Decision Making and Optimization

Master in Data Analytics for Business

Lisbon School
of Economics
& Management
Universidade de Lisboa

2025-2026

Integer Linear Programming

Knapsack Problem

Knapsack Problem

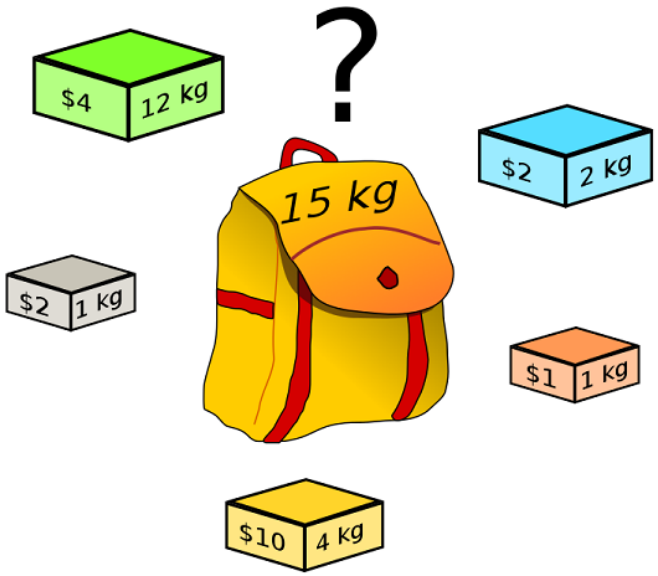
In Knapsack type problems, given n objects, each with an associated cost or utility u_j, $j = 1, \ldots, n$, and weight or volume v_j, $j = 1, \ldots, n$, the decision to be made is whether to select the object j in order to optimize the total utility and not to violate the imposed volume constraint of C.

One must decide if $x_j = 1$, which means that the object j is selected, or if $x_j = 0$, which means that the object j is not selected

The ILP model of the **Binary Knapsack** is

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} u_j x_j \\ \text{s. to:} \quad & \sum_{j=1}^{n} v_j x_j \leq C \\ & x_j \in \{0,1\}, j = 1, \ldots, n \end{aligned}$$

Knapsack Problem: Example



utility =(4,2,2,1,10)
weight=(12,2,1,1,4)
W=15

$$\begin{aligned} \max \quad & 4x_1 + 2x_2 + \\ & 2x_3 + x_4 + 10x_5 \end{aligned}$$

$$\begin{aligned} \text{s. to:} \quad & 12x_1 + 2x_2 + x_3 + \\ & \qquad x_4 + 4x_5 \leq 15 \\ & x_j \in \{0,1\}, j = 1, \ldots, n \end{aligned}$$

$x = (1,1,1,0,0), z = 8, v = 15$
$x = (0,1,1,1,1), z = 15, v = 8$

Knapsack Problem

References:

P. Toth, S. Martello. Knapsack problems: algorithms and computer implementations. Wiley, 1990.
H. Kellerer, U. Pferschy, D. Pisinger. Knapsack Problems. Springer, 2004

The knapsack problem is NP-hard.

There are several variants.

Given

C – capacity of the knapsack,
n – number of different objects,
for $j = 1, \ldots, n$
u_j – utility or cost of object j,
v_j – volume or weight of object j

Knapsack Problem: ILP models

Binary decision variables:

$$x_j = \begin{cases} 1, & \text{if object } j \text{ is selected,} \\ 0, & \text{otherwise,} \end{cases} \quad j = 1, \ldots, n,$$

Binary Knapsack

$$\max \sum_{j=1}^{n} u_j x_j$$

$$s.t. \sum_{j=1}^{n} v_j x_j \leq C$$

$$x_j \in \{0,1\}, j = 1, \ldots, n$$

Subset-sum

$$\max \sum_{j=1}^{n} v_j x_j$$

$$s.t. \sum_{j=1}^{n} v_j x_j \leq C$$

$$x_j \in \{0,1\}, j = 1, \ldots, n$$

Knapsack Problem: ILP models

Integer decision variables:

$$x_j \in \mathbb{N}_0 \text{ number of objects type } j \text{ selected, } j = 1, \ldots, n,$$

Limited Knapsack

$$\max \sum_{j=1}^{n} u_j x_j$$

$$s.t. \sum_{j=1}^{n} v_j x_j \leq C,$$

$$x_j \in \{0, 1, \ldots, \ell_j\}, \quad j = 1, \ldots, n$$

Change Machine

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

$$s.t. \sum_{j=1}^{n} v_j x_j = C,$$

$$x_j \in \{0, 1, \ldots, \ell_j\}, \quad j = 1, \ldots, n$$

$$\ell_j \in \mathbb{N}_0$$

Multiple Knapsack

m – number of different knapsacks,
C_i – capacity of knapsack i, $i = 1, \ldots, m$
Binary decision variables:

$$x_{ij} = \begin{cases} 1, & \text{if object } j \text{ is selected for knapsack } i, \\ 0, & \text{otherwise,} \end{cases}$$

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} u_j x_{ij}$$

$$s.t. \sum_{j=1}^{n} v_j x_{ij} \leq C_i, i = 1, \ldots, m$$

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

$$x_{ij} \in \{0, 1\}, i = 1, \ldots, m, j = 1, \ldots, n$$

Generalized Assignment

u_{ij} – utility obtained from assigning task j to machine i,
v_{ij} – consumption of resource (machine) i by task j,

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} u_{ij} x_{ij}$$

$$s.t. \sum_{j=1}^{n} v_{ij} x_{ij} \leq C_i, i = 1, \ldots, m$$

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

$$x_{ij} \in \{0, 1\}, i = 1, \ldots, m, j = 1, \ldots, n$$

different utility and weights depending on the knapsack selected

Binary Knapsack

From now on, consider the Binary Knapsack.

Binary decision variables:

$$x_j = \begin{cases} 1, & \text{if object } j \text{ is selected,} \\ 0, & \text{otherwise,} \end{cases} \qquad j = 1, \ldots, n,$$

Binary Knapsack

$$\max \sum_{j=1}^{n} u_j x_j$$

$$s.t. \sum_{j=1}^{n} v_j x_j \leq C$$

$$x_j \in \{0,1\}, j = 1, \ldots, n$$

Example:

$n = 6$, $C = 12$, $u = (2, 5, 3, 4, 5, 4)$,
$v = (6, 8, 4, 6, 7, 2)$.

$$\max z = 2x_1 + 5x_2 + 3x_3 + 4x_4 + 5x_5 + 4x_6$$

$$s.t.\ 6x_1 + 8x_2 + 4x_3 + 6x_4 + 7x_5 + 2x_6 \leq 12$$

$$x_j \in \{0,1\}, j = 1, \ldots, 6$$

Binary Knapsack: the Critical index

Assumptions:

$u_j > 0,\ j = 1, \ldots, n,$

$0 < v_j \leq C,\ j = 1, \ldots, n,$

$$\sum_{j=1}^{n} v_j > C > 0$$

$$\frac{u_1}{v_1} \geq \frac{u_2}{v_2} \geq \cdots \geq \frac{u_k}{v_k} \geq \cdots \geq \frac{u_n}{v_n}$$

Critical index:

k such that:

$$\sum_{j=1}^{k-1} v_j \leq C$$

and

$$\sum_{j=1}^{k} v_j > C$$

Example: $n = 6,\ C = 12,\ u = (2, 5, 3, 4, 5, 4),\ v = (6, 8, 4, 6, 7, 2).$

i	1	2	3	4	5	6
$\frac{u_i}{v_i}$	$\frac{2}{6}$	$\frac{5}{8}$	$\frac{3}{4}$	$\frac{4}{6}$	$\frac{5}{7}$	$\frac{4}{2}$
	0.(3)	0.625	0.75	0.(6)	0.7	2

j	1	2	3	4	5	6
$\frac{u_j}{v_j}$	2	0.75	0.7	0.(6)	0.625	0.(3)
i	6	3	5	4	2	1

reorder: $\bar{u} = (4, 3, 5, 4, 5, 2),\ \bar{v} = (2, 4, 7, 6, 8, 6) \rightarrow k = 3$ is the critical index

initial items position $(6, 3, 5, 4, 2, 1)$

Upper bound with the Linear Relaxation:

Linear Relaxation (LR):

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} u_j x_j \\ s.t. \quad & \sum_{j=1}^{n} v_j x_j \leq C \\ & 0 \leq x_j \leq 1, j = 1, \ldots, n \end{aligned}$$

Algorithm:

- Obtain the critical index k
- The optimal LR solution is

$$x_j^* = \begin{cases} 1, & 1 \leq j \leq k-1; \\ \frac{C - \sum_{j=1}^{k-1} v_j}{v_k}, & j = k; \\ 0, & k+1 \leq j \leq n. \end{cases}$$

Example (reordered): $\bar{u} = (4, 3, 5, 4, 5, 2)$, $\bar{v} = (2, 4, 7, 6, 8, 6) \rightarrow k = 3$

the LR optimal solution is $x_{LR}^* = (1, 1, \frac{12-6}{7}, 0, 0, 0) = (1, 1, 0.85, 0, 0, 0)$

with value $z_{LR}^* = 4 + 3 + 5 \times 0.85 = 11.28$

Lower Bounds by feasible solutions:

Greedy Heuristic:

- Obtain the critical index k
- Let

$$x_j = 1,\ j = 1, \ldots, k-1;$$
$$x_j = 0,\ j = k, \ldots, n;$$
$$Z' = \sum_{j=1}^{k-1} u_j.$$

- Take $\underline{Z} = \max\{Z', u_k\}$

Example (reordered):

$\bar{u} = (4, 3, 5, 5, 4, 2)$

$\bar{v} = (2, 4, 7, 8, 6, 6)$

critical index $\rightarrow k = 3$

the greedy feasible solution is

$$\bar{x}_G = (1, 1, 0, 0, 0, 0)$$

with value $Z' = 4 + 3 = 7$

the lower bound is

$\underline{Z} = \max\{Z', u_k\} = \max\{7, 7\} = 7$

Lower Bounds by feasible solutions:

Greedy utility Heuristic:

- Order objects by **non-increasing order of utility**
- For $j = 1, \ldots, n$ take object j
  - $\rightarrow$ if $\sum_{i=1}^{j} v_i \leq C$ then $x_j = 1$, otherwise $x_j = 0$,
  - $\rightarrow Z' = \sum_{i=1}^{j} u_i x_i$.
- Take $\underline{Z} = Z'$

Example:

$u = (2, 5, 3, 4, 5, 4)$

$v = (6, 8, 4, 6, 7, 2)$

Order objects by **non-increasing order of utility**

$\hat{u} = (5, 5, 4, 4, 3, 2)$

$\hat{v} = (8, 7, 6, 2, 4, 6)$

the greedy utility feasible solution is

$\hat{x}_G = (1, 0, 0, 1, 0, 0)$

with value $Z' = 5 + 4 = 9$

and used capacity $8 + 2 = 10$

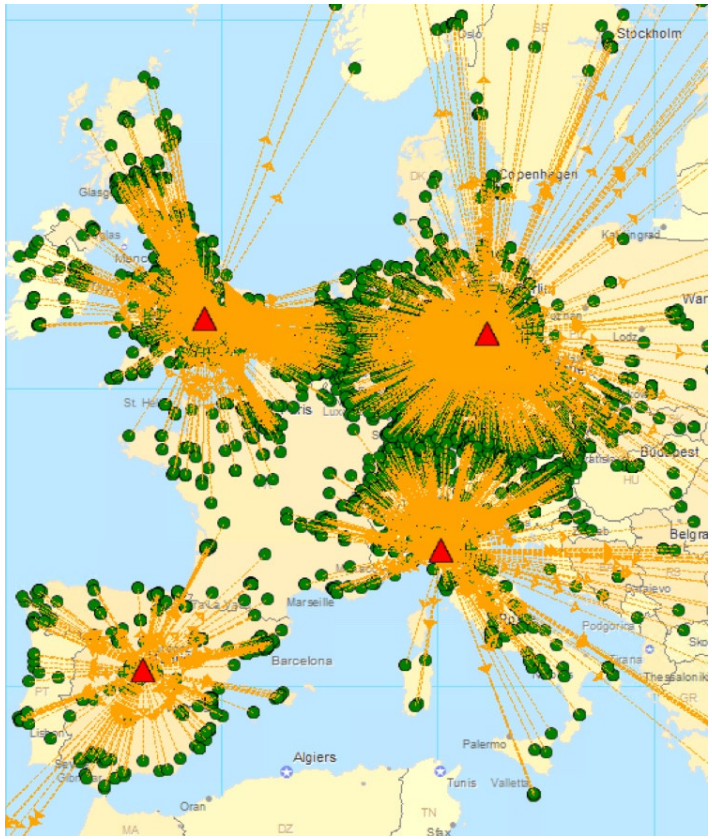
the lower bound is $\underline{Z} = 9$

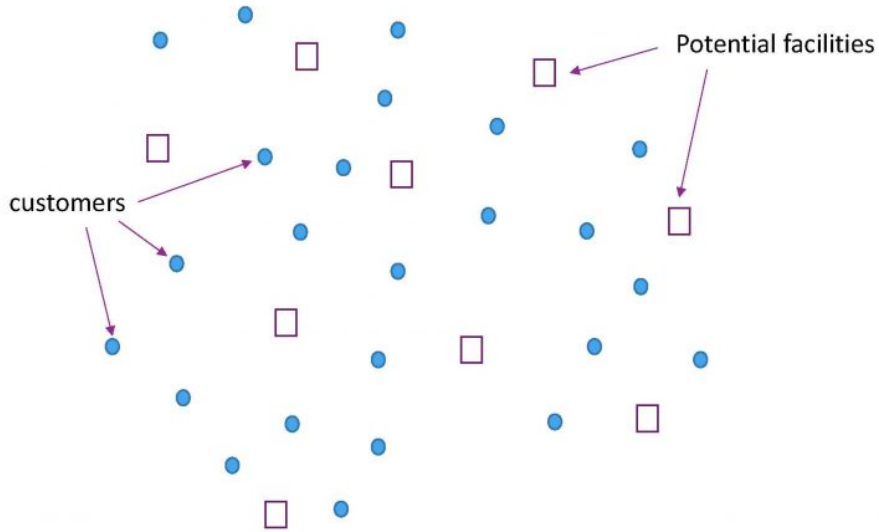
In the example, the ties that occurred in the ordering process were decided without any rules.

Facility Location Problem

Facility Location Problem

This problem involves determining the best location for a facility, like a warehouse, factory, or service center, to serve all the demand and minimize costs





Facility Location Problem: Example

A company plans to build a new warehouse to serve its factories located in three cities: City A, City B, and City C.
The warehouse can be built in any one of the three cities. The transportation cost between a potential warehouse location and each of the cities is shown in the table below (in arbitrary units):

Warehouse Location	To A	To B	To C
A	–	25	19
B	25	–	12
C	19	12	–

The goal is to select the warehouse location that minimizes the total transportation cost to all three cities.

Facility Location Problem

Consider the decision variables

$$y_j = \begin{cases} 1, & \text{if } j \text{ is selected,} \\ 0, & \text{otherwise,} \end{cases} \quad j = 1, \ldots, n,$$

$$x_{ij} = \begin{cases} 1, & \text{if } j \text{ is served by facility in } i, \\ 0, & \text{otherwise,} \end{cases} \quad i, j = 1, \ldots, n,$$

the ILP model of the **Facility Location Problem** is

$$\begin{aligned} \min \quad & \sum_{i,j=1}^{n} c_{ij} x_{ij} \\ \text{s. to:} \quad & \sum_{i=1}^{n} x_{ij} + y_j = 1, \qquad j = 1, \ldots, n \\ & x_{ij} \leq y_j, \qquad i, j = 1, \ldots, n \\ & x_{ij} \geq 0, i, j = 1, \ldots, n \\ & y_j \in \{0, 1\}, j = 1, \ldots, n \end{aligned}$$

Facility Location Problem: ILP Formulation

Sets:

1. $I = \{1, \ldots, m\}-$ customers
2. $J = \{1, \ldots, n\}-$ services

Parameters:

1. $f_j =$ cost of installing a service in j, $j \in J$
2. $c_{ij} =$ cost of customer i being served by the service installed in j, $j \in J$

Decision variables:

$$y_j = \begin{cases} 1, & \text{if a service is installed on } j; \\ 0, & \text{otherwise;} \end{cases} \quad j \in J$$

$$x_{ij} = \begin{cases} 1, & \text{if customer } i \text{ is served by service } j, \\ 0, & \text{otherwise} \end{cases} \quad , i \in I, j \in J$$

Facility Location Problem: ILP Formulation

$$\min \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} f_j y_j$$

$$s.t. : \sum_{j \in J} x_{ij} = 1, \forall i \in I$$

linking constraints

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

$$y_j \in \{0,1\}, j \in J$$

Facility Location Problem: ILP Formulation

Linking constraints - to ensure that a center j can only serve a customer i if a center is installed in j :

$$x_{ij} = 1 \Rightarrow y_j = 1$$

Alternative 1

$$x_{ij} \leq y_j, \forall i \in I, j \in J$$

Alternative 2

$$\sum_{i \in I} x_{ij} \leq m y_j, \forall j \in J$$

Lower and Upper Bounds

Since the problem is formulated as a minimization problem, we have:

Lower bounds

Lower bounds for the optimal value of the problem are provided by relaxations (e.g., linear, Lagrangian, etc.).

Upper bounds

Upper bounds for the optimal value of the problem are provided by feasible solutions (obtained, for example, using heuristics).

Facility Location Problem: Greedy Heuristic

Step 0: **Initialization**

- calculate $z_j = f_j + \sum_{i=1}^{m} c_{ij}$, for all $j = 1, \ldots, n$
- determine j^* such that $z_{j^*} = \min_{j=1,\ldots,n} z_j$
- $S := \{j^*\}$ (solution)
- $C(S) := z_{j^*}$ (solution cost)
- $u_i = c_{ij^*}$ for all $i = 1, \ldots, m$

Step 1: **Selecting a new center**

- for each $j \notin S$ calculate $\rho_j = f_j + \sum_{i=1}^{m} \min(0, c_{ij} - u_i)$
- determine j^* such that $\rho_{j^*} = \min_{j \notin S} \rho_j$
- if $\rho_{j^*} \geq 0$, STOP S contains the obtained solution of cost $C(S)$
- else **(Update)**
  - $S := S \cup \{j^*\}$
  - $C(S) := C(S) + \rho_{j^*}$
  - $u_i := min(u_i, c_{ij^*})$ for all $i = 1, \ldots, m$
  - if $|S| < n$, repeat this step

Facility Location Problem: Example

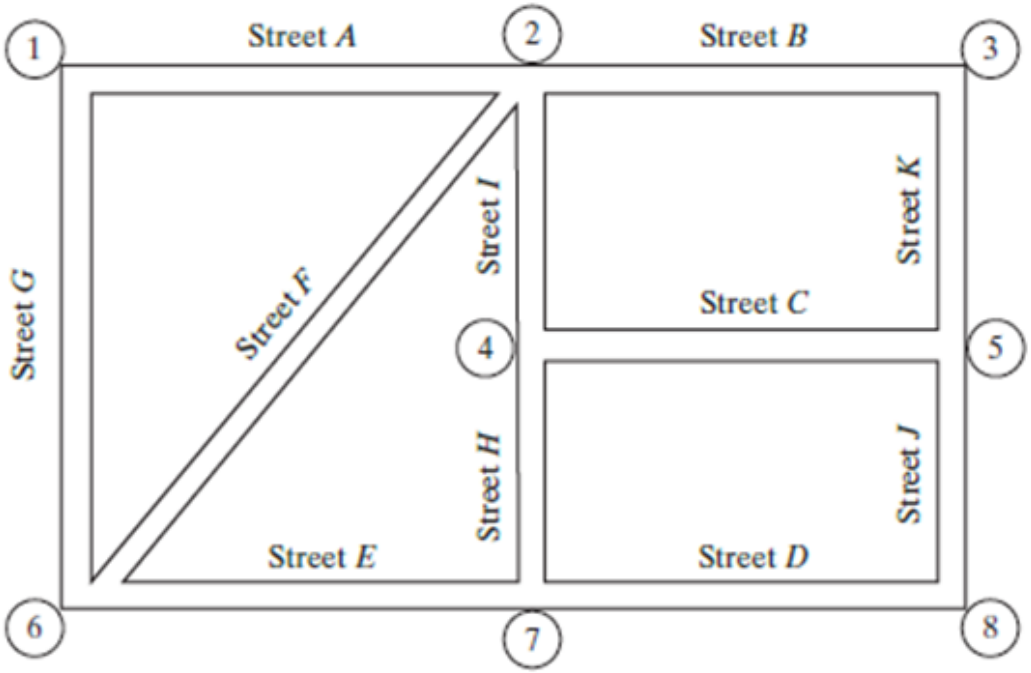
$$m = 4, \quad n = 6, \quad [f_j] = [3\ 2\ 2\ 2\ 3\ 3]$$

$$[c_{ij}] = \begin{bmatrix} 6 & 6 & 8 & 6 & 0 & 6 \\ 6 & 8 & 6 & 0 & 6 & 6 \\ 5 & 0 & 3 & 6 & 3 & 0 \\ 2 & 3 & 0 & 2 & 4 & 4 \end{bmatrix}$$

Set Covering Problem

Set Covering Problem: Example

To promote safety on campus, the Security Department is in the process of installing emergency equipment in selected locations. The department would like to install a minimum number of these devices to serve each of the main campus streets. The figure below shows the main campus roads.



Set Covering Problem

one must decide if $x_j = \begin{cases} 1, & \text{if } j \text{ is selected,} \\ 0, & j \text{ is not selected,} \end{cases} \quad j = 1, \ldots, n,$

the ILP model of the **Set Covering Problem** is

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s. to:} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \ldots, m \\ & x_j \in \{0, 1\}, j = 1, \ldots, n \end{aligned}$$

with $a_{ij} = \begin{cases} 1, & \text{if } j \text{ serves } i, \\ 0, & \text{otherwise,} \end{cases} \quad i = 1, \ldots, m, \; j = 1, \ldots, n,$

Set Covering Problem

Given a matrix of zeros and ones and a cost associated with each column, determine the subset of columns that covers all the rows, i.e. such that for each row there is at least one in one of the selected columns.
The set covering problem is NP-hard.

Sets:

- $N = \{1, \ldots, n\}-$ set of columns
- $M = \{1, \ldots, m\}-$ set of rows
- $N_i = \{j \in N : a_{ij} = 1\}, i \in M.$
- $M_j = \{i \in M : a_{ij} = 1\}, j \in N.$

Parameters:

1. cost c_j associated with each column j, $j \in N$
2. matrix $A = [a_{ij}]$ of zeros and ones, with $a_{ij} = 1$ if column j covers row i and $a_{ij} = 0$ otherwise, for all $i \in M$, $j \in N$

ILP formulation for the Set Covering Problem

Variables:

$$x_j = \begin{cases} 1, & \text{if column } j \text{ is selected} \\ 0, & \text{otherwise;} \end{cases} \quad j \in J$$

$$min \sum_{j \in N} c_j x_j$$

$$s.a : \sum_{j \in N} a_{ij} x_j \geq 1, \forall i \in M$$

$$x_j \in \{0, 1\}, j \in N$$

Note: If constraints $\sum_{j \in N} a_{ij} x_j \geq 1, \forall i \in M$ are replaced by constraints

$$\sum_{j \in N} a_{ij} x_j = 1, \forall i \in M$$

we get the partition problem

Set Covering Problem: Variants

Multiple Set Covering problem

$$min \sum_{j \in N} c_j x_j$$

$$s.a : \sum_{j \in N} a_{ij} x_j \geq b_i, \forall i \in M$$

$$x_j \in \{0, 1\}, j \in N$$

Generalized Set Covering problem

$$min \sum_{j \in N} c_j x_j$$

$$s.a : \sum_{j \in N} a_{ij} x_j \geq b_i, \forall i \in M$$

$$x_j \geq 0 \text{ and integer}, j \in N$$

b_i is an integer greater than or equal to 1. For example, it could represent the minimum number of workers on the shift i.

Set Covering Problem: Preprocessing

Reductions:

1. If there exists $i \in M$ such that $N_i = \emptyset$ then the problem is impossible.
2. If $i \in M$ is such that $N_i = \{j(i)\}$ (i is covered by only one column) then $j(i)$ is in the solution.
3. (Dominance between rows) If $i, \ell \in M$ are such that $N_i \subseteq N_\ell$ then the row ℓ can be eliminated.
4. (Dominance between single columns) If $k, j \in N$ are such that $M_k \subseteq M_j$ and $c_k \geq c_j$ then column k can be removed.
5. (Dominance between columns) If $k, j_1, \ldots, j_s \in N$ are such that $M_k \subseteq \bigcup_{t=1}^{s} M_t$ and $c_k \geq \sum_{t=1}^{s} c_t$ then column k can be removed.
6. (Weak dominance between columns) Let $d_i = min_{j \in N_i} c_j$ and $k \in N$ such that $c_k \geq \sum_{i \in M_k} d_i$ then column k can be removed.

Set Covering Problem: Greedy Algorithm

Greedy Algorithm

Initialise $R \leftarrow M, S \leftarrow \emptyset, t \leftarrow 1$

Step t:

Let $i^* \in R$ be such that $\mid N_{i^*} \mid = \min_{i \in R} \mid N_i \mid$

Choose $j(t)$ such that $f(c_{j(t)}, k_{j(t)}) = \min\{f(c_j, k_j) : j \in N_{i^*} \wedge k_j > 0\}$

where $k_j = \mid M_j \cap R \mid, \forall j \in N_{i^*}$

Make $R \leftarrow R \setminus M_{j(t)}$, $S \leftarrow S \cup \{j(t)\}$,

If $R \neq \emptyset$, set $t \leftarrow t + 1$ and repeat step, otherwise:

Sort the S cover in non-increasing order of costs: $S = \{j_1, \ldots, j_t\}$.

For $i = 1$ to t do:

If $S \setminus \{j_i\}$ is cover then $S \leftarrow S \setminus \{j_i\}$

There are several alternatives to $f(c_j, k_j)$. For example $f(c_j, k_j) = c_j/k_j$.

Set Covering Problem: example

$m = 5, \quad n = 6, \quad [c_j] = [2\ 2\ 3\ 3\ 5\ 7]$

$$[a_{ij}] = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Exercise

Formulate the dual of the linear relaxation of the Set Covering problem. What is the relationship between the optimal value of the Set Covering problem and the value of a feasible solution to the dual?

Traveling Salesperson Problem (TSP)

Example of a Travelling Salesperson Problem

A courier service needs to deliver packages to several locations in a city. The goal is to find the shortest possible route that allows the courier to visit each location exactly once and return to the starting point.

Location	A	B	C
A	5	25	19
B	20	22	12
C	15	35	34

Travelling Salesperson Problem (TSP)

TSP deals with finding the shortest (closed) tour in an n-city situation, where each city is visited exactly once before returning back to the starting point.

The associated TSP model is defined by two pieces of data:

1. the number n of cities,
2. the distances d_{ij} between cities i and j ($d_{ij} = \infty$ if cities i and j are not linked).

The maximum number of tours in an n-city situation is $(n-1)!$ if the network is directed ($d_{ij} \neq d_{ji}$) and half that much if it is not.
Note that $10! = 3\ 638\ 800$

Example of a TSP

The daily production schedule at the Rainbow Company includes batches of white (W), yellow (Y), red (R), and black (B) paints. The production facility must be cleaned between successive batches.

Inter-batch Cleanup Times (in minutes)				
Paint	White	Yellow	Black	Red
White	∞	10	17	15
Yellow	20	∞	19	18
Black	50	44	∞	22
Red	45	40	20	∞

The objective is to determine the sequencing of colors that minimizes the total cleanup time.

Example of a TSP

Solution of the Paint Sequencing Problem by Exhaustive Enumeration

No. of feasible solutions (production loops): $(n-1)! = 3! = 6$

Production loop	Total cleanup time (min)
$W \to Y \to B \to R \to W$	$10 + 19 + 22 + 45 = 96$
$W \to Y \to R \to B \to W$	$10 + 18 + 20 + 50 = 98$
$W \to B \to Y \to R \to W$	$17 + 44 + 18 + 45 = 124$
$W \to B \to R \to Y \to W$	$17 + 22 + 40 + 20 = 99$
$W \to R \to B \to Y \to W$	$15 + 20 + 44 + 20 = 99$
$W \to R \to Y \to B \to W$	$15 + 40 + 19 + 50 = 124$

ILP formulation for the TSP

$$x_{ij} = \begin{cases} 1 & \text{if paint j follows paint i} \\ 0 & \text{otherwise} \end{cases} \qquad i,j = W,Y,B,R; \quad i \neq j$$

$$\min \sum_{i,j=W,Y,B,R;i\neq j} c_{ij}x_{ij}$$

$$\sum_{i=W,Y,B,R} x_{ij} = 1, \qquad j = W,Y,B,R,$$

$$\sum_{i=W,Y,B,R} x_{ji} = 1, \qquad j = W,Y,B,R,$$

$$x_{ij} \in \{0,1\}, \qquad i,j = W,Y,B,R, i \neq j$$

????

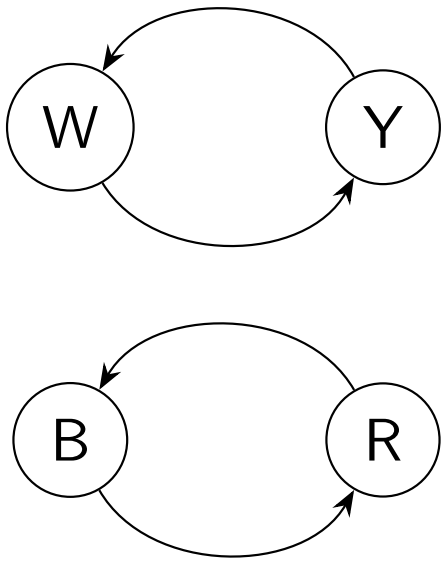
Example of a TSP

The solution

$$x_{WY} = x_{YW} = x_{BR} = x_{RB} = 1$$

satisfies all the previous constraints

$$\begin{aligned}
x_{WY} + x_{WB} + x_{WR} &= 1 \\
x_{YW} + x_{YB} + x_{YR} &= 1 \\
x_{BY} + x_{BR} + x_{RW} &= 1 \\
x_{RW} + x_{RY} + x_{RB} &= 1 \\
x_{YW} + x_{BW} + x_{RW} &= 1 \\
x_{WY} + x_{BY} + x_{RY} &= 1 \\
x_{WB} + x_{YB} + x_{RB} &= 1 \\
x_{WR} + x_{YR} + x_{BR} &= 1 \\
x_{ij} \in \{0,1\} \quad \forall i,j \quad i \neq j
\end{aligned}$$



and $\min 10x_{WY} + 17x_{WB} + 15x_{WR} + 20x_{YW} + 19x_{YB} + 18x_{YR} + 50x_{BW} + 44x_{BY} + 22x_{BR} + 45x_{RW} + 40x_{RY} + 20x_{RB}$

ILP formulation for the TSP

Subtour elimination constraints are missing

for example

$$x_{WB} + x_{WR} + x_{YB} + x_{YR} + x_{BW} + x_{BY} + x_{RW} + x_{RY} \geq 1$$

how to establish such constraints?

Dantzig, Fulkerson, Johnson, 1954:

For every set S of cities, add a constraint saying that the tour leaves S at least once. For every $S \subseteq \{1, 2, \ldots, n\}$ with $1 \leq |S| \leq n - 1$:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1$$

This will happen for any tour: eventually, we must go from a city in S to a city not in S. In a solution to the local constraints with subtours, this is violated if we take S to be the set of cities in a subtour

- The formulation with the subtour elimination constraints describe TSP.
- Number of constraints increase exponentially: for n cities, there are $2^n - 2$ subtour elimination constraints! $2^{n-1} - 1$ if we assume $1 \in S$.

Dantzig, Fulkerson, Johnson, 1954:

The complete model is:

$$\min \sum_{i,j\in\{1,2,\ldots,n\};i\neq j} c_{ij}x_{ij}$$

$$\sum_{i\in\{1,2,\ldots,n\}} x_{ij} = 1, \qquad j \in \{1,2,\ldots,n\},$$

$$\sum_{i\in\{1,2,\ldots,n\}} x_{ji} = 1, \qquad j \in \{1,2,\ldots,n\},$$

$$\sum_{i\in S}\sum_{j\notin S} x_{ij} \geq 1 \qquad S \subseteq \{1,2,\ldots,n\},\ 1 \leq |S| \leq n-1$$

$$x_{ij} \in \{0,1\}, \qquad i,j \in \{1,2,\ldots,n\}, i \neq j$$

Miller, Tucker, Zemlin, 1960:

Add variables representing the time at which a city is visited.
For $i = 1, \ldots, n,$, let t_i denote the time at which we visit city i, with $1 \leq t_i \leq n-1$. We leave t_1 undefined.
We want an inequality to encode the logical implication

if $x_{ij} = 1$, then $t_j \geq t_i + 1$ for every pair of cities $i, j \neq 1$.

How do we know that the timing constraints get rid of subtours?

1. For any tour, we can satisfy the timing constraints. If we visit cities $i_1, i_2, \ldots, i_{(n-1)}$, in that order from city 1, set $i_1 = 1, \quad i_2 = 2, \quad \ldots, \quad i_{(n-1)} = n-1$.
2. If there is a subtour, then we can't satisfy the timing constraints.
3. Suppose $x_{ab} = x_{bc} = x_{ca} = 1$ and none of a, b, c are 1. Then we can't satisfy the three constraints $t_b \geq t_a + 1, \quad t_c \geq t_b + 1 \quad t_a \geq t_c + 1$

Miller, Tucker, Zemlin, 1960:

If $x_{ij} = 1$, then $t_j \geq t_i + 1$.
Using the big number M:

$$t_j \geq t_i + 1 - M(1 - x_{ij}) \text{ for some large } M.$$

When $x_{ij} = 1$, this simplifies to $t_j \geq t_i + 1$.
When $x_{ij} = 0$, we get $t_j \geq t_i + 1 - M$, which has no effect on the value of t_i, t_j.

We can check: if we take $M = n$, then any actual tour can satisfy these constraints. The times $t_2, \ldots, t_n$ can be chosen between 1 and $n - 1$, so $t_j \geq t_i + 1 - n$ always holds.

The inequality is

$$t_j \geq t_i + 1 - n(1 - x_{ij})$$

Miller, Tucker, Zemlin, 1960:

The complete model is:

$$\min \sum_{i,j \in \{1,2,\ldots,n\}; i \neq j} c_{ij} x_{ij}$$

$$\sum_{i \in \{1,2,\ldots,n\}} x_{ij} = 1, \qquad j \in \{1, 2, \ldots, n\},$$

$$\sum_{i \in \{1,2,\ldots,n\}} x_{ji} = 1, \qquad j \in \{1, 2, \ldots, n\},$$

$$t_j \geq t_i + 1 - n(1 - x_{ij}) \qquad i, j \in \{1, 2, \ldots, n\},\ i \neq j$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in \{1, 2, \ldots, n\}, i \neq j$$

DFJ versus MTZ

On the one hand:

- DFJ's formulation has $2^{(n-1)} - 1$ extra constraints, plus the $2n$ local constraints.
- MTZ's formulation has only n^2 extra constraints. There are $n - 1$ extra variables, which can be integer variables, but don't need to be.

On the other hand:

- DFJ's formulation has an efficient branch-and-cut approach.
- MTZ's formulation is weaker: the feasible region has the same integer points, but includes more fractional points.

Relaxations for the TSP

The Assigment relaxation:

$$\min \sum_{i,j \in \{1,2,\ldots,n\}; i \neq j} c_{ij} x_{ij}$$

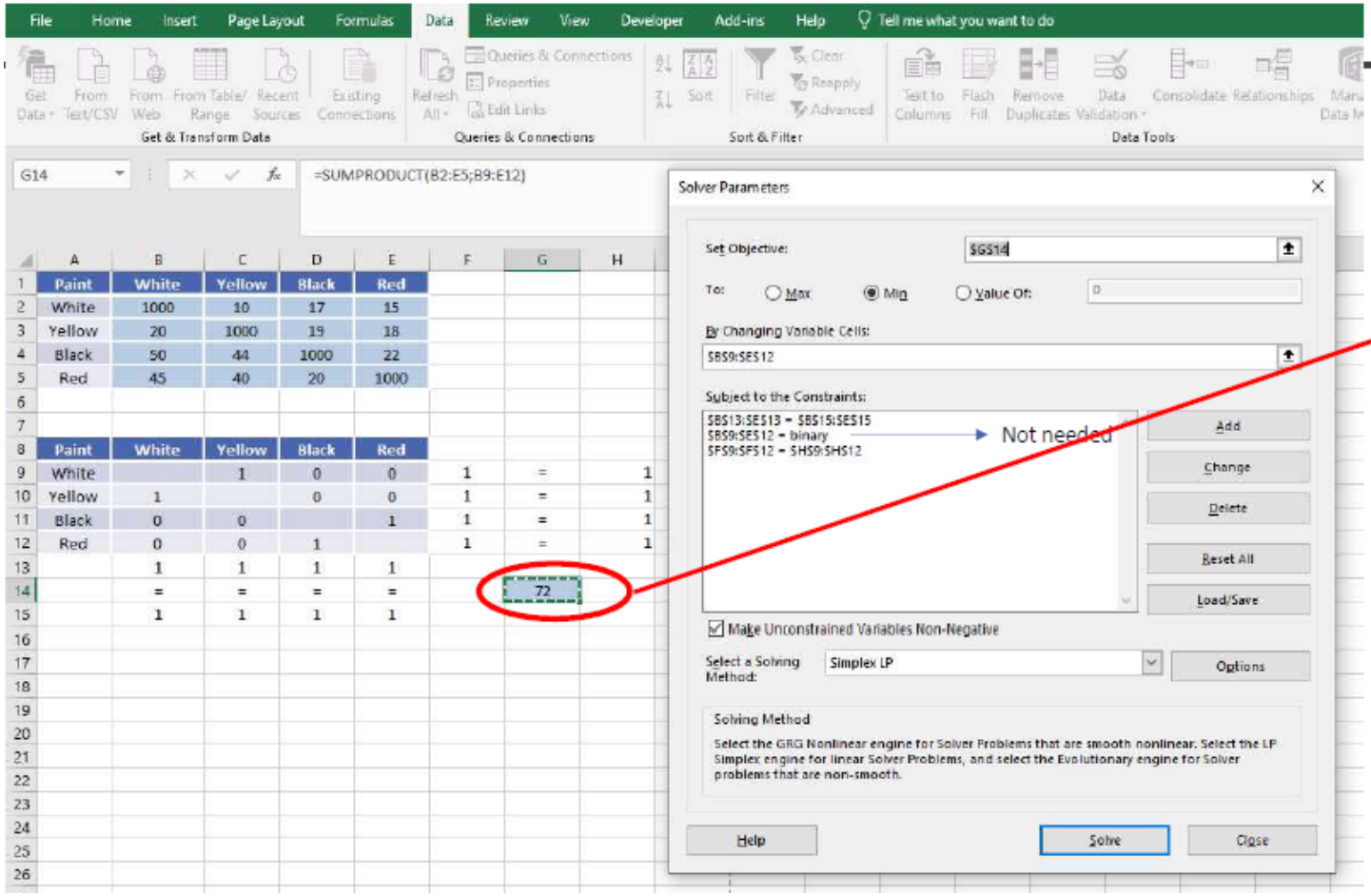
$$\sum_{i \in \{1,2,\ldots,n\}} x_{ij} = 1, \qquad j \in \{1, 2, \ldots, n\},$$

$$\sum_{i \in \{1,2,\ldots,n\}} x_{ji} = 1, \qquad j \in \{1, 2, \ldots, n\},$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in \{1, 2, \ldots, n\}, i \neq j$$

Relaxations for the TSP: the paints example

Solving the Assigment Relaxation using the Solver of the Excel



	A	B	C	D	E	F	G	H
1	Paint	White	Yellow	Black	Red			
2	White	1000	10	17	15			
3	Yellow	20	1000	19	18			
4	Black	50	44	1000	22			
5	Red	45	40	20	1000			
6								
7								
8	Paint	White	Yellow	Black	Red			
9	White		1	0	0	1	=	1
10	Yellow	1		0	0	1	=	1
11	Black	0	0		1	1	=	1
12	Red	0	0	1		1	=	1
13		1	1	1	1			
14		=	=	=	=		72	
15		1	1	1	1			

Assignment problem

Lower bound TSP **72**

Optimal solution by complete enumeration

W -> *Y*-> *B*-> *R*-> *W*

10 + 19 + 22 + 45 = **96**

Constructive heuristics for the TSP: nearest neighbor

Input: $G = (V, A), V = \{1, 2, \ldots, n\}, |V| = n, \text{cost} c_{ij} \rightarrow (i, j) \in A$

Initialization

Arbitrarily choose a city $i \in V$

$L = \{1, 2, \ldots, n\} - \{i\}$ (L set of cities not yet visited)

Iteration

REPEAT

Select in L city j closest to i

Insert the city j immediately after i in the route

Update $i = j$

$L := L - \{i\}$

UNTIL $L = \emptyset$ OR no city can be selected

If possible complete the cycle by going back to the beginning

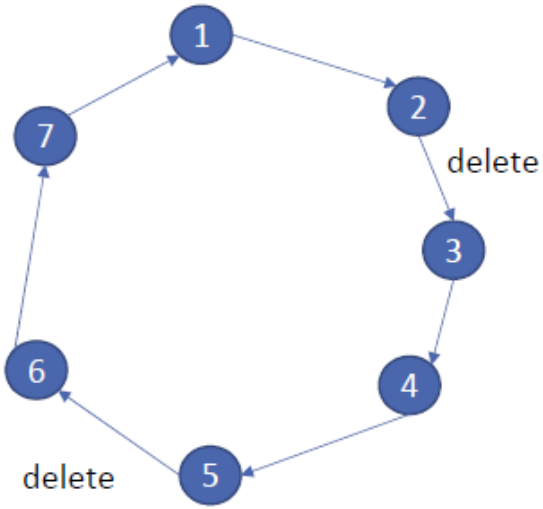
and calculate the total distance

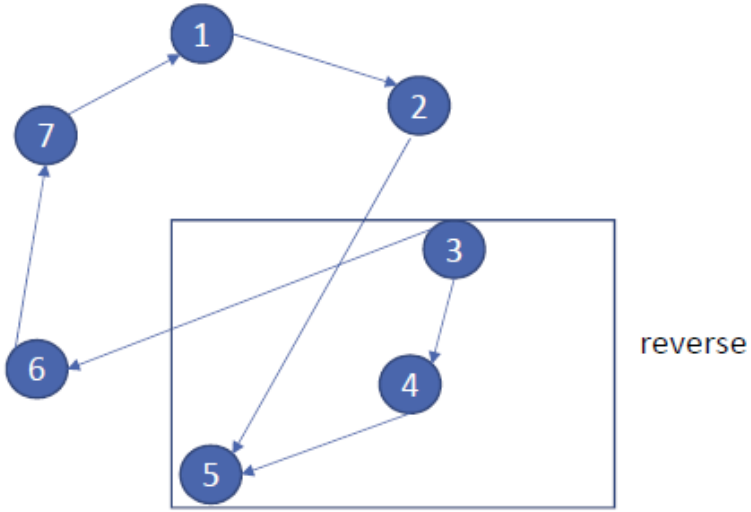
STOP

Improvement Heuristics for the TSP

- Starting from a feasible circuit, try edge swapping that can lead to new lower-cost circuits.
- The algorithm consists of starting with a feasible circuit and swapping r edges until it is no longer possible to improve the solution.

Swapping 2 edges: delete, reverse, connect





Improvement Heuristics for the TSP: 2-optimal

Consider the case of heuristics that perform 2 edge swaps to improve the solution already obtained.

- If you have a circuit and you swap 2 edges that are not consecutive, how many different circuits can you get?

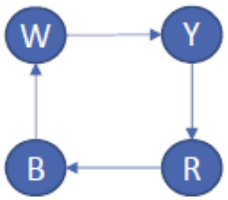
$$\frac{n[(n-1)-2]}{2}$$

- Let $N_2(T)$ be the neighborhood of the circuit T, i.e. $N_2(T)$ is the set of circuits that differ from circuit T on a maximum of 2 (non-consecutive) edges.
- $|N_2(T)| = \frac{n(n-3)}{2} + 1$

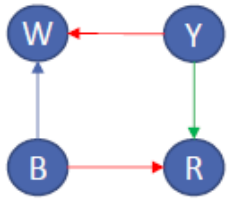
Paints example of swapping 2 edges

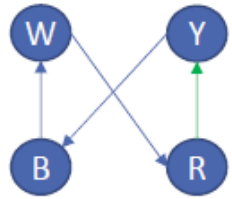
remove

reverse

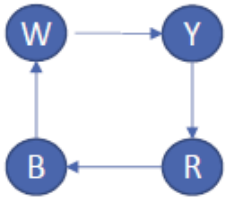


Z=98

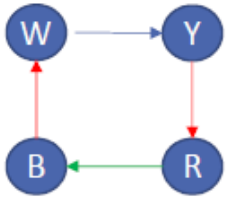


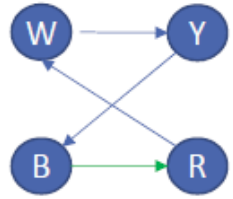


Z=96



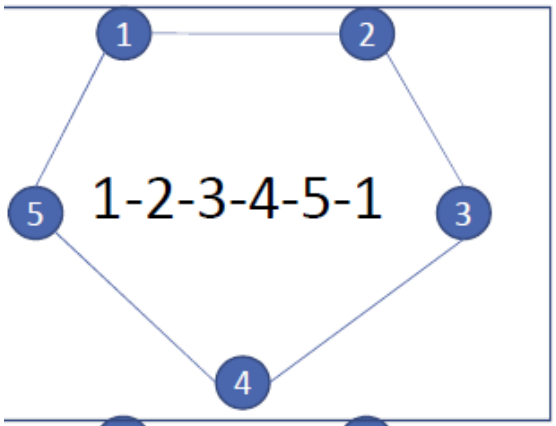
Z=98

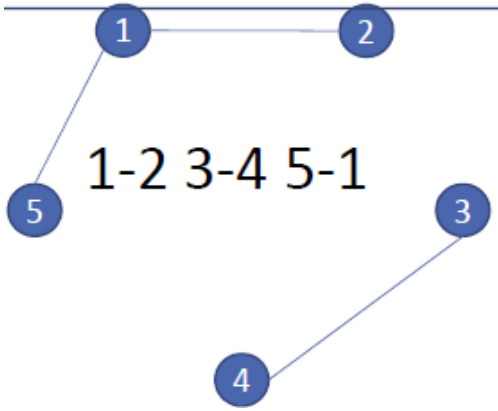


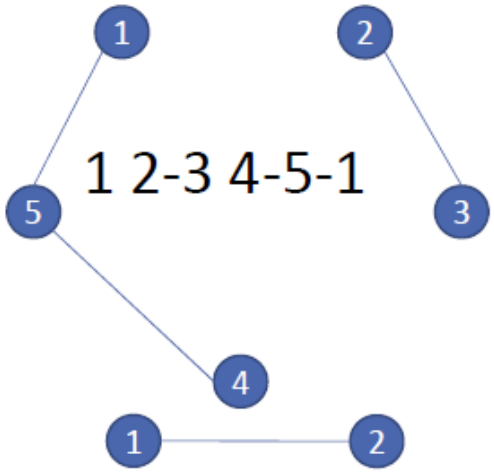


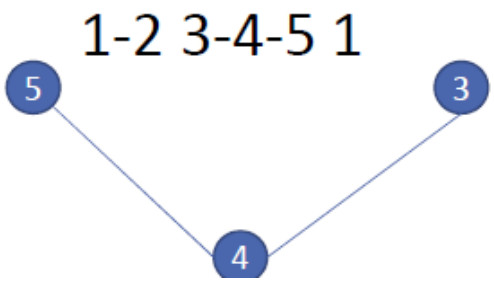
Z=96

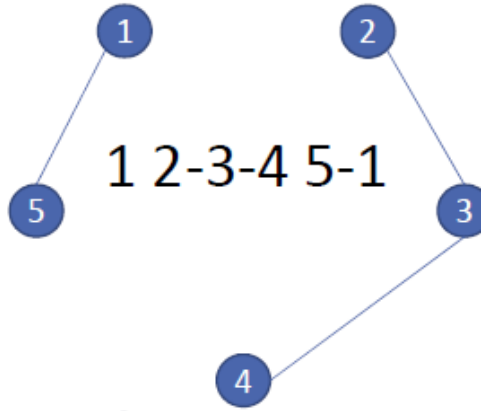
2-opt neighborhood - remove

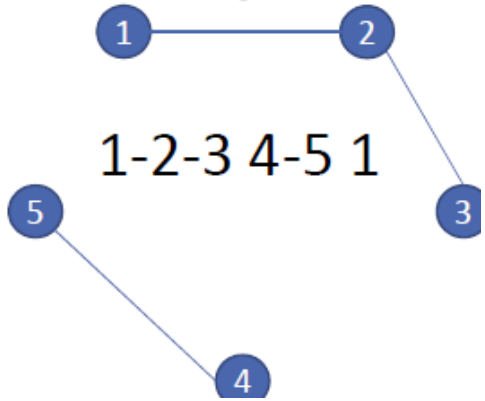




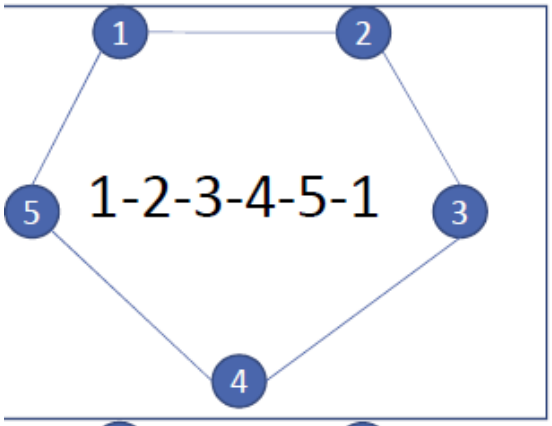


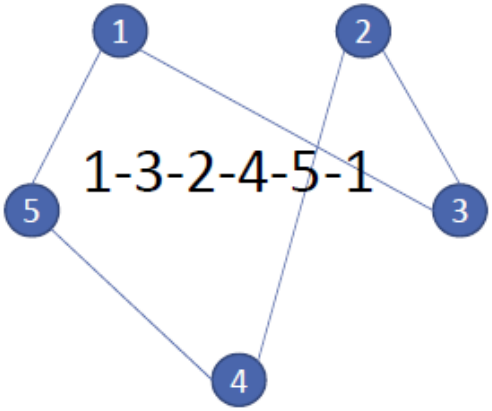


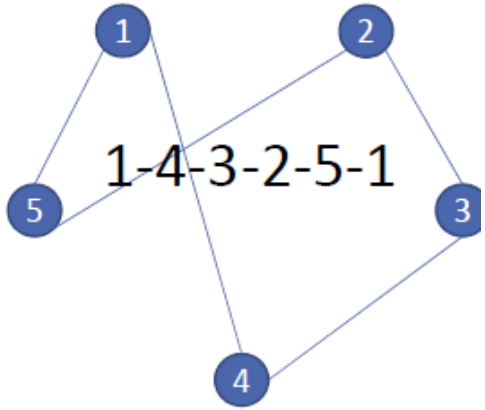


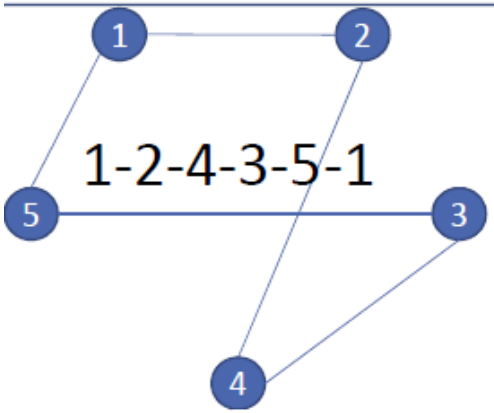


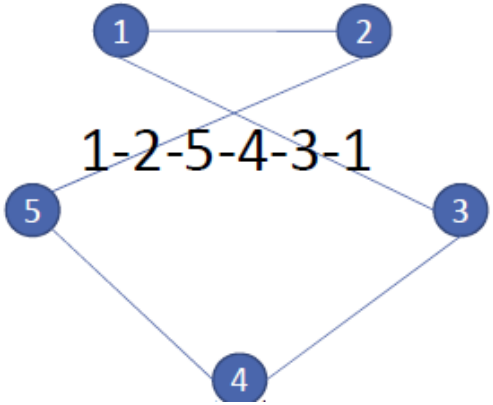
2-opt neighborhood - reverse

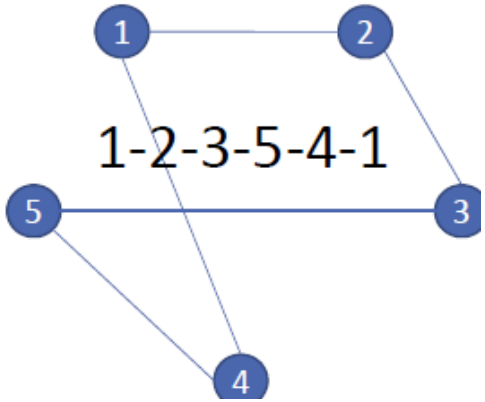












Improvement Heuristics for the TSP

1st version

1. Determine a circuit T.
2. Determine $N_r(T)$ (the set of all possible swaps of r edges) and the cost of all its circuits.
3. Determine a circuit $Q \neq T$ such that Q is the circuit with the minimum cost in $N_r(T) \backslash \{T\}$.
4. If the cost Q is less than the cost T, then do $T := Q$ and return to step 2, otherwise $STOP$, it is not possible to improve the current circuit.

Improvement Heuristics for the TSP

2nd version

1. Determine a circuit T.
2. Sequentially examine the elements $Q \neq T$ of $N_r(T)$ and determine its cost.
3. If cost Q is less than cost T, then do $T := Q$. Return to step 2. If there is no more element to search in $N_r(T)$ then *STOP* (it is not possible to improve the current circuit in the considered neighborhood).

TSP: Exercise

$$[c_{ij}] = \begin{bmatrix} - & 10 & 22 & 12 & 10 \\ & - & 12 & 8 & 13 \\ & & - & 15 & 15 \\ & & & - & 9 \\ & & & & - \end{bmatrix}$$

Sequencing Problem

Example of a Job Sequencing Problem

Jobco uses a single machine to process three jobs. For each job, both the processing time and the due date (in days) are given in the following table. The due dates are measured from zero, the assumed start time of the first job.

Job	Processing time (day)	Due date (day)	Late penalty ($/day)
1	5	25	19
2	20	22	12
3	15	35	34

The objective of the problem is to determine the job sequence that minimizes the late penalty for processing all three jobs.

Sequencing Problem

$$\text{if } x_{ij} = \begin{cases} 1, & \text{if } i \text{ precedes } j, \\ 0, & \text{otherwise,} \end{cases} \quad i, j = 1, \ldots, n,$$

if t_j = start time of job j, $j = 1, \ldots, n$ (measured from time 0)

let δ_j be the processing time of job j, $j = 1, \ldots, n$

the ILP model of the **Sequencing Problem** has the following constraints

$$t_j \geq t_i + \delta_i - M\ (1 - x_{ij}),$$
$$t_i \geq t_j + \delta_j - M\ x_{ij}$$

that model the sequence disjunction:

$$t_j \geq t_i + \delta_i \qquad \text{or} \qquad t_i \geq t_j + \delta_j$$

either job j is after job i or job i is after job j

Sequencing Problem

let d_j be the due date of job j, $j = 1, \ldots, n$

the job j is late if $\quad t_j + \delta_j > d_j$

thus the following constraints

$$t_j + \delta_j - (s_j^+ - s_j^-) = d_j,$$
$$s_j^+, s_j^- \geq 0$$

define that

job j is ahead of schedule if $s_j^- > 0$

job j is behind schedule if $s_j^+ > 0$