Relaxations and Bounds

Lorenzo Castelli, Università degli Studi di Trieste.

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Optimality

Given an ***IP***

$$z = \max\{c(x) : x \in X \subseteq \mathbb{Z}^n\}$$

how can we prove that a given point x^* is optimal?

We need to address this question because we saw that solution enumeration and convex hull identification may not possible, whereas disregarding variables' integrality constraints may not provide useful information.

Therefore we need to find alternatives (i.e., algorithms) to solve an ***IP***.

Bounds

The most common approach to solve ***IP*** problems is to find sequences of bounds until they are "close enough".

Upper bound

If $\boldsymbol{z}$ is the optimal value of an ***IP*** problem, an upper bound is a value $\overline{\boldsymbol{z}}$ such that $\overline{\boldsymbol{z}} \geq \boldsymbol{z}$.

Lower bound

If $\boldsymbol{z}$ is the optimal value of an ***IP*** problem, a lower bound is a value $\underline{\boldsymbol{z}}$ such that $\underline{\boldsymbol{z}} \leq \boldsymbol{z}$.

Ideally, we would like to find $\overline{\boldsymbol{z}}$ and $\underline{\boldsymbol{z}}$ such that $\underline{\boldsymbol{z}} = \boldsymbol{z} = \overline{\boldsymbol{z}}$.

Lower and upper bounds

From a practical point of view, any algorithm will look for a decreasing sequence of upper bounds

$$\overline{z_1} > \overline{z_2} > \ldots > \overline{z_s} \geq z$$

and an increasing sequence of lower bounds

$$\underline{z_1} < \underline{z_2} < \ldots < \underline{z_t} \leq z$$

and stops when

$$\overline{z_s} - \underline{z_t} \leq \epsilon$$

where ϵ is an appropriate non-negative value.

Lower bounds

Lower bound

Every feasible solution $\hat{x} \in X$ provides a lower (or primal) bound $\underline{z} = c(\hat{x}) \leq z$.

For the problem

$$\begin{aligned} \max Z = & 1.00x_1 + 0.64x_2 \\ & 50x_1 + 31x_2 \leq 250 \\ & 3x_1 - 2x_2 \geq -4 \\ & x_1, x_2 \geq 0 \text{ and integer.} \end{aligned}$$

we saw, by rounding the optimal linear solution, that $\hat{x} = (2, 4)$ is a feasible solution such $\underline{z} = c(\hat{x}) = 4.56 \leq z$.

Upper bounds

Findings upper bounds could be less obvious.

The most common idea is to replace a “difficult” $\boldsymbol{IP}$ problem by a simpler optimisation problem, whose optimal value is at least as large as $\boldsymbol{z}$.

The simpler problem can be obtained by “relaxation”, i.e., by

- enlarging the set of feasible solutions so that one optimises over a larger set,
- replacing the max objective function by a function that has the same or a larger value everywhere.

Relaxation

Definition

A problem $(\boldsymbol{RP})\ \boldsymbol{z^R} = \max\{\boldsymbol{f}(\boldsymbol{x}) : \boldsymbol{x} \in \boldsymbol{T} \subseteq \mathbb{R}^{\boldsymbol{n}}\}$ is a relaxation of $(\boldsymbol{IP})\ \boldsymbol{z} = \max\{\boldsymbol{c}(\boldsymbol{x}) : \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{Z}^{\boldsymbol{n}}\}$ if:

(i) $\boldsymbol{X} \subseteq \boldsymbol{T}$, and

(ii) $\boldsymbol{f}(\boldsymbol{x}) \geq \boldsymbol{c}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \boldsymbol{X}$.

Proposition. If $\boldsymbol{RP}$ is a relaxation of $\boldsymbol{IP}$, $\boldsymbol{z^R} \geq \boldsymbol{z}$

If $\boldsymbol{x}^*$ is an optimal solution of $\boldsymbol{IP}$, $\boldsymbol{x}^* \in \boldsymbol{X} \subseteq \boldsymbol{T}$ and $\boldsymbol{z} = \boldsymbol{c}(\boldsymbol{x}^*) \leq \boldsymbol{f}(\boldsymbol{x}^*)$. As $\boldsymbol{x}^* \in \boldsymbol{T}$, $\boldsymbol{f}(\boldsymbol{x}^*)$ is a lower bound on $\boldsymbol{z^R}$, and so $\boldsymbol{z} \leq \boldsymbol{f}(\boldsymbol{x}^*) \leq \boldsymbol{z^R}$. ∎

Hence, $\boldsymbol{z^R}$ is an upper bound!

Linear relaxation

Definition

For the integer program $\max\{\boldsymbol{cx} : \boldsymbol{x} \in \boldsymbol{X} = \boldsymbol{P} \cap \mathbb{Z}^{\boldsymbol{n}}\}$ with formulation $\boldsymbol{P} = \{\boldsymbol{x} \in \mathbb{R}_{+}^{\boldsymbol{n}} : \boldsymbol{Ax} \leq \boldsymbol{b}\}$, the linear programming relaxation is the linear program $\boldsymbol{z}^{\boldsymbol{LP}} = \max\{\boldsymbol{cx} : \boldsymbol{x} \in \boldsymbol{P}\}$.

Since $\boldsymbol{X} = \boldsymbol{P} \cap \mathbb{Z}^{\boldsymbol{n}} \subseteq \boldsymbol{P}$ and the objective function is unchanged, this is clearly a relaxation.

Linear relaxation - example 1

Original *IP* problem

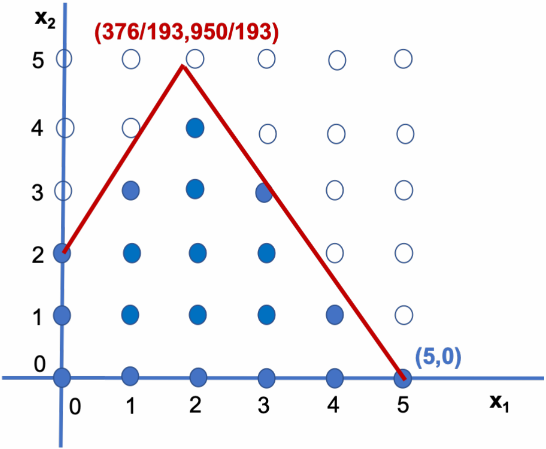
$$\begin{aligned} z = \max & \; 1.00x_1 + 0.64x_2 \\ & 50x_1 + 31x_2 \leq 250 \\ & 3x_1 - 2x_2 \geq -4 \\ & x_1, x_2 \geq 0 \text{ and integer.} \end{aligned}$$

LP relaxation

$$\begin{aligned} z^{LP} = \max & \; 1.00x_1 + 0.64x_2 \\ & 50x_1 + 31x_2 \leq 250 \\ & 3x_1 - 2x_2 \geq -4 \\ & x_1, x_2 \geq 0. \end{aligned}$$

- The optimal integer solution is $(5, 0)$ and $z = 5$
- The optimal solution of the linear relaxation is $(376/193, 950/193)$ and $z^{LP} = 984/193 = 5.098$.

Linear relaxation - example 1



Linear relaxation - example 1

For the ***IP*** problem

$$
\begin{aligned}
z = \max & \mathbf{1.00}x_1 + \mathbf{0.64}x_2 \\
& \mathbf{50}x_1 + \mathbf{31}x_2 \le \mathbf{250} \\
& \mathbf{3}x_1 - \mathbf{2}x_2 \ge -\mathbf{4} \\
& x_1, x_2 \ge \mathbf{0} \text{ and integer.}
\end{aligned}
$$

We know that a lower bound is $\underline{z} = \mathbf{4.560}$ and an upper bound is $\overline{z} = \mathbf{5.098}$. The optimal value z therefore lies within

$$\underline{z} = \mathbf{4.560} \le z \le \mathbf{5.098} = \overline{z}.$$

In fact, $z = \mathbf{5}$. *The information we get by disregarding variables' integrality constraints can be very useful indeed.*

Linear relaxation - example 2

Consider the ***IP*** problem

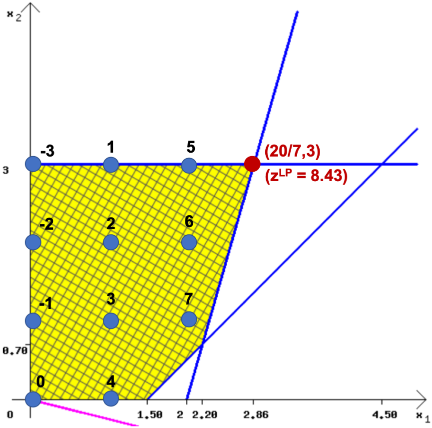
$$
\begin{aligned}
z = \max 4x_1 - x_2 & \\
7x_1 - 2x_2 &\leq 14 \\
x_2 &\leq 3 \\
2x_1 - 2x_2 &\leq 3 \\
x_1, x_2 &\geq 0 \text{ and integer.}
\end{aligned}
$$

It is easy to see that $(2, 1)$ is a feasible solution, thus leading to the lower bound $\underline{z} = 7$. The optimal solution of the linear relaxation is $x^* = (20/7, 3)$ producing the upper bound $\overline{z} = 59/7 = 8.43$. Since all coefficients of the objective function are integer (i.e., $(4, -1)$) the optimal value z must to be integer as well. Hence, we can take as upper bound $\overline{z} = \lfloor 8.43 \rfloor = 8$. Therefore

$$\underline{z} = 7 \leq z \leq 8 = \overline{z}$$

.

Linear relaxation - example 2



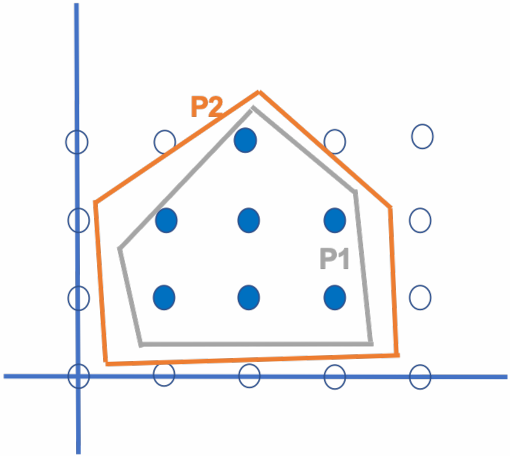
Better formulations

Proposition

Suppose $\boldsymbol{P_1}, \boldsymbol{P_2}$ are two formulations for the $\boldsymbol{IP}$ $\max\{\boldsymbol{cx} : \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{Z}^{\boldsymbol{n}}\}$ with $\boldsymbol{P_1}$ a better formulation than $\boldsymbol{P_2}$, i.e., $\boldsymbol{P_1} \subset \boldsymbol{P_2}$. If $\boldsymbol{z_i^{LP}} = \max\{\boldsymbol{cx} : \boldsymbol{x} \in \boldsymbol{P_i}\}$ for $\boldsymbol{i} = \mathbf{1}, \mathbf{2}$ are the values of the associated linear programming relaxations, then $\boldsymbol{z_1^{LP}} \leq \boldsymbol{z_2^{LP}}$ for all $\boldsymbol{c}$.

The better the formulation the tighter the upper bound.

Better formulations - example



Other propositions

If a relaxation RP is infeasible, the original problem IP is infeasible

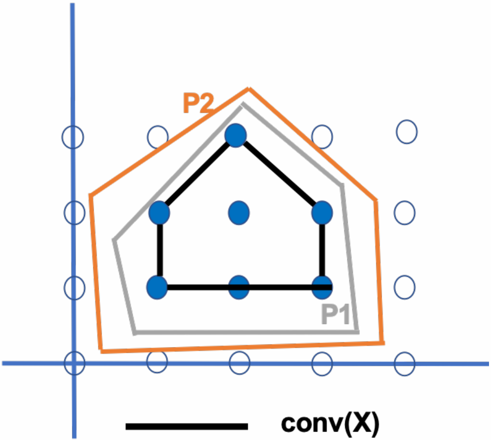
As RP is infeasible, $T = \emptyset$. Since $X \subseteq T$ also $X = \emptyset$.

Let x^* be an optimal solution of RP. If $x^* \in X$ and $f(x^*) = c(x^*)$ then x^* is an optimal solution of IP.

As $x^* \in X, z \geq c(x^*) = f(x^*) = z^R$. As $z \leq z^R$, we obtain $c(x^*) = z = z^R$.

- If the optimal solution x^* of the linear relaxation is integer, x^* is also the optimal solution of the original IP problem.
- The optimal solution of $conv(x)$ is always integer because all extreme points belong to it.

Convex hull – integer solution



Integer solution of linear relaxation

Example 1

$$\max 7x_1 + 4x_2 + 5x_3 + 2x_4$$
$$3x_1 + 3x_2 + 4x_3 + 2x_4 \leq 6$$
$$x_i \in \{0, 1\} \text{ for } i = 1, 2, 3, 4.$$

The linear relaxation has optimal solution $x^* = (1, 1, 0, 0)$. Hence, x^* is the optimal solution of the original binary problem.

Example 2

$$\max 3x_1 + 5x_2$$
$$x_1 \leq 4$$
$$2x_2 \leq 12$$
$$3x_1 + 2x_2 \leq 18$$
$$x_1, x_2 \geq 0 \text{ and integer}$$

The linear relaxation has optimal solution $x^* = (2, 6)$. Hence, x^* is the optimal solution of the original integer problem.

Minimisation problems

In case of the minimisation problem $\boldsymbol{IP}$

$$\boldsymbol{z} = \min\{\boldsymbol{cx} : \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{Z}^{\boldsymbol{n}}\}$$

a feasible solution of $\boldsymbol{X}$ identifies an upper bound for $\boldsymbol{z}$, while a relaxation of $\boldsymbol{IP}$ identifies a lower bound for $\boldsymbol{z}$. In this case, a relaxation exists if $\boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{c}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \boldsymbol{X}$ - see slide (6).

Finding primal bounds

"Good" bounds greatly help in finding the optimal solution. Hence the corresponding feasible solutions are determined according to some rationale, and not randomly.

Definition

A greedy algorithm is one consisting of a sequence of choices that appear to be best in the short run.

A greedy algorithm starts from scratch and at each iteration sets the variable that provides the best immediate reward.

Examples

We show three examples to identify a feasible solution

- Knapsack problem (maximisation)
- Travelling salesman problem (minimisation)
- The minimum spanning tree (minimisation)

The 0–1 Knapsack problem

Suppose there are n projects. The jth project, $j = 1, \ldots, n$ has a cost of a_j and a value of C_j. Each project is either done or not, that is, it is not possible to do a fraction of any of the projects. Also there is a budget of b available to fund the projects. The problem of choosing a subset of the projects to maximise the sum of the values while not exceeding the budget constraint is the 0–1 knapsack problem

$$\max \left\{ \sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_j x_j \leq b, x \in \{0, 1\}^n \right\}$$

Knapsack - lower bound

Consider the problem

$$\begin{aligned} \max & 12x_1 + 8x_2 + 17x_3 + 11x_4 + 6x_5 + 2x_6 + 2x_7 \\ & 4x_1 + 3x_2 + 7x_3 + 5x_4 + 3x_5 + 2x_6 + 3x_7 \leq 9 \\ & x_j \in \{0,1\} \text{ for } i = 1, \ldots, 7 \end{aligned}$$

Variables are ordered such that $c_j/a_j \geq c_{j+1}/a_{j+1}$ for $j = 1, \ldots, n-1$. A large ratio c_j/a_j means that item j has a "small" weight and a "large value". A greedy solution inserts in the knapsack the most valuable variables, starting from x_1 as long as there is enough space, and then moves to the next one, if possible.

Knapsack - lower bound

$$\begin{aligned}\max 12x_1 + 8x_2 + 17x_3 + 11x_4 + 6x_5 + 2x_6 + 2x_7 \\ 4x_1 + 3x_2 + 7x_3 + 5x_4 + 3x_5 + 2x_6 + 3x_7 \leq 9 \\ x_j \in \{0, 1\} \text{ for } i = 1, \ldots, 7\end{aligned}$$

In this case

1. Item 1 can enter as $a_1 = 4 \leq 9$, hence $x_1 = 1$
2. Also $x_2 = 1$ because $a_2 = 3 \leq 9 - a_1 = 5$
3. $x_3 = x_4 = x_5 = 0$ because $a_3, a_4, a_5 > 9 - a_1 - a_2 = 2$
4. $x_6 = 1$ because $a_6 = 2$ and there are just 2 units of space available in the knapsack
5. $x_7 = 0$ because there is no more room available in the knapsack

The greedy feasible solution is therefore $(1, 1, 0, 0, 0, 1, 0)$ and $\underline{z} = 22$.

No guarantee of a good solution

$$
\begin{aligned}
\max 2x_1 + Mx_2 & \\
x_1 + Mx_2 &\le M (> 2) \\
x_j &\in \{0, 1\} \text{ for } i = 1, \ldots, 2
\end{aligned}
$$

1. The optimal solution is $x_2 = 1$ and $z^* = M$
2. The greedy solution is $x_1 = 1$ and $z^g = 2$
3. Hence $\lim_{M \to +\infty} \frac{z^g}{z^*} = 0$

An improvement

Let

$$\bar{z}^g = \max\{z^g, \max_{1\leq j\leq n}\{p_j\}\}$$

$$\begin{aligned} \max 2x_1 + Mx_2 + Mx_3 & \\ x_1 + Mx_2 + Mx_3 &\leq 2M(> 2) \\ x_j \in \{0,1\} &\text{ for } i = 1,\ldots,3 \end{aligned}$$

1. The optimal solution is $x_2 = x_3 = 1$ and $z^* = 2M$
2. The greedy solution is $x_1 = x_2 = 1$, $z^g = 2 + M$, and $\bar{z}^g = \max\{M+2, M\} = M+2$
3. Hence $\lim_{M\to+\infty} \frac{\bar{z}^g}{z^*} = \frac{1}{2}$

The algorithm cannot produce solutions of less than half the optimum value.

TSP - upper bound

Consider the following Symmetric Travelling salesman problem

$$\begin{bmatrix} - & 9 & 2 & 8 & 12 & 11 \\ & - & 7 & 19 & 10 & 32 \\ & & - & 29 & 18 & 6 \\ & & & - & 24 & 3 \\ & & & & - & 19 \\ & & & & & - \end{bmatrix}$$

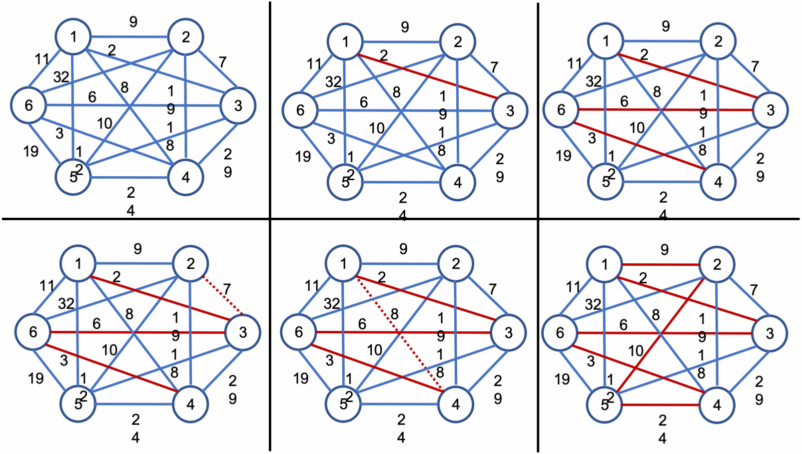
Multi-fragment algorithm. Insert the arcs in non-decreasing order of length, if possible. i.e., no cycles, no vertex with degree > 2.

- The cheapest arc is $(1,3)$ as $c_{13} = 2$, hence $x_{13} = 1$
- Next arcs are $(4,6)$ as $c_{46} = 3$ (hence $x_{46} = 1$) and $(3,6)$ as $c_{36} = 6$ (hence $x_{36} = 1$)
- Next cheapest arc is $(2,3)$ as $c_{23} = 7$, but node 3 has already two incident arcs (hence $x_{23} = 0$)
- Next cheapest arc is $(1,4)$ as $c_{14} = 8$, but this arc forms a tour with arcs $(1,3)$ and $(4,6)$ already chosen (hence $x_{14} = 0$)

By continuing in this way, we get $x_{12} = x_{25} = x_{45} = 1$. The cost of this solution (upper bound) is

$2 + 3 + 6 + 9 + 10 + 24 = 54$.

TSP - upper bound



TSP - upper bound

Consider the following Symmetric Travelling salesman problem

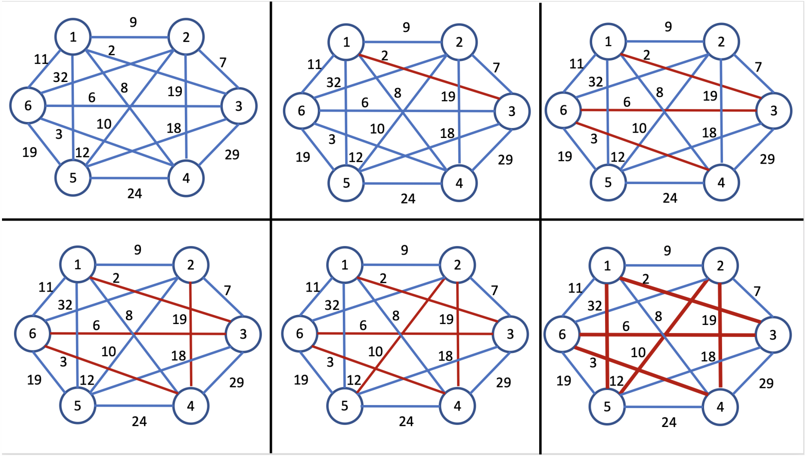
$$\begin{bmatrix} - & 9 & 2 & 8 & 12 & 11 \\ & - & 7 & 19 & 10 & 32 \\ & & - & 29 & 18 & 6 \\ & & & - & 24 & 3 \\ & & & & - & 19 \\ & & & & & - \end{bmatrix}$$

Nearest neighbour algorithm. Start from a vertex and at each iteration extend the path by choosing the cheapest arc among those connected to the last selected vertex that does not go to a vertex already visited.

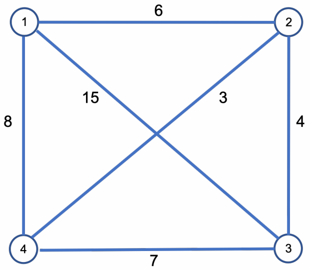
- Start from vertex $\mathbf{1}$. The cheapest arc is $(\mathbf{1}, \mathbf{3})$ as $c_{13} = \mathbf{2}$, hence $x_{13} = \mathbf{1}$
- The cheapest arc from vertex $\mathbf{3}$ is $(\mathbf{3}, \mathbf{6})$ as $c_{36} = \mathbf{6}$ (hence $x_{36} = \mathbf{1}$)
- The cheapest arc from vertex $\mathbf{6}$ is $(\mathbf{6}, \mathbf{4})$ as $c_{64} = \mathbf{3}$ (hence $x_{46} = \mathbf{1}$)
- The cheapest arc from vertex $\mathbf{4}$ is $(\mathbf{1}, \mathbf{4})$ as $c_{14} = \mathbf{8}$, but if we choose a vertex $\mathbf{1}$ a sub-cycle (1-3-6-4-1) is formed. Hence we have to consider to the next cheapest arc, which is $c(\mathbf{2}, \mathbf{4}) = \mathbf{18}$. Hence $x_{24} = \mathbf{1}$.
- The cheapest available arc from vertex $\mathbf{2}$ is $(\mathbf{2}, \mathbf{5})$ as $c_{25} = \mathbf{10}$ (hence $x_{25} = \mathbf{1}$). Neither arc $(\mathbf{1}, \mathbf{2})$ or $(\mathbf{2}, \mathbf{3})$ can be chosen because it leads to a sub-cycle.
- The last arc is mandatory, as all vertexes have been visited and we need to come back to vertex $\mathbf{1}$, hence $x_{15} = \mathbf{1}$

The cost of this solution (upper bound) is $\mathbf{2} + \mathbf{6} + \mathbf{3} + \mathbf{19} + \mathbf{10} + \mathbf{12} = \mathbf{52}$.

TSP - upper bound



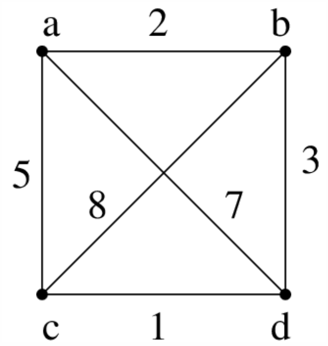
TSP - Nearest neighbour



- Starting from vertex **1** the cycle is **1 — 2 — 4 — 3 — 1**, which has cost: $\mathbf{6 + 3 + 7 + 15 = 31}$
- Starting from vertex **2** the cycle is **2 — 4 — 3 — 1 — 2**, which has cost: $\mathbf{3 + 7 + 15 + 6 = 31}$
- Starting from vertex **3** the cycle is **3 — 2 — 4 — 1 — 3**, which has cost: $\mathbf{4 + 3 + 8 + 15 = 30}$
- Starting from vertex **4** the cycle is **4 — 2 — 3 — 1 — 4**, which has cost: $\mathbf{3 + 4 + 15 + 8 = 30}$

However, the optimal solution is **1 — 2 — 3 — 4 — 1**, which has a cost $\mathbf{6 + 4 + 7 + 8 = 25}$. The optimal solution is never identified.

TSP - Nearest neighbour

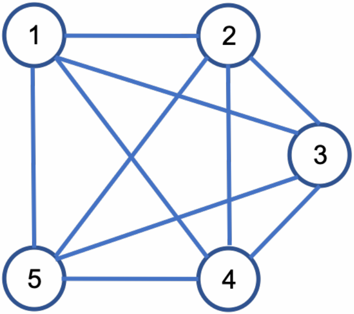


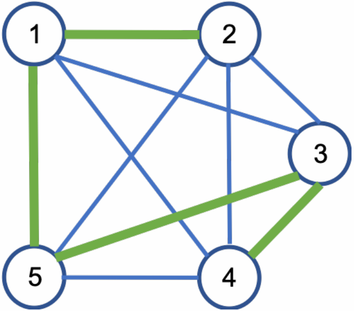
- The solution is always $a - b - d - c - a$, which has a cost $\mathbf{2} + \mathbf{3} + \mathbf{1} + \mathbf{5} = \mathbf{11}$, whichever is the starting node.
- The other two cycles $a - b - c - d - a$ and $a - c - b - d - a$ have costs **18** and **23**, respectively.

In this case, the optimal solution is always identified.

Spanning tree

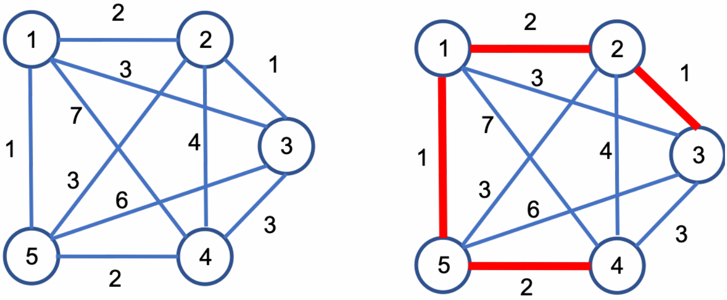
Given a connected undirect graph $\boldsymbol{G} = (\mathcal{N}, \mathcal{E})$, let $\mathcal{E}_1$ be a subset of $\mathcal{E}$ such that $\boldsymbol{T} = (\mathcal{N}, \mathcal{E}_1)$ is a tree. Such tree is called spanning tree.





Minimum spanning tree

We are given a connected undirect graph $G = (\mathcal{N}, \mathcal{E})$, with n nodes and m edges. For each edge $e \in \mathcal{E}$, we are also given a cost coefficient c_e. A minimum spanning tree (MST) is defined as a spanning tree such that the sum of the costs of its edges is as small as possible.



Minimum spanning tree

The minimum spanning tree problem arises naturally in many applications. For example, if edges correspond to communication links, a spanning tree is a set of links that allows every node to communicate (possibly, indirectly) to every other node. Then, a minimum spanning tree is a communication network that provides this type of connectivity, and whose cost is the smallest possible.

MST - Formulation

We define for each $e \in \mathcal{E}$ a variable x_e which is equal to 1 if edge e is included in the tree and 0 otherwise. Sine a spanning tree should have $n-1$ edges, we introduce the constraint

$$\sum_{e \in \mathcal{E}} x_e = n - 1.$$

Moreover, the chosen edges should not contain a cycle. The for any $S \subset \mathcal{N}$, we define

$$E(S) = \{\{i, j\} \in \mathcal{E} | i, j \in S\}$$

and we express this set of constraints as

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad S \subset \mathcal{N}, S \neq \emptyset, \mathcal{N}.$$

MST - Formulation

An integer programming formulation of the MST problem is

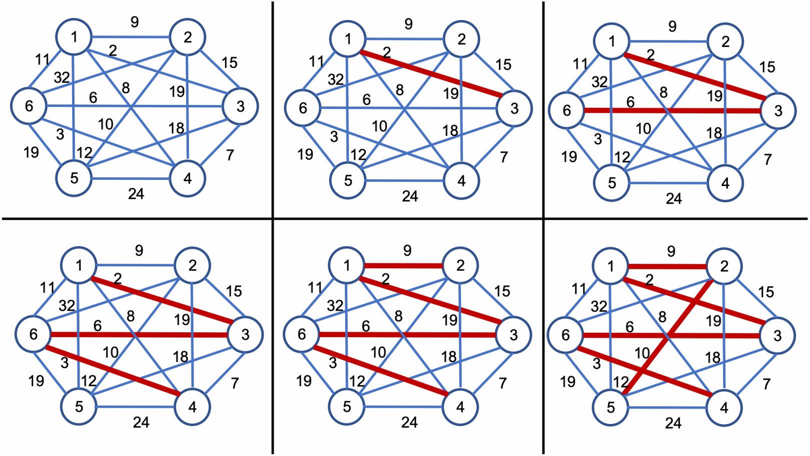
$$\begin{aligned}
\min \sum_{e \in \mathcal{E}} c_e x_e & \\
\sum_{e \in \mathcal{E}} x_e = n - 1 & \\
\sum_{e \in E(S)} x_e \leq |S| - 1 & \qquad S \subset \mathcal{N}, S \neq \emptyset, \mathcal{N} \\
x_e \in \{0, 1\} &
\end{aligned}$$

MST - A greedy algorithm

For certain problems, like the MST, short run optimal decisions turn out to be optimal in the long run as well. The algorithm we describe builds a MST by progressively adding edges to a current tree.

- At any stage we have a tree and we add a least expensive edge that connects a node in the tree with a node outside the tree.
- Since at each stage we connect a node in the current tree with a node outside the tree, no cycles are ever formed, and we always have a tree.

MST - A greedy algorithm



MST - Another greedy algorithm

We insert edges in increasing cost order ensuring that no cycles are formed.