4EK605 Combinatorial Optimization

Jan Fábry

Faculty of Informatics and Statistics
Department of Econometrics

fabry@vse.cz
https://janfabry.cz

February 18, 2022, Prague

Course Syllabus

1. Integer Programming Problem
2. IP and MIP Modelling
3. Graph Modelling
   - Flow Problems
   - Routing Problems
4. Formulations in Logical Variables
5. Polyhedral Theory
6. Solving Problems - Methods & Algorithms
   - Relaxation
   - Exact Methods
   - Computational Complexity
   - Heuristics & Metaheuristics

Course Syllabus

1. Integer Programming Problem
2. IP and MIP Modelling
3. Graph Modelling
   - Flow Problems
   - Routing Problems
4. Formulations in Logical Variables
5. Polyhedral Theory
6. Solving Problems - Methods & Algorithms
   - Relaxation
   - Exact Methods
   - Computational Complexity
   - Heuristics & Metaheuristics

Integer Programming Problem

General linear programming problem (LP):

$$z_{\mathrm{LP}} = \max\{c^T x : Ax \leq b, x \in \mathbb{R}^n_+\}. \quad (1)$$

Integer programming problem (IP):

$$z_{\mathrm{IP}} = \max\{c^T x : Ax \leq b, x \in \mathbb{Z}^n_+\}. \quad (2)$$

$z_{\mathrm{LP}} \geq z_{\mathrm{IP}}$ since $\mathbb{Z}^n_+ \subset \mathbb{R}^n_+$
$P = \{x : Ax \leq b, x \in \mathbb{R}^n_+\}, S = \{x : Ax \leq b, x \in \mathbb{Z}^n_+\}, S \subset P$

Mixed integer programming problem (MIP):

$$z_{\mathrm{MIP}} = \max\{c^T x + h^T y : Ax + Gy \leq b, x \in \mathbb{R}^n_+, y \in \mathbb{Z}^p_+\}. \quad (3)$$

Binary integer programming problem (BIP):

$$z_{\mathrm{BIP}} = \max\{c^T x : Ax \leq b, x \in \mathbb{B}^n\}, \mathbb{B} = \{0, 1\}. \quad (4)$$

Course Syllabus

1. Production Planning Problem

Variables: x_i is a number of pcs. of i-th product

Constraints: $x_1, x_2, \ldots, x_n$ are integers

2. Cutting Stock Problem

Variables: x_i is a number of pcs. of raw products being cut according to i-th cutting pattern

Constraints: $x_1, x_2, \ldots, x_n$ are integers

The objective:

- Minimization of pcs. of cut raw products
- Minimization of total waste
- Maximization of pcs. of assembled products (profit)

3. (0-1) Knapsack Problem

Definition: Budget b is available for investments in n considered projects, where a_j is the outlay for project j and c_j is its expected return. The objective is to choose a set of projects to maximize the total expected return while not exceeding the budget.

Variables:

$$x_j = \begin{cases} 1 & \text{if the project } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Model:

$$\max \sum_{j=1}^{n} c_j x_j \tag{6}$$

$$\sum_{j=1}^{n} a_j x_j \leq b \tag{7}$$

$$x_j \in \{0,1\} \quad \text{for } j = 1, 2, \ldots, n \tag{8}$$

4. Perfect Matching Problem

Definition: On a trip, n (even number) students are to be assigned to double rooms. Satisfaction value c_{ij} is given for potential roommates i and j. The objective is to assign students to maximize the total satisfaction of the group.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if students } i \text{ and } j \text{ are roommates} \\ 0 & \text{otherwise} \end{cases} \qquad i < j \tag{9}$$

Model:

$$\max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij} \tag{10}$$

$$\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = 1 \quad \text{for } i = 1, 2, \ldots, n \tag{11}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } \begin{matrix} i = 1, 2, \ldots, n-1 \\ j = i+1, i+2, \ldots, n \end{matrix} \tag{12}$$

5. Generalized Assignment Problem

Definition: Let us assume m stations taking petrol from n terminals. Each station i can take petrol exactly from one terminal and its requirement a_i is given. Capacity of terminal j is denoted by b_j. If station i takes petrol from terminal j then cost c_{ij} is calculated. The objective is to minimize the total cost.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if station } i \text{ takes petrol from terminal } j \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

5. Generalized Assignment Problem

Model:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{14}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, 2, \ldots, m \tag{15}$$

$$\sum_{i=1}^{m} a_i x_{ij} \leq b_j \quad \text{for } j = 1, 2, \ldots, n \tag{16}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, m \\ j = 1, 2, \ldots, n \end{array} \tag{17}$$

6. Linear Assignment Problem

Definition: There are n people available to carry out n jobs. Each person is assigned to carry out exactly one job. Some individuals are better suited to particular jobs than others, so there is an estimated cost c_{ij} if person i is assigned to job j. The objective is to find a minimum cost assignment.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ does job } j \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

6. Linear Assignment Problem

Model:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{19}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, 2, \ldots, n \tag{20}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1, 2, \ldots, n \tag{21}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 1, 2, \ldots, n \end{array} \tag{22}$$

7. Bottleneck Assignment Problem

Definition: Let n jobs and n parallel machines be given. The coefficient c_{ij} is the time needed for machine j to complete job i. The objective is to minimize the latest completion time.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if job } i \text{ is assigned to machine } j \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

$$T = \text{latest completion time} \tag{24}$$

7. Bottleneck Assignment Problem

Model:

$$\min T \tag{25}$$

$$c_{ij}x_{ij} \leq T \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 1, 2, \ldots, n \end{array} \tag{26}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, 2, \ldots, n \tag{27}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1, 2, \ldots, n \tag{28}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 1, 2, \ldots, n \end{array} \tag{29}$$

8. Quadratic Assignment Problem

Definition: A set of n facilities has to be allocated to a set of n locations. The coefficient c_{ij} is the flow from facility i to facility j and the value d_{kl} is the distance from location k to location l. The objective is to allocate each facility to a location such that the total cost is minimized.

Variables:

$$x_{ik} = \begin{cases} 1 & \text{if facility } i \text{ is assigned to location } k \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

8. Quadratic Assignment Problem

Model:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} c_{ij} d_{kl} x_{ik} x_{jl} \tag{31}$$

$$\sum_{k=1}^{n} x_{ik} = 1 \quad \text{for } i = 1, 2, \ldots, n \tag{32}$$

$$\sum_{i=1}^{n} x_{ik} = 1 \quad \text{for } k = 1, 2, \ldots, n \tag{33}$$

$$x_{ik} \in \{0, 1\} \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ k = 1, 2, \ldots, n \end{array} \tag{34}$$

8. Quadratic Assignment Problem

Linearization of the objective function:

$$y_{ijkl} = \begin{cases} 1 & \text{if facility } i \text{ is assigned to location } k \\ & \text{and facility } j \text{ is assigned to location } l \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

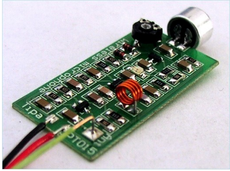
$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} c_{ij} d_{kl} y_{ijkl} \tag{36}$$

$$y_{ijkl} \geq x_{ik} + x_{jl} - 1 \quad \text{for } i, j, k, l = 1, 2, \ldots, n \tag{37}$$

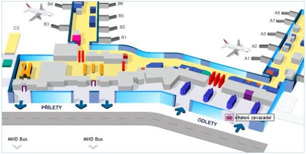
8. Quadratic Assignment Problem

Applications:

- Placement Problem



- The Airport Gate Assignment Problem



9. Set-Covering, Set-Packing and Set-Partitioning Problems

Definition: Let $M = \{1, 2, \dots, m\}$ be a finite set of tasks and $N = \{1, 2, \dots, n\}$ a finite set of their providers. Incidence matrix A is given with values $a_{ij} = 1$ if provider j is able to cover task i, $a_{ij} = 0$ otherwise. If j-th provider is selected, cost c_j is calculated. The objective is to cover all tasks with the minimal total cost.

Let $M_j \subseteq M$ be a set of tasks that provider $j \in N$ is able to cover.

We say that

- $F \subseteq N$ covers M if $\bigcup_{j \in F} M_j = M$
- $F \subseteq N$ is a packing with respect to M if $M_j \cap M_k = \emptyset$ for all $j, k \in F, j \neq k$
- $F \subseteq N$ is a partition of M if F is both a covering and a packing

9. Set-Covering, Set-Packing and Set-Partitioning Problems

Variables:

$$x_j = \begin{cases} 1 & \text{if provider } j \text{ is selected, i.e. } j \in F \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

Model:

$$\min \sum_{j=1}^{n} c_j x_j \tag{39}$$

$$\begin{aligned} &\text{(set-covering)} && \sum_{j=1}^{n} a_{ij} x_j \geq 1 && \text{for } i = 1, 2, \ldots, m \\ &\text{(set-packing)} && \sum_{j=1}^{n} a_{ij} x_j \leq 1 && \text{for } i = 1, 2, \ldots, m \\ &\text{(set-partitioning)} && \sum_{j=1}^{n} a_{ij} x_j = 1 && \text{for } i = 1, 2, \ldots, m \end{aligned} \tag{40}$$

$$x_j \in \{0, 1\} \quad \text{for } j = 1, 2, \ldots, n \tag{41}$$

9. Set-Covering, Set-Packing and Set-Partitioning Problems

Applications:

- Let $N = \{1, 2, \ldots, n\}$ be a set of potential sites for the location of fire stations. A station placed at j costs c_j. Let $M = \{1, 2, \ldots, m\}$ be a set of communities that have to be protected. The subset of communities that can be protected from j (e.g. reached from the fire station in 10 minutes) is M_j.
- Assigning airline crews to flights.
- Scheduling workers to shifts.

10. Facility Location Problem

Definition: A set of potential depots $M = \{1, 2, \ldots, m\}$ and a set of clients $N = \{1, 2, \ldots, n\}$ are given. Suppose a facility located at i has a capacity of a_i and the j-th client has demand b_j. Fixed cost f_i is associated with the use of depot i and transportation cost c_{ij} is charged for shipping unit between location i and client j. The objective is to decide which depots to open and what quantity to transport between locations and clients such that the total cost is minimized.

Variables:

$$x_i = \begin{cases} 1 & \text{if depot at location } i \text{ is open} \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

$$y_{ij} = \text{quantity transported from location } i \text{ to client } j \tag{43}$$

10. Facility Location Problem

Model:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} y_{ij} + \sum_{i=1}^{m} f_i x_i \tag{44}$$

$$\sum_{j=1}^{n} y_{ij} \leq a_i x_i \quad \text{for } i = 1, 2, \ldots, m \tag{45}$$

$$\sum_{i=1}^{m} y_{ij} = b_j \quad \text{for } j = 1, 2, \ldots, n \tag{46}$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, 2, \ldots, m \tag{47}$$

$$y_{ij} \in \mathbb{R}_+ \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, m \\ j = 1, 2, \ldots, n \end{array} \tag{48}$$

11. Fixed-Cost Production Planning Problem

Definition: Suppose the possible production of n products on n production lines (each product on exactly one PL). Fixed cost f_i has to be considered if PL i is used (i.e. product i is produced). Unit profit c_i is given for product i. Standard production planning (capacity) constraints are defined. The objective is to maximize total profit decreased by fixed cost.

Variables:

$$x_i = \begin{cases} 1 & \text{if product } i \text{ is produced (on PL } i\text{)} \\ 0 & \text{otherwise} \end{cases} \tag{49}$$

$$y_i = \text{quantity of product } i \text{ being produced} \tag{50}$$

11. Fixed-Cost Production Planning Problem

Model:

$$\max \sum_{i=1}^{n} c_i y_i - \sum_{i=1}^{n} f_i x_i \tag{51}$$

$$\text{(capacity constraints)} \quad \sum_{i=1}^{n} a_{li} y_i \le b_l \quad \text{for } l = 1, 2, \ldots, m \tag{52}$$

$$y_i \le M x_i \quad \text{for } i = 1, 2, \ldots, n \tag{53}$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, 2, \ldots, n \tag{54}$$

$$y_i \in \mathbb{R}_+ \quad \text{for } i = 1, 2, \ldots, n \tag{55}$$

M = big number

12. Container Transportation Problem

Definition: Goods are directly transported from m sources to n destinations. Supply a_i of source i and demand b_j of destination j are given. Containers of capacity K are used for transport and shipping cost c_{ij} is known for the transport of a container from source i to destination j. The objective is to satisfy all demands at the minimum total shipping cost.

Variables:

$$y_{ij} = \text{quantity of goods transported from source } i \text{ to destination } j \tag{56}$$

$$x_{ij} = \text{number of containers used for the transport of goods from source } i \text{ to destination } j \tag{57}$$

12. Container Transportation Problem

Model:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{58}$$

$$\sum_{j=1}^{n} y_{ij} \leq a_i \quad \text{for } i = 1, 2, \ldots, m \tag{59}$$

$$\sum_{i=1}^{m} y_{ij} = b_j \quad \text{for } j = 1, 2, \ldots, n \tag{60}$$

$$y_{ij} \leq K x_{ij} \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, m \\ j = 1, 2, \ldots, n \end{array} \tag{61}$$

$$y_{ij} \in \mathbb{R}_+ \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, m \\ j = 1, 2, \ldots, n \end{array} \tag{62}$$

$$x_{ij} \in \mathbb{Z}_+ \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, m \\ j = 1, 2, \ldots, n \end{array} \tag{63}$$

13. Bin Packing Problem

Definition 1: Suppose a set of n items that can be packed into m containers. The weight w_j and value c_j of item j are given. Let K_i be a weight capacity of container i. The objective is to maximize the total value of all assigned items.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if item } j \text{ is assigned to container } i \\ 0 & \text{otherwise} \end{cases} \tag{64}$$

13. Bin Packing Problem

Model:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} c_j x_{ij} \tag{65}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \quad \text{for } j = 1, 2, \ldots, n \tag{66}$$

$$\sum_{j=1}^{n} w_j x_{ij} \leq K_i \quad \text{for } i = 1, 2, \ldots, m \tag{67}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, m \\ j = 1, 2, \ldots, n \end{array} \tag{68}$$

13. Bin Packing Problem

Definition 2: Suppose a set of n types of items that have to be transported using m containers of identical weight capacity K. Let w_j be a weight of item type j and r_j be a number of them to be transported. The objective is to minimize a number of containers used to transport all items.

Variables:

$$x_i = \begin{cases} 1 & \text{if container } i \text{ is used} \\ 0 & \text{otherwise} \end{cases} \tag{69}$$

$$y_{ij} = \text{a number of items of type } j \text{ being transported in container } i \tag{70}$$

13. Bin Packing Problem

Model:

$$\min \sum_{i=1}^{m} x_i \tag{71}$$

$$\sum_{i=1}^{m} y_{ij} = r_j \quad \text{for } j = 1, 2, \ldots, n \tag{72}$$

$$\sum_{j=1}^{n} w_j y_{ij} \leq K x_i \quad \text{for } i = 1, 2, \ldots, m \tag{73}$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, 2, \ldots, m \tag{74}$$

$$y_{ij} \in \mathbb{Z}_+ \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, m \\ j = 1, 2, \ldots, n \end{array} \tag{75}$$

Course Syllabus

Introduction to Graph Theory

Graph is a set $G = \{V, E\}$, where V is a set of vertices (nodes) and E is a set of edges (arcs).

Undirected arc is a set of two vertices $\{i, j\}$.

Directed arc is an ordered pair of two vertices (i, j).

In undirected graph all arcs are undirected.

In directed graph (digraph) all arcs are directed.

Mixed graph contains both undirected and directed arcs.

Two nodes that are contained in an arc are adjacent.

Two arcs that share a node are adjacent.

An arc and a node contained in that arc are incident.

Degree of a node (in undirected graph) is a number of incident arcs.

In-degree of a node (in directed graph) is a number of incident arcs in which the node is the terminal one.

Out-degree of a node (in directed graph) is a number of incident arcs in which the node is the initial one.

Introduction to Graph Theory

Walk from node i to node j is a sequence of nodes and arcs, where i is the initial node and j is the terminal node (nodes and arcs may be repeated).

Trail is a walk with no repeated arc.

Path is a trail with no repeated node.

Cycle is closed walk (the initial node is the terminal one).

In directed path (in directed graph) a direction of all arcs is respected.

In undirected path (in directed graph) a direction of all arcs may not be respected.

Undirected graph is connected if between each pair of nodes there is a path.

Directed graph is connected if there is a directed or undirected path between each pair of nodes.

Graph Modelling

Introduction to Graph Theory

Directed graph is strongly connected if there is a directed path between each pair of nodes.

Undirected graph is complete if there is an arc between each pair of nodes.

Tree is a connected undirected graph with no cycles.

Subgraph of graph $G = \{V, E\}$ is a graph $G' = \{V', E'\}$, where $V' \subseteq V$ and $E' \subseteq E$.

Spanning tree of the graph G is a subgraph G', where $V' = V$ and which is a tree.

Valued graph has numbers associated with nodes or/and arcs.

Hamiltonian cycle is a cycle that includes each node of the graph exactly once.

Eulerian cycle includes each arc of the graph exactly once.

Eulerian trail is a trail that includes each arc of the graph.

Eulerian graph is a graph in which the Eulerian cycle can be found.

Course Syllabus

1. Maximum Flow Problem

Definition: Let $G = \{V, E\}$ be a digraph with the flow capacity k_{ij} given for each arc (i, j). The objective is to identify the maximum amount of flow that can occur from source node s to sink node d.

Variables:

$$x_{ij} = \text{flow from node } i \text{ to node } j \tag{76}$$

$$F = \text{total flow} \tag{77}$$

Model:

$$\max F \tag{78}$$

$$\sum_{\substack{j \in V \\ (i,j) \in E}} x_{ij} - \sum_{\substack{j \in V \\ (j,i) \in E}} x_{ji} = \begin{cases} F & \text{for } i = s \\ 0 & \text{for } i \in V \setminus \{s, d\} \\ -F & \text{for } i = d \end{cases} \tag{79}$$

$$x_{ij} \leq k_{ij} \quad \text{for } (i, j) \in E \tag{80}$$

$$x_{ij} \in \mathbb{R}_+ \quad \text{for } (i, j) \in E \tag{81}$$

$$F \in \mathbb{R}_+ \tag{82}$$

1. Maximum Flow Problem

Alternative approach to modelling: Adding a dummy backward arc from d to s with the capacity $k_{ds} = M$ (big number).

Variables:

$$x_{ij} = \text{flow from node } i \text{ to node } j \tag{83}$$

Model:

$$\max\ x_{ds} \tag{84}$$

$$\sum_{\substack{j \in V \\ (i,j) \in E}} x_{ij} - \sum_{\substack{j \in V \\ (j,i) \in E}} x_{ji} = 0 \quad \text{for } i \in V \tag{85}$$

$$x_{ij} \leq k_{ij} \quad \text{for } (i,j) \in E \tag{86}$$

$$x_{ij} \in \mathbb{R}_+ \quad \text{for } (i,j) \in E \tag{87}$$

2. Minimum-Cost Flow Problem

Definition: Let $G = \{V, E\}$ be a digraph with the flow capacity k_{ij} and unit cost c_{ij} given for each arc (i, j). The objective is to satisfy required total flow F_0 (from source node s to sink node d) with the minimum total cost.

Variables:
$$x_{ij} = \text{flow from node } i \text{ to node } j \tag{88}$$

Model:
$$\min \sum_{\substack{i \in V \\ (i,j) \in E}} \sum_{j \in V} c_{ij} x_{ij} \tag{89}$$

$$\sum_{\substack{j \in V \\ (i,j) \in E}} x_{ij} - \sum_{\substack{j \in V \\ (j,i) \in E}} x_{ji} = \begin{cases} F_0 & \text{for } i = s \\ 0 & \text{for } i \in V \setminus \{s, d\} \\ -F_0 & \text{for } i = d \end{cases} \tag{90}$$

$$x_{ij} \leq k_{ij} \quad \text{for } (i, j) \in E \tag{91}$$

$$x_{ij} \in \mathbb{R}_+ \quad \text{for } (i, j) \in E \tag{92}$$

3. Maximum Flow Cost-Limited Problem

Definition: Let $G = \{V, E\}$ be a digraph with the flow capacity k_{ij} and unit cost c_{ij} given for each arc (i, j). The objective is to identify the maximum amount of flow that can occur from source node s to sink node d with respect to limited total cost C_0.

Variables:

$$x_{ij} = \text{flow from node } i \text{ to node } j \tag{93}$$

$$F = \text{total flow} \tag{94}$$

Model:

$$\max F \tag{95}$$

$$\sum_{i \in V} \sum_{\substack{j \in V \\ (i,j) \in E}} c_{ij} x_{ij} \leq C_0 \tag{96}$$

and constraints (79) - (82)

4. Fixed-Charge Flow Problem

Definition: Let $G = \{V, E\}$ be a digraph with the flow capacity k_{ij} given for each arc (i, j). Using arc (i, j) is charged c_{ij}. The objective is to satisfy required total flow F_0 (from source node s to sink node d) with the minimum total cost.

Variables:

$$x_{ij} = \text{flow from node } i \text{ to node } j \tag{97}$$

$$y_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is used} \\ 0 & \text{otherwise} \end{cases} \tag{98}$$

4. Fixed-Charge Flow Problem

Model:

$$\min \sum_{i \in V} \sum_{\substack{j \in V \\ (i,j) \in E}} c_{ij} y_{ij} \tag{99}$$

$$\sum_{\substack{j \in V \\ (i,j) \in E}} x_{ij} - \sum_{\substack{j \in V \\ (j,i) \in E}} x_{ji} = \begin{cases} F_0 & \text{for } i = s \\ 0 & \text{for } i \in V \setminus \{s, d\} \\ -F_0 & \text{for } i = d \end{cases} \tag{100}$$

$$x_{ij} \leq k_{ij} y_{ij} \quad \text{for } (i,j) \in E \tag{101}$$

$$y_{ij} \in \{0,1\} \quad \text{for } (i,j) \in E \tag{102}$$

5. Multi-Commodity Flow Problem

Definition: Let $G = \{V, E\}$ be a digraph and Q be a set of commodities that have to be transported from source node s to sink node d. For each commodity $q \in Q$, the required quantity F_0^q is given. Total flow of all commodities through arc (i, j) should not exceed its capacity k_{ij}. Unit cost c_{ij}^q is defined for the flow of commodity q through arc (i, j). The objective is to transport required amounts of all commodities with the minimum total cost.

Variables:

$$x_{ij}^q = \text{quantity of commodity } q \text{ transported from node } i \text{ to node } j \tag{103}$$

5. Multi-Commodity Flow Problem

Model:

$$\min \sum_{q\in Q}\sum_{\substack{i\in V\\(i,j)\in E}}\sum_{j\in V} c_{ij}^q x_{ij}^q \tag{104}$$

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij}^q - \sum_{\substack{j\in V\\(j,i)\in E}} x_{ji}^q = \begin{cases} F_0^q & \text{for } i = s, q \in Q \\ 0 & \text{for } i \in V \setminus \{s,d\}, q \in Q \\ -F_0^q & \text{for } i = d, q \in Q \end{cases} \tag{105}$$

$$\sum_{q\in Q} x_{ij}^q \le k_{ij} \quad \text{for } (i,j) \in E \tag{106}$$

$$x_{ij}^q \in \mathbb{R}_+ \quad \text{for } (i,j) \in E, q \in Q \tag{107}$$

6. Transshipment Problem

Definition: Let $G = \{V, E\}$ be a digraph with three sets of nodes: set of sources V_s, set of destinations V_d and set of transshipment nodes V_t. Let $a_i > 0$ be a supply of the product in source node $i \in V_s$ and $a_i < 0$ be a demand for the product in destination $i \in V_d$. For each transshipment node $i \in V_t$ it is valid $a_i = 0$. Flow capacity k_{ij} and unit cost c_{ij} are given for each arc (i, j). Demand in all destinations has to be satisfied without exceeding any supply. The objective is to minimize total flow cost. Suppose total demand is equal to total supply.

Assumptions:

$$V = V_s \cup V_d \cup V_t \quad \text{and} \quad V_s \cap V_d \cap V_t = \emptyset \tag{108}$$

$$\sum_{i \in V_s} a_i + \sum_{i \in V_d} a_i = 0 \tag{109}$$

6. Transshipment Problem

Variables:

$$x_{ij} = \text{flow from node } i \text{ to node } j \tag{110}$$

Model:

$$\min \sum_{i \in V} \sum_{\substack{j \in V \\ (i,j) \in E}} c_{ij} x_{ij} \tag{111}$$

$$\sum_{\substack{j \in V \\ (i,j) \in E}} x_{ij} - \sum_{\substack{j \in V \\ (j,i) \in E}} x_{ji} = a_i \quad \text{for } i \in V \tag{112}$$

$$x_{ij} \leq k_{ij} \quad \text{for } (i,j) \in E \tag{113}$$

$$x_{ij} \in \mathbb{R}_+ \quad \text{for } (i,j) \in E \tag{114}$$

7. Minimal Spanning Tree

Definition: Let $G = \{V, E\}$ be an undirected graph with cost c_{ij} given for each arc $\{i, j\}$. The objective is to search a spanning tree of G minimizing total cost.

Graph transformation: Set of undirected arcs E is transformed to set of directed arcs A in the following way:
Each arc $\{i, j\} \in E$ is replaced with directed arcs $(i, j) \in A$ and $(j, i) \in A$, $c_{ji} = c_{ij}$.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \tag{115}$$

$$y_{ij} = \text{flow from node } i \text{ to node } j \tag{116}$$

7. Minimal Spanning Tree

Model:

$$\min \sum_{i \in V} \sum_{\substack{j \in V \\ (i,j) \in A}} c_{ij} x_{ij} \tag{117}$$

$$\sum_{\substack{j \in V \\ (1,j) \in A}} x_{1j} = 0 \tag{118}$$

$$\sum_{\substack{j \in V \\ (i,j) \in A}} x_{ij} = 1 \quad \text{for } i \in V \setminus \{1\} \tag{119}$$

$$\sum_{\substack{j \in V \\ (i,j) \in A}} y_{ij} - \sum_{\substack{j \in V \\ (j,i) \in A}} y_{ji} = 1 \quad \text{for } i \in V \setminus \{1\} \tag{120}$$

$$0 \le y_{ij} \le (|V| - 1) x_{ij} \quad \text{for } (i,j) \in A \tag{121}$$

$$x_{ij} \in \{0,1\} \quad \text{for } (i,j) \in A \tag{122}$$

8. Minimal Steiner Tree

Definition: Let $G = \{V, E\}$ be a digraph, $s \in V$ be a source of the signal (transmitter), $D \subset V$ a set of users (receivers, destinations) and $T \subset V$ a set of transfer stations. Using cables, users can be connected to transmitter directly or through the transfer stations. Let c_{ij} be cost for connection $(i, j) \in E$. The use of transfer station $i \in T$ is charged f_i. The objective is to connect all users to the source with the minimal total cost.

Variables:

$$z_i = \begin{cases} 1 & \text{if node } i \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \tag{123}$$

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \tag{124}$$

$$y_{ij} = \text{flow from node } i \text{ to node } j \tag{125}$$

8. Minimal Steiner Tree

Model:

$$\min \sum_{i\in V} \sum_{\substack{j\in V\\(i,j)\in E}} c_{ij}x_{ij} + \sum_{i\in T} f_i z_i \tag{126}$$

$$z_i = 1 \quad \text{for } i \in D \tag{127}$$

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij} = z_i \quad \text{for } i \in V \setminus \{s\} \tag{128}$$

$$\sum_{\substack{j\in V\\(i,j)\in E}} y_{ij} - \sum_{\substack{j\in V\\(j,i)\in E}} y_{ji} = z_i \quad \text{for } i \in V \setminus \{s\} \tag{129}$$

$$0 \le y_{ij} \le (|V| - 1)x_{ij} \quad \text{for } (i,j) \in E \tag{130}$$

$$x_{ij} \in \{0,1\} \quad \text{for } (i,j) \in E \tag{131}$$

$$z_i \in \{0,1\} \quad \text{for } i \in T \tag{132}$$

Course Syllabus

Routing Problems

Classification of Problems

- Network routing
  - Node routing
    - Travelling Salesman Problem (TSP) - infinite capacity of vehicles
    - Vehicle Routing Problem (VRP) - limited capacity of vehicles
  - Arc routing
    - Chinese Postman Problem (CPP)
- Number of depots and vehicles
  - One depot with one or multiple vehicles
  - Multiple depots
- Knowledge of customers
  - Static problems - all customers are known in advance
  - Dynamic problems - advanced customers and on-line customers
- Objective
  - Total travelled distance (total cost) minimization
  - Total travelled time minimization
  - Minimizing the longest time of completing all routes

1. Symmetric Travelling Salesman Problem

Definition: Let $G = \{U, E\}$ be a complete graph with distance c_{ij} given for each arc $\{i, j\}$ (matrix C is symmetric). Let node 1 be a depot and $|U| = n$. The objective is to determine the minimal Hamiltonian cycle.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if arc } \{i,j\} \text{ is used on the tour} \\ 0 & \text{otherwise} \end{cases} \qquad i < j \tag{133}$$

1. Symmetric Travelling Salesman Problem

Model I (Dantzig, Fulkerson, Johnson):

$$\min \sum_{\substack{i \in U \\ i<j}} \sum_{j \in U} c_{ij} x_{ij} \tag{134}$$

$$\sum_{\substack{j \in U \\ j<i}} x_{ji} + \sum_{\substack{j \in U \\ j>i}} x_{ij} = 2 \quad \text{for } i \in U \tag{135}$$

$$\sum_{\substack{i \in U' \\ i<j}} \sum_{j \in U'} x_{ij} \leq |U'| - 1 \quad \text{for } U' \subset U, 3 \leq |U'| \leq \left\lfloor \frac{|U|}{2} \right\rfloor \tag{136}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } i, j \in U, i < j. \tag{137}$$

1. Symmetric Travelling Salesman Problem

Model II (Dantzig, Fulkerson, Johnson):

Constraints (136) are replaced with

$$\sum_{i \in U'} \sum_{\substack{j \in U \setminus U' \\ i<j}} x_{ij} + \sum_{\substack{i \in U \setminus U' \\ i<j}} \sum_{j \in U'} x_{ij} \geq 1 \quad \text{for } U' \subset U, 3 \leq |U'| \leq \left\lfloor \frac{|U|}{2} \right\rfloor \tag{138}$$

2. Asymmetric Travelling Salesman Problem

Definition: Let $G = \{U, E\}$ be a complete digraph with distance c_{ij} given for each arc (i, j) (matrix C is generally asymmetric). Let node 1 be a depot and $|U| = n$. The objective is to determine the minimal Hamiltonian cycle.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if a vehicle travels directly} \\ & \text{between nodes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \tag{139}$$

$$u_i = \text{dummy variable in sub-tours eliminating constraints} \tag{140}$$

2. Asymmetric Travelling Salesman Problem

Model (Miller, Tucker, Zemlin):

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{141}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, 2, \ldots, n \tag{142}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1, 2, \ldots, n \tag{143}$$

$$u_i + 1 - (n-1)(1 - x_{ij}) \leq u_j \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 2, 3, \ldots, n \end{array} \tag{144}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 1, 2, \ldots, n \end{array} \tag{145}$$

$$u_i \in \mathbb{R}_+ \quad \text{for } i = 1, 2, \ldots, n \tag{146}$$

3. Travelling Salesman Problem with Time Windows (TSPTW)

Definition: Let Asymmetric TSP be defined. Each node i has to be visited within time interval $\langle e_i, l_i \rangle$. A vehicle spends given time S_i at node i. Let d_{ij} be the traversal time between nodes i and j. The objective is to determine the minimal Hamiltonian cycle (in terms of distance) respecting all time windows.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if a vehicle travels directly} \\ & \text{between nodes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \tag{147}$$

$$t_i = \text{time node } i \text{ is visited} \tag{148}$$

3. Travelling Salesman Problem with Time Windows (TSPTW)
Model modification:

$$e_i \leq t_i \leq l_i \quad \text{for } i = 2, 3, \ldots, n \tag{149}$$

$$t_1 = 0 \tag{150}$$

$$t_i \in \mathbb{R}_+ \quad \text{for } i = 2, 3, \ldots, n \tag{151}$$

Variables u_i are eliminated and constraints (144) are replaced with

$$t_i + S_i + d_{ij} - M(1 - x_{ij}) \leq t_j \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 2, 3, \ldots, n \end{array} \; i \neq j \tag{152}$$

4. Metric TSP (Δ - TSP)

Triangle inequality:

$$c_{ij} \leq c_{ik} + c_{kj} \quad \text{for } i, j, k = 1, 2, \ldots, n, i \neq j \neq k \tag{153}$$

5. Euclidian TSP (Planar TSP)

Euclidian distance:

$$c_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \quad \text{for } i, j = 1, 2, \ldots, n, i \neq j \tag{154}$$

6. Open TSP

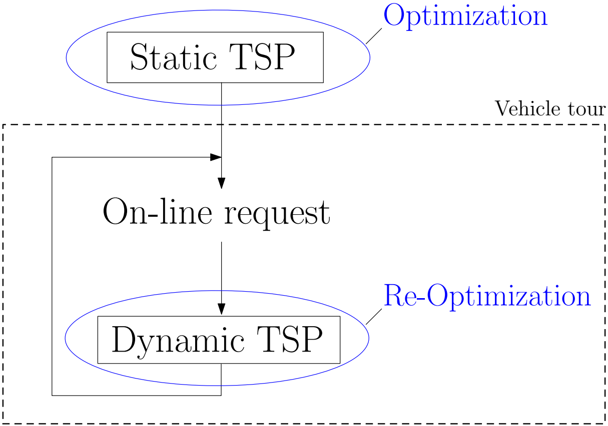
Instead of minimal Hamiltonian cycle, minimal open path through all nodes is being searched for (a tour is not finished at the depot):

$$\text{we set } c_{i1} = 0 \quad \text{for } i = 2, 3, \ldots, n \tag{155}$$

7. Dynamic TSP

In static version of TSP, all customers are known in advance.

In dynamic version, on-line requests occur when optimal tour is being realized.



8. Vehicle Routing Problem

Definition: Let $G = \{U, E\}$ be a complete digraph with distance c_{ij} given for each arc (i, j). Let node 1 be a depot, where one vehicle with capacity V is available. Let $|U| = n$. Each customer i is associated with request of size q_i. The objective is to satisfy all customers' requirements and to minimize total length of the routes.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{if a vehicle travels directly} \\ & \text{between nodes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \tag{156}$$

$$u_i = \text{dummy variable for the balance of load on the vehicle} \tag{157}$$

Assumptions:

$$\sum_{i=2}^{n} q_i > V \tag{158}$$

$$q_i \leq V \quad \text{for } i = 2, 3, \ldots, n \tag{159}$$

8. Vehicle Routing Problem

Model:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{160}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 2, 3, \ldots, n \tag{161}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 2, 3, \ldots, n \tag{162}$$

$$u_i + q_j - V(1 - x_{ij}) \leq u_j \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 2, 3, \ldots, n \end{array} \tag{163}$$

$$u_i \leq V \quad \text{for } i = 2, 3, \ldots, n \tag{164}$$

$$u_1 = 0 \tag{165}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 1, 2, \ldots, n \end{array} \tag{166}$$

$$u_i \in \mathbb{R}_+ \quad \text{for } i = 2, 3, \ldots, n \tag{167}$$

9. Heterogenous Fleet Vehicle Routing Problem

Definition: Let VRP be defined with K types of vehicles available in depot. For each type k its capacity V_k, available number p_k and cost coefficient d_k are given. The objective is to satisfy all customers' requirements and to minimize total cost.

Variables:

$$x_{ij}^k = \begin{cases} 1 & \text{if a vehicle of type } k \text{ travels directly} \\ & \text{between nodes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \quad (168)$$

$$u_i = \text{dummy variable for the balance of load on the vehicle} \quad (169)$$

Assumption:

$$\sum_{i=2}^{n} q_i \leq \sum_{k=1}^{K} p_k V_k \quad (170)$$

Notation:

$$\overline{V} = \max_{k=1,2,\ldots,K} V_k \quad (171)$$

9. Heterogenous Fleet Vehicle Routing Problem

Model:

$$\min \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{j=1}^{n} d_k c_{ij} x_{ij}^k \tag{172}$$

$$\sum_{k=1}^{K} \sum_{j=1}^{n} x_{ij}^k = 1 \quad \text{for } i = 2, 3, \ldots, n \tag{173}$$

$$\sum_{i=1}^{n} x_{ij}^k = \sum_{i=1}^{n} x_{ji}^k \quad \text{for } \begin{array}{l} j = 1, 2, \ldots, n \\ k = 1, 2, \ldots, K \end{array} \tag{174}$$

$$\sum_{j=2}^{n} x_{1j}^k \leq p_k \quad \text{for } k = 1, 2, \ldots, K \tag{175}$$

$$u_i + q_j - \overline{V}(1 - x_{ij}^k) \leq u_j \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 2, 3, \ldots, n \\ k = 1, 2, \ldots, K \end{array} \tag{176}$$

9. Heterogenous Fleet Vehicle Routing Problem

Model (continued):

$$u_i \leq \sum_{j=1}^{n} \sum_{k=1}^{K} x_{ij}^k V_k \quad \text{for } i = 1, 2, \ldots, n \tag{177}$$

$$u_1 = 0 \tag{178}$$

$$x_{ij}^k \in \{0, 1\} \quad \text{for } \begin{array}{l} i = 1, 2, \ldots, n \\ j = 1, 2, \ldots, n \\ k = 1, 2, \ldots, K \end{array} \tag{179}$$

$$u_i \in \mathbb{R}_+ \quad \text{for } i = 2, 3, \ldots, n \tag{180}$$

10. Vehicle Routing Problem with Time Windows (VRPTW)
If time windows are defined in the problem, variable t_i and all associated constraints are introduced similarly to TSPTW.

11. Split Delivery Vehicle Routing Problem (SDVRP)
The model of VRP cannot be used if $\exists i, q_i > V$. Such a request must be split into multiple routes. Even in case $q_i \leq V, \forall i$, it could be advantageous to split deliveries. In the model, variable Q_i^k denotes a part of request q_i covered on route k:

$$\sum_{k=1}^{K} Q_i^k = q_i \quad \text{for } i = 2, 3, \ldots, n \tag{181}$$

12. Pickup and Delivery Problem (PDP)

- One-to-One PDP (Dial-a-Ride Problem, Messenger Problem)
  Each request originates at one location and is destined for another location. Vehicle routes start and end at a common depot.
- Many-to-Many PDP
  Commodity may be picked up at one of many locations and also be delivered to one of many locations.
- One-to-Many-to-One PDP
  Each customer receives a delivery originating at a common depot and sends a pickup quantity to the depot.

13. Undirected Chinese Postman Problem

Definition: Let $G = \{U, E\}$ be an undirected and connected graph. Cost c_{ij} for each arc $\{i, j\}$ is given. The objective is to find a minimum cost tour passing through each arc at least once.

Theorem: An undirected graph G is Eulerian if and only if G is connected and the degrees of all of its nodes are even.

If G is not Eulerian, we will construct a supergraph G^* of G such that G^* is Eulerian and includes an Eulerian tour that is shorter than the Eulerian tour in any other supergraph of G.

13. Undirected Chinese Postman Problem

Variables in model I:

$$x_{ij} = \begin{cases} 1 & \text{if arc } \{i,j\} \text{ is copied in } G^* \\ 0 & \text{otherwise} \end{cases} \qquad i < j \tag{182}$$

$$y_i = \text{a dummy variable for the expression of an even/odd number} \tag{183}$$

Notation in model I:

$U_0 \subset U$ is a set of nodes with even degree
$U_1 \subset U$ is a set of nodes with odd degree

13. Undirected Chinese Postman Problem

Model I:

$$\min \sum_{\substack{\{i,j\} \in E \\ i<j}} c_{ij} x_{ij} \tag{184}$$

$$\sum_{\substack{\{j,i\} \in E \\ j<i}} x_{ji} + \sum_{\substack{\{i,j\} \in E \\ j>i}} x_{ij} = 2y_i \quad \text{for } i \in U_0 \tag{185}$$

$$\sum_{\substack{\{j,i\} \in E \\ j<i}} x_{ji} + \sum_{\substack{\{i,j\} \in E \\ j>i}} x_{ij} = 2y_i + 1 \quad \text{for } i \in U_1 \tag{186}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } \{i, j\} \in E, i < j \tag{187}$$

$$y_i \in \mathbb{Z}_+ \quad \text{for } i \in U \tag{188}$$

13. Undirected Chinese Postman Problem

Variables in model II:

$$x_{ij} = \text{a number of copies of arc } \{i,j\} \text{ in } G^* \tag{189}$$

Model II:

$$\min \sum_{\{i,j\}\in E} c_{ij}x_{ij} \tag{190}$$

$$x_{ij} + x_{ji} \geq 1 \quad \text{for } \{i,j\},\{j,i\} \in E \tag{191}$$

$$\sum_{\{j,i\}\in E} x_{ji} = \sum_{\{i,j\}\in E} x_{ij} \quad \text{for } i \in U \tag{192}$$

$$x_{ij} \in \mathbb{Z}_+ \quad \text{for } \{i,j\} \in E \tag{193}$$

14. Directed Chinese Postman Problem

Definition: Let $G = \{U, E\}$ be a strongly connected digraph. Cost c_{ij} for each arc (i, j) is given. The objective is to find a minimum cost tour passing through each arc at least once.

Theorem: An directed graph G is Eulerian if and only if G is strongly connected and in-degree of each node is equal to its out-degree.

If G is not Eulerian, we will construct a supergraph G^* of G such that G^* is Eulerian and includes an Eulerian tour that is shorter than the Eulerian tour in any other supergraph of G.

14. Directed Chinese Postman Problem

Notation:

$\deg_i^- =$ in-degree of node i
$\deg_i^+ =$ out-degree of node i
$I =$ set of nodes i for which $\deg_i^- > \deg_i^+$
$J =$ set of nodes j for which $\deg_j^- < \deg_j^+$
$a_i = \deg_i^- - \deg_i^+$ for $i \in I$
$b_j = \deg_j^+ - \deg_j^-$ for $j \in J$
$d_{ij} =$ the length of a shortest path from $i \in I$ to $j \in J$

Variables:

$$x_{ij} = \text{a number of extra times each arc of the shortest path from } i \text{ to } j \text{ has to be traversed} \quad (194)$$

14. Directed Chinese Postman Problem

Model:

$$\min \sum_{i \in I} \sum_{j \in J} d_{ij} x_{ij} \tag{195}$$

$$\sum_{j \in J} x_{ij} = a_i \quad \text{for } i \in I \tag{196}$$

$$\sum_{i \in I} x_{ij} = b_j \quad \text{for } j \in J \tag{197}$$

$$x_{ij} \in \mathbb{R}_+ \quad \text{for } \begin{array}{c} i \in I \\ j \in J \end{array} \tag{198}$$

15. Mixed Chinese Postman Problem (street sweeping)
CPP on mixed graph $G = \{U, E\}$ is defined.

16. Rural Postman Problem (mail delivery)
Let $G = \{U, E\}$ be a connected graph with set $R \subset E$ of required arcs that must be traversed at least once. The remaining arcs in $E \setminus R$ are optional and may be used in the optimal solution.

17. Capacitated Arc Routing Problem (garbage collection)
Let $G = \{U, E\}$ be a connected graph with requirement q_{ij} given for each arc $\{i, j\}$ (for each required arc in a rural version). Capacity of a vehicle covering requirements is limited. Multiple tours have to be found without exceeding the vehicle capacity on any of them.

18. Hierarchical Postman Problem (snow plowing)
Different priority levels are defined for arcs in the graph.

Course Syllabus

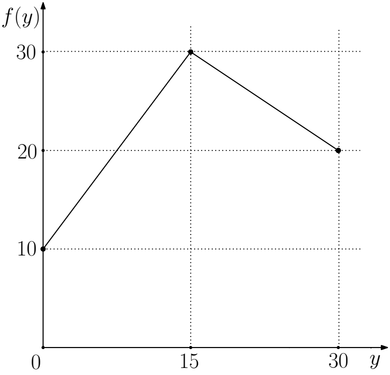
1. Piecewise linear function

Definition: Let piecewise linear function $f(y)$ be defined on a set of intervals $I_1, I_2, \ldots, I_{r-1}$ denoted $\langle a_1, a_2\rangle, \langle a_2, a_3\rangle, \ldots, \langle a_{r-1}, a_r\rangle$.
For each interval bound a_k, function value $f(a_k)$ is given.
Create a model of the function with the use of discrete variables.

Example:

$a_1 = 0$
$a_2 = 15$
$a_3 = 30$

$f(a_1) = 10$
$f(a_2) = 30$
$f(a_3) = 20$



1. Piecewise linear function

Variables:

$$x_1 = \begin{cases} 1 & \text{if } y \in \langle 0, 15 \rangle \\ 0 & \text{otherwise} \end{cases} \tag{199}$$

$$x_2 = \begin{cases} 1 & \text{if } y \in \langle 15, 30 \rangle \\ 0 & \text{otherwise} \end{cases} \tag{200}$$

$$\lambda_1, \lambda_2, \lambda_3 = \text{dummy variables used in convex combinations} \tag{201}$$

1. Piecewise linear function

Formulation:

- If $y \in \langle 0, 15 \rangle$ then

$$\left.\begin{aligned} y &= 0\lambda_1 + 15\lambda_2 = a_1\lambda_1 + a_2\lambda_2 \\ f(y) &= 10\lambda_1 + 30\lambda_2 = f(a_1)\lambda_1 + f(a_2)\lambda_2 \end{aligned}\right\} \begin{aligned} &\lambda_1 + \lambda_2 = 1 \\ &\lambda_1, \lambda_2 \in \mathbb{R}_+ \end{aligned} \tag{202}$$

- If $y \in \langle 15, 30 \rangle$ then

$$\left.\begin{aligned} y &= 15\lambda_2 + 30\lambda_3 = a_2\lambda_2 + a_3\lambda_3 \\ f(y) &= 30\lambda_2 + 20\lambda_3 = f(a_2)\lambda_2 + f(a_3)\lambda_3 \end{aligned}\right\} \begin{aligned} &\lambda_2 + \lambda_3 = 1 \\ &\lambda_2, \lambda_3 \in \mathbb{R}_+ \end{aligned} \tag{203}$$

$$\left.\begin{aligned} x_1 = 0 \Rightarrow \lambda_1 = 0 \\ x_2 = 0 \Rightarrow \lambda_3 = 0 \end{aligned}\right\} \begin{aligned} &\lambda_1 \le x_1 \\ &\lambda_3 \le x_2 \\ &(\lambda_2 \le x_1 + x_2)^* \end{aligned} \tag{204}$$

* The inequality is important in case of more than 2 intervals being defined ($x_1 = 0 \wedge x_2 = 0 \Rightarrow \lambda_2 = 0$).

1. Piecewise linear function

Model:

$$y = 0\lambda_1 + 15\lambda_2 + 30\lambda_3 \tag{205}$$

$$f(y) = 10\lambda_1 + 30\lambda_2 + 20\lambda_3 \tag{206}$$

$$\lambda_1 \leq x_1 \tag{207}$$

$$\lambda_2 \leq x_1 + x_2 \tag{208}$$

$$\lambda_3 \leq x_2 \tag{209}$$

$$x_1 + x_2 = 1 \tag{210}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{211}$$

$$x_1, x_2 \in \{0, 1\} \tag{212}$$

$$\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}_+ \tag{213}$$

Formulations in Logical Variables

1. Piecewise linear function

Variables in general model:

$$x_i = \begin{cases} 1 & \text{if } y \in I_i = \langle a_i, a_{i+1} \rangle \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1, 2, \ldots, r-1 \tag{214}$$

$$\lambda_i = \begin{array}{l}\text{dummy variable used} \\ \text{in convex combinations}\end{array} \quad \text{for } i = 1, 2, \ldots, r \tag{215}$$

General model:

$$y = \sum_{i=1}^{r} a_i \lambda_i \tag{216}$$

$$f(y) = \sum_{i=1}^{r} f(a_i) \lambda_i \tag{217}$$

1. Piecewise linear function

General model (continued):

$$\sum_{i=1}^{r-1} x_i = 1 \tag{218}$$

$$\sum_{i=1}^{r} \lambda_i = 1 \tag{219}$$

$$\lambda_1 \leq x_1 \tag{220}$$

$$\lambda_r \leq x_{r-1} \tag{221}$$

$$\lambda_i \leq x_{i-1} + x_i \quad \text{for } i = 2, 3, \ldots, r-1 \tag{222}$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, 2, \ldots, r-1 \tag{223}$$

$$\lambda_i \in \mathbb{R}_+ \quad \text{for } i = 1, 2, \ldots, r \tag{224}$$

2. Nonconvex solution space

Example: Let the following model be given. Use discrete variables it could be solved as MIP model.

$$y_1 + y_2 \leq 40$$
$$y_1 \geq 20 \text{ or } y_2 \geq 10$$
$$y_1, y_2 \in \mathbb{R}_+$$

Variables:

$$x_1 = \begin{cases} 1 & \text{if } y_1 \geq 20 \\ 0 & \text{otherwise} \end{cases} \tag{225}$$

$$x_2 = \begin{cases} 1 & \text{if } y_2 \geq 10 \\ 0 & \text{otherwise} \end{cases} \tag{226}$$

2. Nonconvex solution space

Model:

$$y_1 + y_2 \leq 40 \tag{227}$$

$$y_1 \geq 20x_1 \tag{228}$$

$$y_2 \geq 10x_2 \tag{229}$$

$$x_1 + x_2 \geq 1 \tag{230}$$

$$y_1, y_2 \in \mathbb{R}_+ \tag{231}$$

$$x_1, x_2 \in \{0, 1\} \tag{232}$$

3. Disjunctive constraints (either-or constraints)

Example: Three products can be produced on a machine either in the sequence $P_1 \to P_2 \to P_3$ or $P_3 \to P_2 \to P_1$. Assume production of P_i takes t_i. Formulate the constraints for allowable production.

Variables:

$$y_i = \text{starting production time of product } P_i \tag{233}$$

$$x = \begin{cases} 1 & \text{if sequence } P_1 \to P_2 \to P_3 \text{ is used} \\ 0 & \text{if sequence } P_3 \to P_2 \to P_1 \text{ is used} \end{cases} \tag{234}$$

3. Disjunctive constraints (either-or constraints)

Model:

$$y_1 + t_1 \leq y_2 + M(1 - x) \tag{235}$$

$$y_2 + t_2 \leq y_3 + M(1 - x) \tag{236}$$

$$y_3 + t_3 \leq y_2 + Mx \tag{237}$$

$$y_2 + t_2 \leq y_1 + Mx \tag{238}$$

$$y_i \in \mathbb{R}_+ \quad \text{for } i = 1, 2, 3 \tag{239}$$

$$x \in \{0, 1\} \tag{240}$$

4. Disjunctive constraints (k out of m constraints must hold)

Example: Suppose a model includes a set of m constraints. Let constraint i be defined as $a_i^T y \leq b_i$. Assure exactly k of all constraints must hold ($k < m$).

Dummy variables:

$$x_i = \begin{cases} 1 & \text{if constraint } i \text{ is chosen} \\ 0 & \text{otherwise} \end{cases} \tag{241}$$

Constraints:

$$a_i^T y \leq b_i + M(1 - x_i) \quad \text{for } i = 1, 2, \ldots, m \tag{242}$$

$$\sum_{i=1}^{m} x_i = k \tag{243}$$

Formulations in Logical Variables

5. Production level planning with Yes-or-No decision

Example: A company is considering whether to produce a new product or not. If so, the level of production should be at least 500 units but not more than 1000 units. Formulate the decision constraints.

Variables:

$$y = \text{level of production} \tag{244}$$

$$x = \begin{cases} 1 & \text{if the decision for production is yes} \\ 0 & \text{if the decision for production is no} \end{cases} \tag{245}$$

Model:

$$500x \leq y \leq 1000x \tag{246}$$

$$y \in \mathbb{Z}_+ \tag{247}$$

$$x \in \{0, 1\} \tag{248}$$

6. Planning production on discrete levels

Example: A company decides to produce either 500 or 1000 or 2000 units of certain product. Formulate the decision constraints.

Variables:

$$y = \text{level of production} \tag{249}$$

$$x_i = \begin{cases} 1 & \text{if the production is set on } i\text{-th level} \\ 0 & \text{otherwise} \end{cases} \qquad i = 1, 2, 3 \tag{250}$$

Model:

$$y = 500x_1 + 1000x_2 + 2000x_3 \tag{251}$$

$$x_1 + x_2 + x_3 = 1 \tag{252}$$

$$y \in \mathbb{Z}_+ \tag{253}$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, 2, 3 \tag{254}$$

Formulations in Logical Variables

7. Disjunctive variables (k out of n variables must be positive)

Example: A company is able to produce n types of products. It is considering to produce only k of them. For each product i, maximal production level q_i is given. Formulate the decision constraints.

Variables:

$$y_i = \text{production level of product } i \tag{255}$$

$$x_i = \begin{cases} 1 & \text{if product } i \text{ is produced} \\ 0 & \text{otherwise} \end{cases} \tag{256}$$

Constraints:

$$\frac{1}{M} x_i \leq y_i \leq q_i x_i \quad \text{for } i = 1, 2, \ldots, n \tag{257}$$

$$\sum_{i=1}^{n} x_i = k \tag{258}$$

7. Disjunctive variables (k out of n variables must be positive)

Example (equality condition): In previous example, production levels of k selected products must be equal.

Additional variable:

$$w = \text{production level of produced products} \tag{259}$$

Additional constraints:

$$w - M(1 - x_i) \leq y_i \leq w + M(1 - x_i) \quad \text{for } i = 1, 2, \ldots, n \tag{260}$$

8. Definition of a variable equal to the minimum of other variables

Example: Define a variable equal to the minimum of n variables.

Variables:

$$w = \min(y_1, y_2, \dots, y_n) \tag{261}$$

$$x_i = \begin{cases} 1 & \text{if } w = y_i \\ 0 & \text{otherwise} \end{cases} \tag{262}$$

Constraints:

$$w \le y_i \le w + M(1 - x_i) \quad \text{for } i = 1, 2, \dots, n \tag{263}$$

$$\sum_{i=1}^{n} x_i \ge 1 \tag{264}$$

9. Simplifying Product of Binary Variables

Example: Simplify the maximization objective function $x_1 x_2 x_3$ (all variables are binary) to use a linear model (MIP model).

Dummy variable:

$$w = x_1 x_2 x_3 \tag{265}$$

Model:

$$\max w \tag{266}$$

$$x_1 + x_2 + x_3 - 2 \leq w \tag{267}$$

$$w \leq x_i \quad \text{for } i = 1, 2, 3 \tag{268}$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, 2, 3 \tag{269}$$

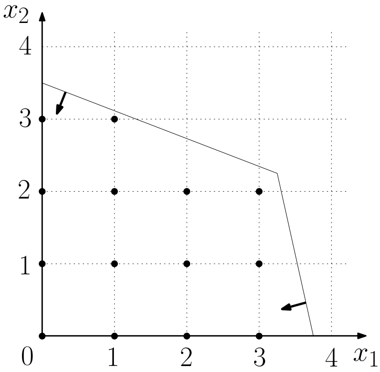
$$w \in \mathbb{R}_+ \tag{270}$$

Course Syllabus

Feasible set of points in linear and integer programming

$$\text{(LP)} \qquad P = \{x \in \mathbb{R}^n_+ : Ax \le b\} \tag{271}$$

$$\text{(IP)} \qquad S = \{x \in \mathbb{Z}^n_+ : Ax \le b\} \tag{272}$$



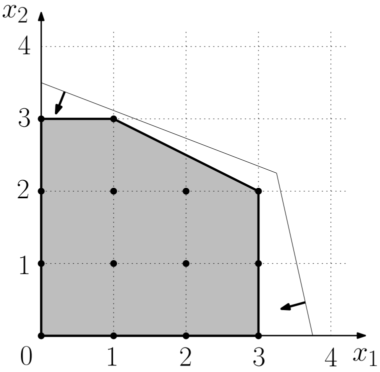
Polyhedral Theory

Definition: A *polyhedron* $P \subseteq \mathbb{R}^n$ is the set of points that satisfy a finite number of linear inequalities: $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. A polyhedron is *bounded* if there exists $\delta \in \mathbb{R}_+$ such that $P \subseteq \{x \in \mathbb{R}^n : -\delta \leq x_j \leq \delta \text{ for } j = 1, 2, \ldots, n\}$. A bounded polyhedron is called a *polytope*.

Definition: Given a set $S \subseteq \mathbb{R}^n$, a point $x \in \mathbb{R}^n$ is a *convex combination* of points of S if there exists a finite set of points $\{x^1, x^2, \ldots, x^t\}$ in S and $\lambda \in \mathbb{R}^t_+$ such that $\sum_{i=1}^t \lambda_i = 1$ and $x = \sum_{i=1}^t \lambda_i x^i$.

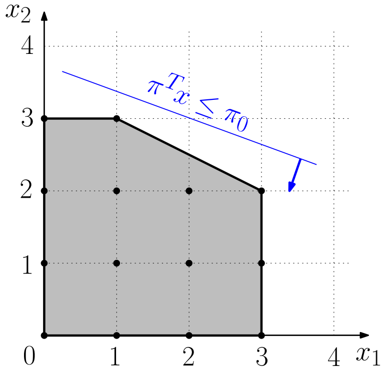
Definition: $T \subseteq \mathbb{R}^n$ is a *convex set* if $x^1, x^2 \in T$ implies that $\lambda x^1 + (1 - \lambda)x^2 \in T$ for all $\lambda \in \langle 0, 1 \rangle$.

Definition: A *convex hull* of S, denoted by $\text{conv}(S)$, is the set of all points that are convex combinations of points in S.

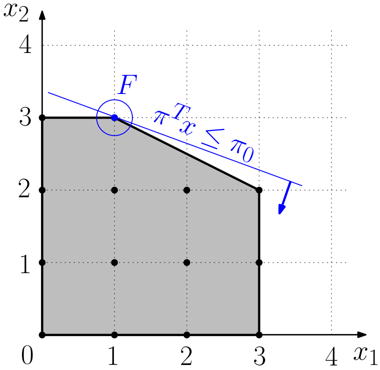


$S \subset \text{conv}(S) \subset P$

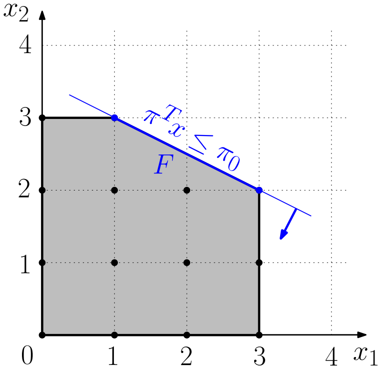
Definition: The inequality $\pi^T x \leq \pi_0$ is called a *valid inequality* for S if it is satisfied by all points in S.



Definition: If $\pi^T x \leq \pi_0$ is a valid inequality for S and $\exists x^0 \in S$ such that $\pi^T x^0 = \pi_0$ we say that the inequality *supports* S. The set $F = \{x \in \text{conv}(S) : \pi^T x = \pi_0\}$ is called a *face* of conv(S). We say that the inequality $\pi^T x \leq \pi_0$ *represents* F.



Definition: A face F of $\text{conv}(S)$ is called a *facet* of $\text{conv}(S)$ if $\dim F = \dim \text{conv}(S) - 1$.



Polyhedral Theory

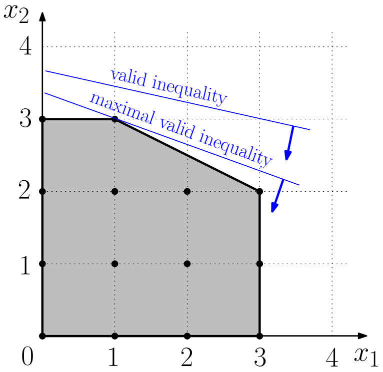
Definition: The valid inequalities $\pi^T x \leq \pi_0$ and $\gamma^T x \leq \gamma_0$ are said to be *equivalent* if $\gamma = \lambda\pi$ and $\gamma_0 = \lambda\pi_0$ for some $\lambda > 0$.

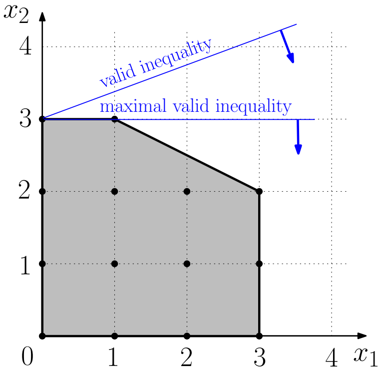
Definition: Let $\pi^T x \leq \pi_0$ and $\gamma^T x \leq \gamma_0$ be two valid inequalities for $\text{conv}(S)$ that are not equivalent. If there exists $\lambda > 0$ such that $\gamma \geq \lambda\pi$ and $\gamma_0 \leq \lambda\pi_0$ then we say that $\gamma^T x \leq \gamma_0$ *dominates* or *is stronger than* $\pi^T x \leq \pi_0$. We can also say that $\pi^T x \leq \pi_0$ *is dominated* or *is weaker than* $\gamma^T x \leq \gamma_0$.

Observe that if $\gamma^T x \leq \gamma_0$ *dominates* $\pi^T x \leq \pi_0$ then $\{x \in \mathbb{R}^n_+ : \gamma^T x \leq \gamma_0\} \subset \{x \in \mathbb{R}^n_+ : \pi^T x \leq \pi_0\}$.

Definition: A *maximal* valid inequality is one that is not dominated by any other valid inequality.

Any maximal valid inequality for S defines a nonempty face of $\text{conv}(S)$, and the set of maximal valid inequalities contains all of the facet-defining inequalities for $\text{conv}(S)$.

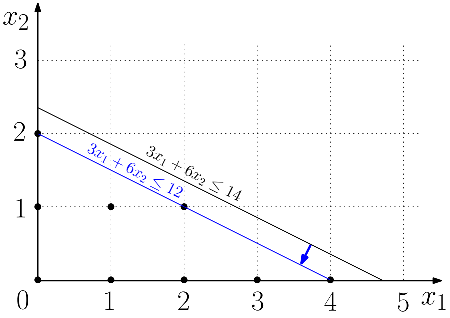




Strengthening Inequalities

Theorem (Diophantos): The linear equation $\sum_{j=1}^{n} \pi_j x_j = \pi_0$, where $\pi_j \in \mathbb{Z}$ $(j = 0, 1, \ldots, n)$ has a solution $x \in \mathbb{Z}^n$ if and only if the greatest common divisor of π_j $(j = 1, 2, \ldots, n)$ divides π_0 in integers.

Example: Let $3x_1 + 6x_2 \leq 14$ be the valid inequality for S. The objective is to find stronger valid inequality that supports S.



Strengthening Inequalities - Lifting

Definition: Let the inequality $\sum_{j=1}^{n} \pi_j x_j \leq \pi_0$ be given, where $\pi_j \in \mathbb{R}_+$ $(j = 0, 1, \ldots, n)$ and $x \in \mathbb{B}^n$. If for some $\Delta_k > 0$ the inequality $\sum_{j=1}^{n} \pi_j x_j + \Delta_k x_k \leq \pi_0$ is valid, then it is said to have been *lifted* from the original inequality with respect to x_k.

Algorithm:

Repeat for $k = 1, 2, \ldots, n$:

1. Set $x_k = 1$ and denote $\alpha_k = \max\limits_{x_k \in \mathbb{B}} \sum_{j=1}^{n} \pi_j x_j \leq \pi_0$
2. Set $\Delta_k = \pi_0 - \alpha_k$
3. Replace π_k by $\pi_k + \Delta_k$
4. The inequality $\sum_{j=1}^{n} \pi_j x_j \leq \pi_0$ is lifted with respect to variable x_k

The inequality $\sum_{j=1}^{n} \pi_j x_j \leq \pi_0$ is lifted with respect to all variables.

Strengthening Inequalities - Lifting

Example: Lift the inequality $4x_1 + 5x_2 + 6x_3 + 8x_4 \leq 13$, where $x_j \in \mathbb{B}$ $(j = 1, 2, 3, 4)$.

$x_1 = 1 \rightarrow \alpha_1 = 12 \rightarrow \Delta_1 = 1 \rightarrow \pi_1 = 5 \rightarrow 5x_1 + 5x_2 + 6x_3 + 8x_4 \leq 13$

$x_2 = 1 \rightarrow \alpha_2 = 13 \rightarrow \Delta_2 = 0 \rightarrow \pi_2 = 5 \rightarrow 5x_1 + 5x_2 + 6x_3 + 8x_4 \leq 13$

$x_3 = 1 \rightarrow \alpha_3 = 11 \rightarrow \Delta_3 = 2 \rightarrow \pi_3 = 8 \rightarrow 5x_1 + 5x_2 + 8x_3 + 8x_4 \leq 13$

$x_4 = 1 \rightarrow \alpha_4 = 13 \rightarrow \Delta_4 = 0 \rightarrow \pi_4 = 8 \rightarrow 5x_1 + 5x_2 + 8x_3 + 8x_4 \leq 13$

Strengthening Inequalities - Variable fixing

Example: Let $2x_1 + 3x_2 + 4x_3 - 15x_4 \geq 2$ with $x_j \in \mathbb{B}$ $(j = 1, 2, 3, 4)$.
If $x_4 = 1$ the inequality cannot be satisfied $\rightarrow$ feasible solutions exist if variable is fixed $x_4 = 0$.

Example: Let $20x_1 + 5x_2 + 1x_3 - 8x_4 \geq 7$ with $x_j \in \mathbb{B}$ $(j = 1, 2, 3, 4)$.
Fixing $x_1 = 1$.

Example: Let $x_1 + x_2 + 3x_3 = 4$ with $x_j \in \mathbb{B}$ $(j = 1, 2, 3)$.
Because $x_3 = 0$ is impossible, variable is fixed $x_3 = 1$. Then, the equation is reduced to $x_1 + x_2 = 1$.

Strengthening Inequalities - Gomory cut

Let $S = \{x \in \mathbb{Z}_+^n : \pi^T x = \pi_0\}$, where $\pi_j \in \mathbb{R}$ $(j = 0, 1, \ldots, n)$. Let us select some $d \in \mathbb{N}$, then each π_j is possible to express as

$$\pi_j = \alpha_j d + \pi'_j, \tag{273}$$

where $\alpha_j = \left\lfloor \frac{\pi_j}{d} \right\rfloor$ and $\pi'_j = \pi_j \bmod d$. Thus $\alpha_j \in \mathbb{Z}$ and $\pi'_j \in \langle 0, d)$.
Then, the equation

$$\sum_{j=1}^{n} \pi_j x_j = \pi_0 \tag{274}$$

can be written as

$$\sum_{j=1}^{n} (\alpha_j d + \pi'_j) x_j = \alpha_0 d + \pi'_0 \tag{275}$$

or

$$d(\sum_{j=1}^{n} \alpha_j x_j - \alpha_0) = \pi'_0 - \sum_{j=1}^{n} \pi'_j x_j. \tag{276}$$

Strengthening Inequalities - Gomory cut

Because on left-hand side value in (276) is the integer multiple of d, right-hand side must be also integer value. Due to $\pi'_0 \in \langle 0, d)$ and $\sum_{j=1}^{n} \pi'_j x_j \geq 0$, right-hand side value cannot be positive multiple of d, i.e. it must be non-positive. Hence, left-hand side value must be non-positive as well.

Fractional Gomory cut:

$$\pi'_0 - \sum_{j=1}^{n} \pi'_j x_j \leq 0 \qquad \rightarrow \qquad \sum_{j=1}^{n} \pi'_j x_j \geq \pi'_0 \tag{277}$$

All-integer Gomory cut:

$$\sum_{j=1}^{n} \alpha_j x_j - \alpha_0 \leq 0 \qquad \rightarrow \qquad \sum_{j=1}^{n} \alpha_j x_j \leq \alpha_0 \tag{278}$$

Strengthening Inequalities - Gomory cut

Example: Let $S = \{x \in \mathbb{Z}^3_+ : 37x_1 - 68x_2 + 78x_3 \leq 141\}$. Find a stronger valid inequality for S.

Transformation to the equation $37x_1 - 68x_2 + 78x_3 + x_4 = 141$

Selection of $d = 12$

$$37 = 3.12 + 1$$
$$-68 = -6.12 + 4$$
$$78 = 6.12 + 6$$
$$1 = 0.12 + 1$$
$$141 = 11.12 + 9$$

Fractional Gomory cut: $x_1 + 4x_2 + 6x_3 + x_4 \geq 9$

All-integer Gomory cut: $3x_1 - 6x_2 + 6x_3 \leq 11$

Course Syllabus

Unimodularity

Definition: A square, integer matrix A is called *unimodular* if its determinant $\det(A) = \pm 1$. An integer matrix A is called *totally unimodular* if every square, nonsingular submatrix of A is unimodular.

Observation: If matrix A is totally unimodular, $a_{ij} \in \{+1, -1, 0\}$ for all i, j.

Theorem (sufficient condition): A matrix A is totally unimodular if

- $a_{ij} \in \{+1, -1, 0\}$ for all i, j
- each column contains at most two nonzero coefficients
- the rows of A can be partitioned into two sets such that
  - if a column has two coefficients of the same sign, their rows are in different sets
  - if a column has two coefficients of different signs, their rows are in the same set

Unimodularity

Let the following linear programming problem with integral data A, b be given:

$$z_{\text{LP}} = \max\{c^T x : Ax \leq b, x \in \mathbb{R}^n_+\}. \tag{279}$$

A vector of basic variables can be expressed as

$$x_B = B^{-1}b = \frac{B^{\text{adj}}}{\det(B)}b, \tag{280}$$

where B is basis, B^{-1} its inverse and B^{adj} is the adjoint matrix of B (the transpose of the cofactor matrix of B).

Observation: If the optimal basis B is unimodular, then the optimum solution is integral.

Proposition: If matrix A is totally unimodular, then the optimum solution is integral.

Examples: transportation problem, flow problem, ...

Course Syllabus

Relaxation

Definition: Let the following integer programming problem (IP) be given:

$$z_{\mathrm{IP}} = \max\{c^T x : x \in S \subseteq \mathbb{Z}_+^n\}. \tag{281}$$

The problem (R)

$$z_{\mathrm{R}} = \max\{d^T x : x \in X \subseteq \mathbb{R}_+^n\} \tag{282}$$

is a *relaxation* of (IP) if

- $S \subseteq X$
- $d^T x \geq c^T x$ for all $x \in X$.

Proposition: If (R) is a relaxation of (IP) then $z_{\mathrm{R}} \geq z_{\mathrm{IP}}$, i.e. z_{R} is the upper bound for z_{IP}.

Linear Programming Relaxation

Definition: Let the following integer programming problem (IP) be given:

$$z_{\text{IP}} = \max\{c^T x : Ax \leq b, x \in \mathbb{Z}_+^n\}. \tag{283}$$

The problem (LP)

$$z_{\text{LP}} = \max\{c^T x : Ax \leq b, x \in \mathbb{R}_+^n\} \tag{284}$$

is a *linear programming relaxation* of (IP).

In case of binary integer programming problem (BIP)

$$z_{\text{BIP}} = \max\{c^T x : Ax \leq b, x \in \mathbb{B}^n\}, \mathbb{B} = \{0, 1\} \tag{285}$$

a linear programming relaxation is defined as

$$z_{\text{LP}} = \max\{c^T x : Ax \leq b, 0 \leq x_j \leq 1, j = 1, 2, \ldots, n\}. \tag{286}$$

Linear Programming Relaxation

Definition: The *absolute integrality gap* is defined as the difference

$$\text{Gap} = z_{\text{LP}} - z_{\text{IP}} \tag{287}$$

and for $z_{\text{IP}} \neq 0$, the *relative integrality gap* is defined as

$$\text{Gap\%} = \frac{z_{\text{LP}} - z_{\text{IP}}}{|z_{\text{IP}}|} 100\%. \tag{288}$$

Linear programming relaxation can be also written as

$$z_{\text{LP}} = \max\{c^T x : x \in Q\}, \tag{289}$$

where $S = \{x : Ax \leq b, x \in \mathbb{Z}_+^n\} \subset \text{conv}(S) \subset Q$.

Lagrangian Relaxation

Definition: Let the following integer programming problem (IP) be given:

$$z_{\mathrm{IP}} = \max\{c^T x : Ax \leq b, x \in \mathbb{Z}_+^n\}. \tag{290}$$

The problem can be rewritten as

$$z_{\mathrm{IP}} = \max\{c^T x : A^1 x \leq b^1, A^2 x \leq b^2, x \in \mathbb{Z}_+^n\}, \tag{291}$$

where $A = \begin{pmatrix} A^1 \\ A^2 \end{pmatrix}, b = \begin{pmatrix} b^1 \\ b^2 \end{pmatrix}$.

$A^1 x \leq b^1$ are m_1 "complicating constraints" and
$A^2 x \leq b^2$ are m_2 "nice constraints".

Now for any $\lambda \in \mathbb{R}_+^{m_1}$, the problem (LR($\lambda$))

$$z_{\mathrm{LR}}(\lambda) = \max\{c^T x + \lambda^T (b^1 - A^1 x) : A^2 x \leq b^2, x \in \mathbb{Z}_+^n\} \tag{292}$$

is called the *Lagrangian relaxation* of (IP) with respect to $A^1 x \leq b^1$.

Lagrangian Relaxation

Proposition: LR(λ) is a relaxation of (IP) for all $\lambda \geq 0$, i.e. $z_{\mathrm{LR}}(\lambda) \geq z_{\mathrm{IP}}$ for all $\lambda \geq 0$.

Definition: The least upper bound available from the infinite family of relaxations $\{(LR(\lambda)\}_{\lambda \geq 0}$ is $z_{\mathrm{LR}}(\lambda^*)$, where λ^* is an optimal solution to the problem (LD)

$$z_{\mathrm{LD}} = \min_{\lambda \geq 0} z_{\mathrm{LR}}(\lambda). \tag{293}$$

Problem (LD) is called the *Lagrangian dual* of (IP) with respect to the constraints $A^1 x \leq b^1$.

Course Syllabus

Cutting-Plane Algorithm

The method is derived from the simplex method.

Cutting-plane (or simply a *cut*) is a linear constraint that does not exclude any integer feasible solution.

1. Solve the linear programming relaxation.
2. If the optimal solution is integer, go to Step 5.
3. Select the variable which optimal value is not integer and build a Gomory cut using (277). It is added to simplex tableau as

$$-\sum_{j=1}^{n} \pi'_j x_j + x_{n+1} = -\pi'_0, \qquad (294)$$

   where x_{n+1} is a slack variable.
4. Use the dual simplex algorithm to obtain the optimal solution. Go to Step 2.
5. End

Branch and Bound Algorithm

It is an enumerative algorithm.

Proposition: Let the problem $z_{\text{IP}} = \max\{c^T x : x \in S\}$ be given. Let $S = S_1 \cup S_2 \cup \ldots \cup S_K$ be a decomposition of S into smaller sets, and let $z^k = \max\{c^T x : x \in S_k\}$ for $k = 1, 2, \ldots, K$. Then $z_{\text{IP}} = \max_k z^k$.

Notation:

M is a sequence of problems to be solved in particular branches of an enumeration tree,
x^* is the best found integer solution,
$z^* = c^T x^*$ is the best objective value.

Algorithm:

1. *Initial settings*
   $M = (LP), LP$ is a linear programming relaxation,
   x^* is not defined,
   $z^* = -\infty$.

Branch and Bound Algorithm

2. *Selection of the problem to be solved*
   If $M = ()$ then go to Step 5
   else select the last problem in the sequence M.
3. *Solution of selected problem*
   (a) If no feasible solution exists then remove the problem from M and go to Step 2.
   (b) If the optimal solution x^0 is found with the objective value z^0 then
      (b1) if $z^0 \leq z^*$ then remove the problem from M and go to Step 2,
      (b2) if $z^0 > z^*$ and x^0 is integer then set $x^* = x^0, z^* = z^0$, remove the problem from M and go to Step 2,
      (b3) if $z^0 > z^*$ and x^0 is not integer then go to Step 4.

Branch and Bound Algorithm

4. *Branching*

   Select the variable x_k the optimal value of which is not integer. Copy the last solving problem and add it to the end of the sequence M together with the constraint

   $$x_k \leq \lfloor x_k^0 \rfloor. \tag{295}$$

   Add the constraint

   $$x_k \geq \lfloor x_k^0 \rfloor + 1 \tag{296}$$

   to the last but one problem in M.
   Go to Step 2.

5. *End*

   Print the optimum integer solution x^* and the optimum objective value z^*.

Exact Methods

Branch and Bound Algorithm

Proposition: The enumeration tree can be pruned at the node if any one of the following three conditions holds:

- problem is infeasible,
- optimal solution x^0 is integer,
- it is valid $z^0 \leq z^*$.

In case of binary enumeration tree (if n binary variables are given), n is a maximal depth of the tree and 2^n is a maximal number of leaves of the tree.

Branch and Bound Algorithm

Node Selection

- A priori rules
  - depth-first search plus backtracking (LIFO)
  - breadth-first search
- Adaptive rules
  - best upper bound

Branching Variable Selection

- Most infeasible branching
- Strong branching

Branch and Bound Algorithm

Improvements

- Branch and Cut Method
  Branch and Bound Method with Cutting Plane Method are combined to tighten the linear programming relaxations at nodes of enumeration tree.
- Branch and Price Method
  It is used for IP and MIP problems with many variables. The method is a hybrid of Branch and Bound method and Column Generation Algorithm.

Course Syllabus

Computational Complexity

Complexity of Algorithms

The size of an instance:

- linear programming
  $m \ldots$ number of constraints
  $n \ldots$ number of variables
- graph modelling
  $|U| \ldots$ number of nodes
  $|E| \ldots$ number of arcs

Computational complexity of the algorithm is the function of the size of an instance that the algorithm solves, e.g. $f(n)$.

Complexity of Algorithms

Let elementary computer operation take 1 ns. The following table shows the growth of computational time for various functions depending on a size of an instance.

$f(n)$	n (size of instance)				
	10	20	50	100	1000
n	10 ns	20 ns	50 ns	100 ns	1 µs
$n \log n$	10 ns	26 ns	85 ns	200 ns	3 µs
n^2	100 ns	400 ns	2.5 µs	10 µs	1 ms
n^3	1 µs	8 µs	125 µs	1 ms	1 s
2^n	1 µs	1 ms	13 days	10^{13} years	-
3^n	59 µs	4 s	10^7 years	-	-
$n!$	4 ms	77 years	-	-	-

Complexity of Algorithms

We are interested in the asymptotic rate of growth of the complexity of the algorithm.

Definition: Let $f(n), g(n)$ be functions from the positive integers to the positive reals.

- We write $f(n) = O(g(n))$ if there exists a constant $c > 0$ such that, for large enough n, $f(n) \leq cg(n)$ (the big O notation).
- We write $f(n) = \Omega(g(n))$ if there exists a constant $c > 0$ such that, for large enough n, $f(n) \geq cg(n)$ (the big omega notation).
- We write $f(n) = \Theta(g(n))$ if there exist constants $c, c' > 0$ such that, for large enough $n, cg(n) \leq f(n) \leq c'g(n)$ (the big theta notation).

Polynomial algorithms: $n, n^2, n^3, \log n, n \log n$

Non-polynomial algorithms: $2^n, e^n, n!$

Complexity Classes

Class $\mathcal{P}$

It is a class of decision problems that can be solved in polynomial time. The decision problem of size n is in $\mathcal{P}$ if there exists the algorithm with $f(n) = O(n^p)$ fore some fixed p.

Class $\mathcal{PO}$

It is a class of optimization problems that can be solved in polynomial time. The optimization problem of size n is in $\mathcal{PO}$ if there exists the algorithm with $f(n) = O(n^p)$ fore some fixed p.

Some problems solvable in polynomial time.

- Minimal spanning tree.
- Shortest path problem.
- Maximal flow problem.
- Assignment problem.
- Linear programming problem.

Complexity Classes

Class $\mathcal{NP}$

It is a class of decision problems that can be solved by a *nondeterministic polynomial algorithm* consisting in two stages:

1. Guessing (nondeterministic) stage
   The solution is generated.
2. Checking stage
   It is proved by a polynomial algorithm whether the solution is feasible.

The decision problem is in $\mathcal{NP}$ if a positive decision can be checked in polynomial time.

If a problem is in $\mathcal{P}$ then it is in $\mathcal{NP}$, i.e. $\mathcal{P} \subset \mathcal{NP}$.

Examples of $\mathcal{NP}$-solvable problem: finding of the Hamiltonian cycle in given graph, finding of a feasible solution of MIP, ...

Complexity Classes

Class $\mathcal{NPO}$

The optimization problem is in $\mathcal{NPO}$ if its decision version is in $\mathcal{NP}$.

If a problem is in $\mathcal{PO}$ then it is in $\mathcal{NPO}$, i.e. $\mathcal{PO} \subset \mathcal{NPO}$.

Definition: Decision problem A is *polynomially reducible* to decision problem B if there exists a polynomial function transforming definition of problem A to definition of problem B such that from the solution of B, it is possible to derive the solution of A.

If decision problem A is reducible to decision problem B then, if we have the algorithm to solve B, it can be used to solve A.

Decision problem A is a special instance of decision problem B, i.e. B is more general and, therefore, more difficult.

Complexity Classes

Proposition: If A is polynomially reducible to B and $B \in \mathcal{P}$, then $A \in \mathcal{P}$.
Proposition: If A is polynomially reducible to B and $B \in \mathcal{NP}$, then $A \in \mathcal{NP}$.

Class $\mathcal{NPC}$

Decision problem $A \in \mathcal{NP}$ is said to be $\mathcal{NP}$-*complete* if all problems in $\mathcal{NP}$ can be polynomially reduced to A.

Proposition: If $A \in \mathcal{NPC}$ is polynomially reducible to $B \in \mathcal{NP}$, then $B \in \mathcal{NPC}$.
Corollary: In order to prove that a problem is $\mathcal{NP}$-complete, we must show:

- that the problem is in $\mathcal{NP}$ and
- that any known $\mathcal{NP}$-complete problem is reducible to the problem.

Complexity Classes

Proposition: If $\mathcal{P} \cap \mathcal{NPC} \neq \varnothing$ then $\mathcal{P} = \mathcal{NP}$.

Corollary: If there is a polynomial algorithm to solve any $\mathcal{NP}$-complete problem then using the reducibility we will be able to solve all problems in $\mathcal{NP}$ in polynomial time.

Examples of $\mathcal{NP}$-complete problems (some of them are binary versions of optimization problems):

- Binary programming feasibility problem.
- Set partitioning feasibility problem.
- Knapsack lower-bound feasibility problem.
- Finding of Hamiltonian cycle.
- Travelling salesman upper-bound feasibility problem.
- Quadratic assignment upper-bound feasibility problem.
- Partition problem.

Complexity Classes

Class $\mathcal{NPH}$

An optimization problem is $\mathcal{NP}$-*hard* if its decision version is in $\mathcal{NPC}$.

Examples of $\mathcal{NP}$-hard problems:

- IP problem.
- Knapsack problem.
- TSP.
- Minimal Steiner tree.
- Quadratic assignment problem.
- Container transportation problem.

Course Syllabus

Approximation algorithms:

- Heuristic
  It is a procedure that determines good or near-optimal solutions to a specific optimization problem.
- Metaheuristic
  It is an approach that can be adapted to solve a wide variety of problems.

Basic principles of using the approximation method:

- it is used to solve $\mathcal{NPC}$ and $\mathcal{NPH}$ problems,
- it does not guarantee that an optimal solution will be found (it provides so called suboptimal solution),
- it is a polynomial algorithm,
- it can be easily designed to solve a specific problem.

Heuristics for TSP

Classification of algorithms:

- Constructive heuristics.
- Merge heuristics.
- Improvement heuristics.

Assumption: the matrix of distances between all pairs of nodes is given (complete graph is defined)

The Nearest Neighbor Algorithm

1. Select any node as the initial one of the tour.
2. Find the nearest node (not selected before) to the last node and add it to the tour. If it is impossible (all nodes have been selected) then add the initial node to the tour (Hamiltonian cycle is created) and go to Step 4.
3. Go to Step 2.
4. End.

Savings Algorithm (Clarke and Wright)

1. Compute savings

$$s_{ij} = c_{i1} + c_{1j} - c_{ij} \quad \text{for} \begin{array}{l} i = 2, 3, \ldots, n \\ j = 2, 3, \ldots, n \end{array} \; i \neq j. \tag{297}$$

2. Create $(n-1)$ vehicle routes $(1, i, 1)$ for $i = 2, 3, \ldots, n$ and order the savings in a non-increasing fashion.

Parallel version

3. (Best feasible merge)
   Starting from the top of savings list, execute the following. Given a saving s_{ij}, determine whether there exist two routes, one containing arc $(1, j)$ and the other containing $(i, 1)$, that can feasibly be merged. If so, combine these two routes by deleting $(1, j)$ and $(i, 1)$ and introducing (i, j).

Savings Algorithm (Clarke and Wright)

Sequential version

3. (Route extension)
   Consider the route $(1, i, \ldots, j, 1)$. Determine the first saving s_{ki} or s_{jl} such that k and l are included in other routes containing arc $(k, 1)$ or containing arc $(1, l)$.
4. Implement the merge and repeat this operation until Hamiltonian cycle is created.

Insertion Algorithm

1. Select any node as the initial one, e.g. node 1.
2. Find the farthest node s to the initial one and create the vehicle route $(1, s, 1)$.
3. Execute the most effective insertion of not-included nodes to existing route (minimizing the increase of the length of the route) until Hamiltonian cycle is created.

Double Spanning-Tree Heuristic

Let a complete graph $G = \{U, E\}$ be given.

1. Find the minimal spanning tree $G' = \{U, E'\}$ of G.
2. Construct the multigraph G^* from G' by duplicating each arc from E'.
3. Find an Eulerian cycle Q on G^*.
4. Delete all node repetitions from Q except for the final return to the first node. The resulting node sequence T is a Hamiltonian route on G.

Christofides' Heuristic (Spanning-Tree/Perfect-Matching)

Let a complete graph $G = \{U, E\}$ be given.

1. Find the minimal spanning tree $G' = \{U, E'\}$ of G.
2. Find the minimal perfect matching on the induced subgraph $G(U_0)$ of G, where $U_0 \subseteq U$ is the set of nodes of U that are of odd degree in G'. Let M be the arc set of the perfect matching.
3. Find an Eulerian cycle Q on the multigraph $G^* = \{U, E' \cup M\}$.
4. Delete all node repetitions from Q except for the final return to the first node. The resulting node sequence T is a Hamiltonian route on G.

Cycle Merging Heuristic

Let a complete graph $G = \{U, E\}$ be given.

1. Find the initial system of cycles $\mathcal{F}$ (e.g. using minimal perfect matching; if the size of U is odd, one of cycles contains 3 nodes).
2. Merge two cycles α^* and β^* using the following metrics:
$$D_{\alpha^*\beta^*} = \min_{\alpha,\beta\in\mathcal{F}} D_{\alpha\beta} = \min_{\substack{i,k\in\alpha \\ j,l\in\beta}}(c_{ij} + c_{kl} - c_{ik} - c_{jl}). \qquad (298)$$
Let γ be the cycle created by merging operation.
3. Exclude α^* and β^* from $\mathcal{F}$, include γ in $\mathcal{F}$.
4. If γ is not the Hamiltonian cycle then go to Step 2.

Exchange Heuristic (Lin & Kernighen)

Let a complete graph $G = \{U, E\}$ be given.

1. Let the Hamiltonian cycle be found using any constructive or merge heuristic.
2. Exchange two non-incident arcs from the route for other two non-incident arcs to obtain the Hamiltonian cycle.
3. If the exchange operation improves the solution, realize it.
4. Repeat the process of all possible exchanges while any improvement is achieved. Terminate the process when no improvement is possible.
5. Achieved Hamiltonian cycle is the local optimal (2-opt) route.

Metaheuristics

Notation in algorithms:

x is a feasible solution to the given problem

X is a feasible solution space, i.e. a set of all x

$N(x)$ is a neighborhood of solution x (a set of close solutions)

$f(x)$ is a minimization objective function

x^* is the currently best found solution

Local Search (LS)

1. Choose an initial solution $x \in X$ and set $x^* = x$.
2. Define the neighborhood $N(x) \subseteq X$ and evaluate all solutions.
3. Let x' be the best solution from $N(x)$.
   If $f(x') < f(x^*)$ then set $x^* = x'$ and $x = x'$, stop otherwise.
4. If the stopping rule is not met go to Step 2.
5. Solution x^* is a local minimum solution.

Tabu Search (TS)

TL is the sequence of forbidden solutions (Tabu List)
$MaxSize$ is the maximal size of Tabu List

1. Choose an initial solution $x \in X$ and set $x^* = x$. Adjust $TL = \{x\}$.
2. Define the neighborhood $N(x) \subseteq X \setminus TL$ and evaluate all solutions.
3. Let x' be the best solution from $N(x)$. Set $x = x'$. If $f(x') < f(x^*)$ then set $x^* = x'$.
4. $TL = TL \cup \{x\}$. If $|TL| > MaxSize$ then remove the first solution from TL.
5. If the stopping rule is not met go to Step 2.
6. Solution x^* is the best found solution.

Threshold Accepting Algorithm (TA)

T is the threshold value for accepting worse solutions
T_0 is the initial value of threshold
$r \in (0,1)$ is the rate of threshold reduction

1. Choose an initial solution $x \in X$ and set $x^* = x$. Adjust $T = T_0$.
2. Repeat n-times:
   - choose $x' \in N(x)$,
   - if $f(x') - T < f(x)$ then $x = x'$,
   - if $f(x') < f(x^*)$ then $x^* = x'$.
3. If the stopping rule is not met then execute the reduction $T = rT$ and go to Step 2.
4. Solution x^* is the best found solution.

Simulated Annealing Method (SIAM)

T is the temperature value

T_0 is the initial temperature value

$r \in (0, 1)$ is the rate of temperature reduction (cooling rate)

1. Choose an initial solution $x \in X$ and set $x^* = x$.
   Adjust $T = T_0$.
2. Repeat n-times:
   - choose $x' \in N(x)$,
   - if $f(x') < f(x)$ then $x = x'$,
   - if $f(x') \geq f(x)$ then $x = x'$ with the probability $e^{-\frac{\Delta}{T}}$, where $\Delta = f(x') - f(x)$,
   - if $f(x') < f(x^*)$ then $x^* = x'$.
3. If the stopping rule is not met (or the process has not yet frozen) then execute the reduction $T = rT$ and go to Step 2.
4. Solution x^* is the best found solution.

Genetic Algorithm (GA)

Definitions:

Population $R \subseteq X$ is a finite set of feasible solutions.
Fitness value $f(x)$ is the evaluation of solution $x \in R$.
Parents selection is a selection of certain pair of solutions (parents) $x, y \in R$ based on their fitness value.
Crossover is an operation of combining parents to produce one or two new solutions (*offspring*).
Mutation is a random modification of the offspring.

The idea of parents selection is to choose better solution with higher probability. Let us assume fitness $f(x)$ is maximized. Then the probability of the selection of parent x is given as

$$\frac{f(x)}{\sum\limits_{\forall y \in R} f(y)} \tag{299}$$

Genetic Algorithm (GA)

Let parent x be encoded as $x_1 x_2 \dots x_r$, parent y be encoded as $y_1 y_2 \dots y_r$ and child z be encoded as $z_1 z_2 \dots z_r$.

- Crossover of parents
  - 1-point crossover
    Child #1: $x_1 \dots x_p y_{p+1} \dots y_r$,
    Child #2: $y_1 \dots y_p x_{p+1} \dots x_r$.
  - 2-point crossover
    Child #1: $x_1 \dots x_p y_{p+1} \dots y_q x_{q+1} \dots x_r$,
    Child #2: $y_1 \dots y_p x_{p+1} \dots x_q y_{q+1} \dots y_r$.
  - Uniform crossover
    Child: $z_1 z_2 \dots z_r$, where $z_i \in \{x_i, y_i\}, i = 1, 2, \dots, r$.
- Mutation of a child
  - 1-point mutation
    Modified child: $z_1 \dots z_{p-1} z_p^* z_{p+1} \dots z_r$, where $z_p^* \neq z_p$.
  - 2-point mutation
    Modified child: $z_1 \dots z_{p-1} z_q z_{p+1} \dots z_{q-1} z_p z_{q+1} \dots z_r$.

Ant Colony Optimization (ACO)

- The method is based on *swarm intelligence*.
- Each ant tries to find a route between its nest and a food source.
- On the path, ants lay down a *pheromone* trail.
- Ants prefer to take those paths where there is a larger amount of pheromone.
- The pheromone trails on the longer paths *evaporate* faster than on the shorter paths.
- Pheromone evaporation also has the advantage of avoiding the convergence to a local optimal solution.
- The idea of the ACO is to mimic this behavior with "simulated ants" (agents) walking around the graph representing the problem to solve.