Chapter 1

Preliminaries

In this introductive chapter, we want to recall several basic definitions and properties ([10], [64], [68], [77], [86]) that will be used in the subsequent chapters. A list of further notations can be found at the end of the dissertation.

First of all, a *Linear Programming* problem (LP) consists in minimizing or maximizing a linear function, called objective function, on a feasible region defined by a series of linear constraints. An example of LP problem in standard form looks like the following:

$$\begin{aligned} &\min c^T x \\ &s.t. \\ &\quad A\,x = b \\ &\quad x \geq 0 \end{aligned} \tag{1.1}$$

where A is a $m \times n$ real matrix with rank m, c is an n-dimensional vector, b an m-dimensional vector and x an n-dimensional vector of decision variables.

If the decision variables take only integer values, the problem:

$$\begin{aligned}
&\min c^T x \\
&s.t. \\
&\quad A\,x = b \\
&\quad x \geq 0 \\
&\quad x \in \mathbb{Z}^n
\end{aligned} \tag{1.2}$$

is an *Integer Linear Programming* (IP) problem. In particular, if all the decision variables are restricted to $0-1$ values, the problem is called *Binary Integer Programming* (BIP).

If some, but not all the decision variables are integer, the problem:

$$\begin{aligned}
&\min c^T x + d^T y \\
&s.t. \\
&\quad A\,x + By = b \\
&\quad x \geq 0,\ y \geq 0 \\
&\quad x \in \mathbb{Z}^n
\end{aligned} \tag{1.3}$$

is called *Mixed Integer Programming* (MIP) and B is a $m \times p$ matrix, d is a p-dimensional vector and y is a p-dimensional vector of real variables.

1.1 Formulations

Definition 1.1.1. The *feasible region* of an LP problem (1.1) is the set $P = \{x \in \mathbb{R}^n_+ :\ A\,x = b\}$ which is a polyhedron, while the feasible region of an IP problem (1.2) is the set $S := P \cap \mathbb{Z}^n$. If the polyhedron P is bounded, it is called polytope.

Definition 1.1.2 (Relaxation of an IP problem). Given the IP problem (1.2) with feasible region S, a problem of this type: $\min\{c^T x : \ x \in T \subseteq \mathbb{R}^n\}$ is a *relaxation* of it if $S \subseteq T$.

Naturally, the optimal value of a relaxation of an IP problem is lower than the optimal value of the IP problem and so it represents a lower bound for the optimal value of the IP problem.

There are several possible relaxations of an IP problem, but in the following we will consider only the linear relaxation.

Definition 1.1.3 (Linear relaxation). The *linear programming relaxation* of an IP problem: $\min\{c^T x : \ x \in P \cap \mathbb{Z}^n\}$ with formulation $P = \{x \in \mathbb{R}^n : A\,x \geq b\}$ is the LP problem: $\min\{c^T x : \ x \in P\}$.

The linear programming relaxation can be, thus, obtained by eliminating the restriction that the variables x need to be integer. For this reason, again, the optimal value of the linear relaxation of an IP problem is a lower bound of the optimal value of the IP problem itself.

Definition 1.1.4. Given two linear formulations P_1 and P_2 for an integer problem:

(i) the formulation P_1 is *better* than P_2 if and only if $P_1 \subset P_2$,

(ii) the formulation P_1 is *equivalent* to P_2 if and only if $P_1 = P_2$,

(iii) if neither formulation is better than the other they are *incomparable*.

Definition 1.1.5 (Convex hull). Given a set $S \subseteq \mathbb{R}^n$, the *convex hull* of S, denoted by $conv(S)$, is the set of all the possible finite convex combination of elements of S, i.e. $conv(S) := \{x \in \mathbb{R}^n : \ x = \sum_{i=1}^{k} \alpha_i x_i, \ \sum_{i=1}^{k} \alpha_i = 1, \ \alpha_i \geq 0 \ \forall i \in \{1, .., k\}, \ \text{ for all } \{x_1, .., x_k\} \text{ subsets of } S\}$.

Among all the possible linear relaxations of an integer programming problem, the best one is the convex hull of all its feasible points:

$$P_I := conv(P \cap \mathbb{Z}^n) = conv(\{x \in \mathbb{R}^n : \ Ax \geq b, \ x \text{ integer}\}). \tag{1.4}$$

Proposition 1.1.1. *It holds that $P_I \subseteq P$.*

In Figure 1.1, the yellow polytope is the convex hull of a feasible set S of integer points and it represents an ideal formulation for an IP problem with feasible set S, while the polytope which is the union of the yellow and green portions is a possible linear relaxation of the IP formulation.

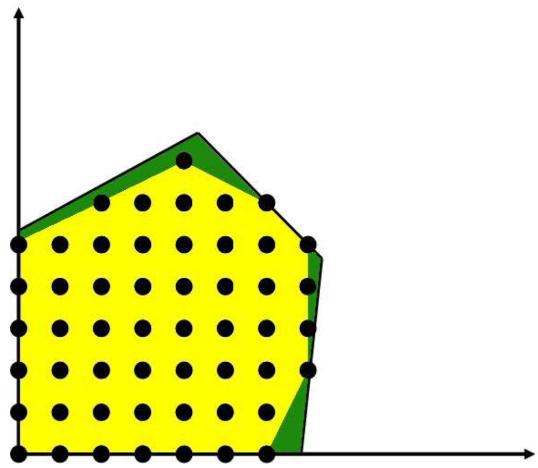

Figure 1.1: The ideal formulation and a possible LP relaxation of an IP problem

Definition 1.1.6 (Full–dimensional polyhedron). A polyhedron $P = \{x \in \mathbb{R}^n : \ A\,x \geq b\}$ is *full-dimensional* if and only if $dim(P) = n$, where $dim(P)$ is the maximum number of affinely independent points of P minus one.

In general, it is not trivial to give a complete description of the polyhedron P_I of an IP or MIP problem, so that it is interesting to strengthen certain inequalities, in particular, to find facet defining inequalities.

Definition 1.1.7 (Valid inequalities). Let $\pi \in R^n$, $\pi_0 \in R$ and let $P \subseteq \mathbb{R}^n$ be a polyhedron; the inequality $\pi^T x \leq \pi_0$ is a *valid inequality* for the polyhedron P if $\pi^T x \leq \pi_0$ for all the points $x \in P$, that is if $P \subseteq \{x \in \mathbb{R}^n : \ \pi^T x \leq \pi_0\}$.

Definition 1.1.8 (Facet defining inequalities). A valid inequality $\pi^T x \leq \pi_0$ is a *facet defining inequality* for a polyhedron P if and only if the equality $\pi^T x = \pi_0$ is verified for $dim(P)$ affinely independent points of P.

Another definition we should give is the definition of the support of a vector:

Definition 1.1.9 (Support). If x^* is an n-dimensional vector its *support* is the set:

$$Supp := \{j \in \{1, 2, .., n\} : \ x_j^* \neq 0\}.$$

1.2 Set Covering problem

The Set Covering problem is a classical Combinatorial Optimization problem of great theoretical and practical interest.

Definition 1.2.1 (Set Covering problem). Given a finite set I and a family $F = \{F_j\}_{j \in J}$ of subsets of I, given a cost $c_j \in \mathbb{R}^+$ associated with each element F_j of the family F. A subset $\overline{J}$ of the set J is a *cover* of I if

- $I = \bigcup_{i \in \overline{J}} F_i$

and it has the minimum cost if

$$\sum_{j \in \overline{J}} c_j \leq \sum_{j \in J'} c_j, \qquad \forall \ J' \subseteq J, J' \text{ cover of } I.$$

The Set Covering problem consists, thus, in finding a subset $\overline{J}$ of J such that

$$I = \bigcup_{j \in \overline{J}} F_j$$

and the cost $\sum_{j \in \overline{J}} c_j$ is the minimum of the costs of all the possible covers of I.

The Set Covering problem has been shown to be NP-complete in 1972 [45]. This type of problem can be formulated as an optimization problem introducing a $0-1$ matrix $A \in \mathbb{R}^{n \times m}$ called incidence matrix whose generic element a_{ij} is defined by:

$$a_{ij} = \begin{cases} 1 & \text{if } i \in F_j, \\ 0 & \text{otherwise.} \end{cases}$$

A formulation of the Set Covering problem can be, thus, the following:

$$\begin{aligned} & \min c^T x \\ & s.t. \\ & \quad Ax \geq \mathbf{1} \\ & \quad x \in \{0,1\}^n \end{aligned} \tag{1.5}$$

where c is a n-dimensional vector of costs.

There are several conditions for reducing the size of the incidence matrix of the Set Covering problem. Indeed, denoting by a_i^T the i^{th} row of A and by A_j the j^{th} column of A, the next proposition states some dominance rules for rows and columns of A.

Proposition 1.2.1 (Dominance of rows and columns).

i) If the i^{th} row is null, then the Set Covering problem is infeasible.

ii) *If the i^{th} row has only one element equal to one in the k^{th} column, then set $x_k = 1$ and erase not only the column A_k, but also all the rows j such that $a_{jk} = 1$.*

iii) *Let A_i and A_j be two columns such that $a_{ki} \geq a_{kj}$ for every row index k. If the corresponding costs are such that $c_i \leq c_j$, then erase the column j.*

iv) *Let a_i^T and a_j^T be two rows such that $a_{ik} \geq a_{jk}$ for every column index k, then covering the j^{th} row implies the covering of the i^{th} row so that, the i^{th} row can be erased.*

We denote by M the set of the row indices of the incidence matrix A and by N the set of the column indices of A. The Set Covering polytope $P_I(A)$ is:

$$P_I(A) := conv\big(\{x \in \mathbb{R}_+^{|N|} : \; A\,x \geq \mathbf{1}, \; x \leq \mathbf{1}, \; x \text{ integer}\}\big)$$

and the relaxed polytope $P(A)$ is:

$$P(A) := \{x \in \mathbb{R}_+^{|N|} : \; A\,x \geq \mathbf{1}, \; x \leq \mathbf{1}\}.$$

For each $i \in M$, we denote by N^i the set of the column indices j such that the value of the element a_{ij} of the matrix A is one, i.e.,

$$N^i := \{j \in N : \; a_{ij} = 1\}.$$

The Set Covering polytope has been widely studied (see e.g. [7], [8], [22], [76]) and here we summarize some of its properties.

Proposition 1.2.2.

- *$P_I(A)$ is full-dimensional if and only if $|N^i| \geq 2$ for all $i \in M$;*
- *if $P_I(A)$ is full-dimensional, then the inequality $x_i \geq 0$ defines a facet of $P_I(A)$ if and only if $|N^i \setminus \{j\}| \geq 2$ for all $i \in M$;*

- *if $P_I(A)$ is full-dimensional, then all the inequalities $x_j \leq 1$ for all $j \in N$ define facets of $P_I(A)$;*
- *if $P_I(A)$ is full-dimensional and $\pi_0 > 0$, then all facet defining inequalities $\pi\, x \geq \pi_0$ for $P_I(A)$ have $\pi_j \geq 0$ for all $j \in N$.*

Remark 1.2.1. The only facet defining inequalities for the Set Covering polytope having right hand side equal to one are among the inequalities of the system $A\, x \geq \mathbf{1}$.

1.3 Graphs

We report here several definitions about the graphs.

Definition 1.3.1 (Undirected and directed Graph). An *undirected graph* G is a pair $G = (V, E)$, where V is a finite set of nodes or vertices and E is a family of subsets of V of cardinality two, called edges. Furthermore, a *directed graph* D is a pair $D = (V, A)$ where V is the set of vertices and A is a set of ordered pairs of vertices, called arcs.

Definition 1.3.2 (Path). Given a graph $G = (V, E)$ a *path* is a sequence $[v_1, v_2, ..., v_k]$ of nodes with $k > 1$, such that each pair of consecutive nodes belongs to E and there is no repetition of nodes in the sequence.

Definition 1.3.3 (Cycle). Given a graph $G = (V, E)$ a *cycle* is a sequence $[v_1, v_2, ..., v_k]$ with $k \geq 1$, such that each pair of consecutive nodes belongs to E, the nodes $v_1, v_2, ..., v_{k-1}$ are distinct and $v_1 = v_k$.

Definition 1.3.4 (Tree). A *tree* $T = (V', E')$ is a connected graph with no cycles.

Definition 1.3.5 (Cutset). Let $G = (V, E)$ be an undirected graph, S be a subset of V and S^c its complementary in V, a *cutset* is the set: $\delta(S) :=$

$\{e = \{i, j\} \in E : i \in S, j \in S^c\}$. If the graph $G = (V, A)$ is a directed graph, then for $S \subset V$ two directed cuts can be defined:

$$\delta^+(S) := \{(i, j) \in A : i \in S, j \in S^c\}$$

is the set of the arcs outgoing from S and

$$\delta^-(S) := \{(i, j) \in A : i \in S^c, j \in S\}$$

is the set of the incoming arcs in S.

Definition 1.3.6 (degree). The *degree* of a node $v \in V$ is the cardinality of $\delta(\{v\})$. For simplicity it is common to use $\delta(v)$ instead of $\delta(\{v\})$. In a directed graph, the set of the incoming arcs in v is denoted by $\delta^-(v)$, whereas the set of the outgoing arcs from v is denoted by $\delta^+(v)$.

1.4 Shortest Path, Spanning Tree and Maximum Flow problems

Three well studied problems are defined in this section: the Shortest Path problem, the Minimum Spanning Tree problem and the Maximum Flow problem.

Definition 1.4.1 (The Shortest Path). Given a graph $G = (V, E)$ with nonnegative cost (or length) associated with each edge $e \in E$, the *Shortest Path* (SP) problem consists in finding a path from a source node s to a terminal node t with the minimum total cost (or length).

The Shortest Path problem is polynomially solvable and Dijkstra's algorithm is an efficient algorithm for solving it. This algorithm [27] starts with the node $s \in V$ and a set $L := \{s\}$; at each iteration the algorithm labels a node $i \in L^c$ with the shortest length of a path from s to i with internal nodes in L, updates the set $L := L \cup \{i\}$ and updates the distances from s to the nodes in L. This process is repeated until $t \in L$.

Definition 1.4.2 (The Minimum Spanning tree). Let $G = (V, E)$ be a graph with nonnegative cost (or weight) associated with each edge $e \in E$, the *Minimum Spanning Tree* problem consists in finding a tree with the minimum total cost (or weight) that spans all the nodes of G.

The greedy process that underlies Dijkstra's algorithm is similar to the process used in Prim's algorithm. Prim's algorithm [70] is used to find the Minimum Spanning Tree in a graph $G = (V, E)$. Starting with a node $s \in V$ and a set $L := \{s\}$, at each iteration the algorithm chooses a minimum-cost edge $e = \{u, v\} \in E$, connecting a node $u \in L$ to a node $v \in L^c$ and updates the set $L := L \cup \{v\}$. This process is repeated until $L = V$.

Definition 1.4.3 (Maximum Flow problem in capacitated graph). Given a directed graph $G = (V, A)$, two different nodes s and t belonging to V and a nonnegative capacity u_{ij} for each arc $(i, j) \in A$, the *Maximum Flow* problem consists in finding the maximum value of f such that a $|A|$-dimensional nonnegative vector x satisfies the flow conservation constraints

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} f & \text{if } i = s, \\ 0 & \forall\, i \in V \setminus \{s, t\}, \\ -f & \text{if } i = t, \end{cases}$$

not exceeding the capacities on the arcs $(0 \leq x_{ij} \leq u_{ij},\ \forall (i, j) \in A)$.

Definition 1.4.4 (Cut and capacity of a cut). Given a directed graph $G = (V, A)$ with a nonnegative capacity u_{ij} for each arc $(i, j) \in A$ and given two different nodes s and t, an $s - t$ *cut* is a partition (S, S^c) of the set V

such that $s \in S$ and $t \in S^c$. The *capacity* of this $s-t$ cut is

$$C(S, S^c) := \sum_{\substack{(i,j) \in A \\ i \in S,\ j \in S^c}} u_{ij}$$

Remark 1.4.1. The maximum flow value equals the total net flow across any $s-t$ cut (S, S^c):

$$f = \sum_{\substack{(i,j) \in A \\ i \in S,\ j \in S^c}} x_{ij} - \sum_{\substack{(j,i) \in A \\ j \in S^c,\ i \in S}} x_{ji}$$

Proposition 1.4.1 (Max-flow–Min-cut). *The value of a Maximum Flow problem equals the capacity of a Minimum cut [33].*

1.5 Steiner Tree problem

The Steiner Tree problem in a network is the problem of connecting a set of required vertices with the minimum cost.

Definition 1.5.1 (The Steiner Tree Problem (ST)). Given an undirected graph $G = (V, E)$ with a cost (or weight) c_e on each edge $e \in E$ and given a subset of the nodes R, called required nodes; the *Steiner Tree* problem consists in finding a minimum cost subtree of G that spans all the nodes in R with the possibility of including or not the nodes in $V \setminus R$, which are called Steiner nodes.

In general, the Steiner Tree problem is an NP-complete problem. Two special versions of the problem are polynomially solvable: if $|R| = 2$, then the problem reduces to the Shortest Path problem and if $R = V$, then the problem is the minimum Spanning Tree problem.

Definition 1.5.2 (Steiner Arborescence problem). The *Steiner Arborescence* problem is the directed version of the ST problem; the graph G is a directed weighted graph, a root node s, called source, is given and it is required to find a directed path from s to every terminal nodes in R with the minimum cost.

The cost or weight of a Steiner Tree T is indicated by $c(T)$ and it is defined as follows:

$$c(T) := \sum_{e \in T} c_e.$$

1.5.1 Preprocessing

Preprocessing the graph is an important factor for solving the ST problem in a reasonable time. It is applied on the undirected graph $G = (V, E)$ and the goal of this process is to reduce the size of the problem contracting or deleting nodes or edges in order to obtain an equivalent but reduced graph $G' = (V', E')$ ([6], [9], [16], [47], [81]).

Definition 1.5.3 (Feasible reduction). Given a Steiner Tree problem on the graph $G = (V, E)$ with terminal set R and costs c, a *feasible reduction* is a transformation of the problem into a Steiner Tree problem on the graph $G' = (V', E')$, with terminal set R', costs c' and constant cost $c_r \in \mathbb{R}_+$ with the properties that:

(i) $|V'| \leq |V|$,

(ii) $|E'| \leq |E|$,

(iii) $|R'| \leq |R|$,

(iv) if S is a feasible solution for the original problem, then there exists a feasible solution S' for the reduced problem with $c(S) = c'(S') + c_r$.

Quite simple reduction tests for the Minimum Steiner Tree are the degree tests applied recursively to each reduced graph until no more reduction can be performed.

Proposition 1.5.1 (Degree Reductions). *Given a Steiner Tree problem on the graph $G = (V, E)$, with terminal set R and vector of costs c:*

(i) *A Steiner node with degree less than or equal to one can be eliminated;*

(ii) *If a node i in R has degree one, its incident edge $\{i, j\}$ is contained in every feasible solution and can be contracted;*

(iii) *If a Steiner node i has degree two and $\{i, j\}$ and $\{i, k\}$ are its adiacent edges, then these edges can be replaced by the edge $\{j, k\}$ whose associated cost is $c_{(j,k)} = c_{(i,j)} + c_{(i,k)}$.*

Remark 1.5.1. Contracting an edge $\{i, j\}$ incident to a node $i \in R$ means:

- if $j \in R$, identify node i with j, eliminate the edge $\{i, j\}$, reduce the cardinality of R and store the cost $c_{(i,j)}$, that is, the costant cost c_r of the definition above is updated, i.e. $c_r := c_r + c_{(i,j)}$;
- if $j \in V \setminus (R \cup \{s\})$, identify nodes i with j (that becomes a required node) and update c_r.

1.5.2 Reduced costs fixing

Definition 1.5.4 (Reduced costs). Given an LP problem of the form (1.1), let B be an $m \times m$ nonsingular submatrix of A, x be a basic solution and c_B be the vector of costs of the basic variables. For each $j \in \{1, .., n\}$ the *reduced cost* $\bar{c}_j$ of the variable x_j is defined according to the formula:

$$\bar{c}_j = c_j - c_B^T B^{-1} A_j.$$

Let z_{LP} be the optimal value of the linear relaxation of an IP problem (see the problem (1.2)) and let z_{UB} be the value of the best feasible solution known for the problem (an upper bound for the optimal value of the problem).

Proposition 1.5.2 (Reduced costs fixing). *[64] If a nonbasic variable x_j at its lower bound in the optimal solution of the linear relaxation of an IP is such that $z_{LP} + \bar{c}_j \geq z_{UB}$, then there exists an optimal solution of the IP with x_j at its lower bound. Similarly, if a nonbasic variable x_k at its upper bound in the optimal solution of the linear relaxation of an IP is such that $z_{LP} - \bar{c}_k \geq z_{UB}$, then there exists an optimal solution of the IP with x_k at its upper bound.*