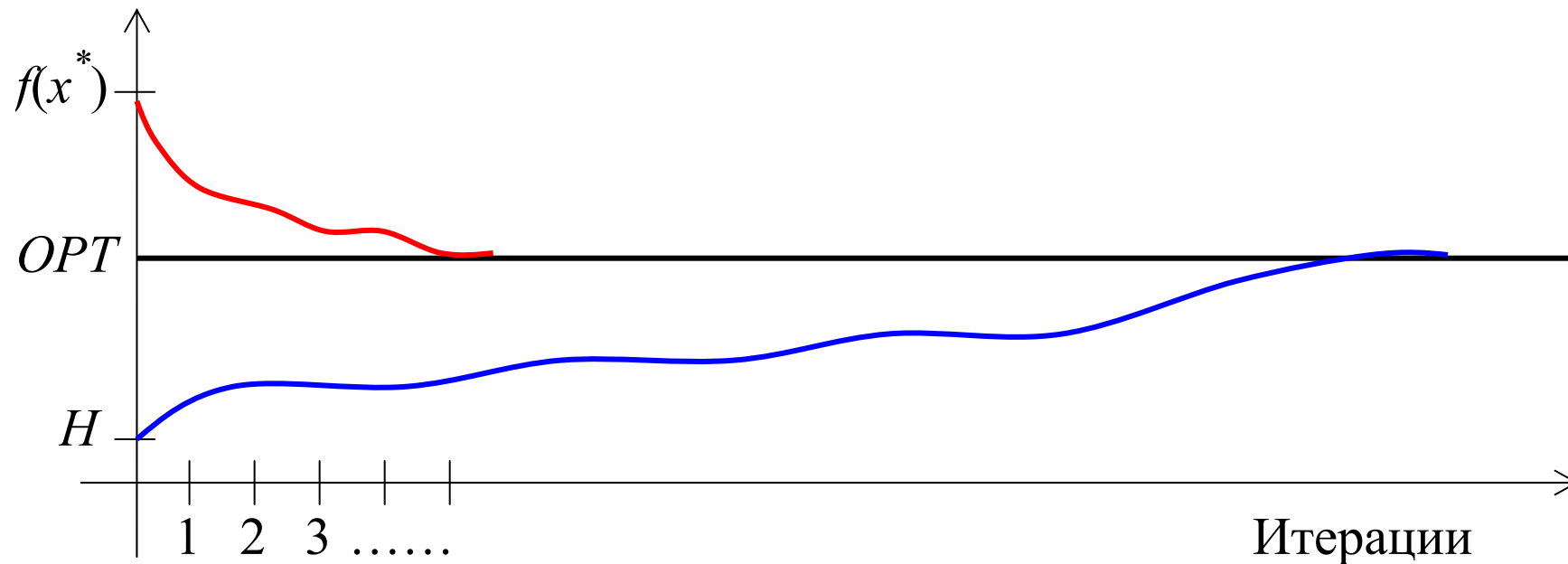
Первая схема требует много оперативной памяти, но в среднем просматривает меньше вершин, чем вторая. Возможна комбинация этих схем: сначала первая, пока хватает памяти, затем вторая.

# Влияние основных элементов на трудоемкость

Верхняя оценка  $UB$

Нижняя оценка  $LB$

Схема ветвления и выбор переменной для ветвления



$$H = \min_{i_1, \dots, i_k} LB(d_i)$$

# Задача коммивояжера в Интернет

- TSPBIB Home Page

[http://www.ing.unlp.edu.ar/cetad/mos/TSPBIB\\_home.html](http://www.ing.unlp.edu.ar/cetad/mos/TSPBIB_home.html)

- The Hamiltonian Page: Hamiltonian cycle and path problems, their generalizations and variations

<http://www.ing.unlp.edu.ar/cetad/mos/Hamilton.html>

- Fractal Instances of the Traveling Salesman Problem

[http://www.ing.unlp.edu.ar/cetad/mos/FRACTAL\\_TSP\\_home.html](http://www.ing.unlp.edu.ar/cetad/mos/FRACTAL_TSP_home.html)

- **DIMACS:** The Traveling Salesman Problem

<http://www.research.att.com/~dsj/chtsp/>

## Задача маршрутизации

**Дано:** множество клиентов  $J = \{2, 3, \dots, n\}$ ,  
 $j = 1$  — гараж,  
множество грузовиков  $K = \{1, \dots, m\}$   
 $Q_k$  — грузоподъемность  $k$ -го грузовика  
 $q_j$  — вес груза для  $j$ -го клиента  
 $c_{ij}$  — расстояние между клиентами  $i$  и  $j$ .

**Найти:** набор циклов, начинающихся и заканчивающихся в гараже  $j = 1$  (по одному циклу на транспортное средство) такой, что суммарная длина циклов была бы минимальной и позволила бы обслужить всех клиентов данным набором транспортных средств.

# Математическая модель

## Переменные задачи

$$x_{ijk} = \begin{cases} 1, & \text{если транспорт } k \text{ посещает клиента } j \text{ сразу после клиента } i, \\ 0 & \text{в противном случае} \end{cases}$$

$$y_{ik} = \begin{cases} 1, & \text{если транспорт } k \text{ посещает клиента } i, \\ 0 & \text{в противном случае} \end{cases}$$

## Модель

$$\min \sum_{ij \in J} c_{ij} \sum_{k \in K} x_{ijk}$$

при ограничениях

$$\sum_{k \in K} y_{ik} = \begin{cases} 1, & i = 2, \dots, n \\ m, & i = 1 \end{cases}$$

$$\sum_{i \in J} y_{ik} q_i \leq Q_k, \quad k \in K$$

$$\sum_j x_{ijk} = \sum_j x_{jik} = y_{ik}, \quad i \in J, k \in K$$

$$\sum_{ij \in S} x_{ijk} \leq |S| - 1, \text{ для всех } S \subseteq \{2, \dots, n\}, \quad k \in K$$

$$y_{ik} \in \{0, 1\}, \quad x_{ijk} \in \{0, 1\}, \quad i, j \in J, \quad k \in K.$$



## Возможные обобщения модели

1. Каждый клиент имеет «временное окно», в течение которого транспортное средство должно его посетить.
2. Клиенты могут не только получать продукцию, но и отправлять ее в гараж.
3. Обслуживание клиентов происходит не мгновенно, а в течение определенного времени: разгрузка, загрузка, оформление документов и т.п.
4. Транспортное средство может несколько раз вернуться в гараж (пройти несколько циклов).
5. Транспортное средство имеет временное окно (рабочий день водителя) для обслуживания клиентов.

## Целевые функции

- Суммарное расстояние или расход бензина (каждое транспортное средство имеет свой расход на 100 км).
- Зарплата водителей и эксплуатационные расходы на транспортные средства.
- Оптимизация парка транспортных средств (расширить или сузить, заменить тяжелые на легкие или наоборот, ...).
- Оптимальное размещение гаражей, их количество и вместимость.

Системы поддержки решений по оперативному управлению транспортными средствами, замена водителей, форс-мажорные обстоятельства, изменения потребностей клиентов,....