UNIVERSITÀ
DEGLI STUDI
DI PADOVA

INFORMATION ENGINEERING DEPARTMENT

COMPUTER ENGINEERING - AI & ROBOTICS

OPERATIONS RESEARCH 1 2022/2023

Francesco Crisci 2076739

Convex Programming

Definition 1.1.1

Given $x, i \in R^n$, the point $z := \lambda x + (1-\lambda)y$ is said to be a convex combination of x,y for all $\lambda \in [0,1]$. The combination is said to be strict if $0 < \lambda < 1$.
More generally, the convex combination of k points $x^1, ..., x^k \in R^n$ is defined as $\Sigma_{i=1}^k \lambda_i x^i$, with $\lambda_i, ..., \lambda_k \geq 0$ and $\Sigma_{i=1}^k \lambda_i = 1$.

Definition 1.1.2

A set $X \subseteq R^n$ is said to be convex if $\forall x, y \in X$ we have that X contains all the convex combinations of x and y, i.e : $z := [\lambda x + (1-\lambda)y], \forall \lambda \in [0,1]$.

Proposition 1.1.1

The intersection of two convex set $A, B \in R^n$ is still a convex set.
Proof:
Given $x, y \in A \cap B$, for all $\lambda \in [0,1]$ we have $z := \lambda x + (1-\lambda)y \in A$ by convexity of A, $z \in B$ for convexity of B, hence $z \in A \cap B$, as requested.

Definition 1.1.3

A function $f : X \to R$ defined on a convex set $X \subseteq R^n$ is said to be convex if $\forall x, y \in X$ and $\forall \lambda \in [0,1]$ we have that $f(z) \leq \lambda f(x) + (1-\lambda) f(y)$, where $z = \lambda x + (1-\lambda)y$.

Theorem 1.1.1

Let $X = \{x \in R^n : g_i(x) \leq 0, i = 1, ..., m\}$. If for all $i \in \{1, ..., m\}$ the functions $g_i : R^n \to R$ are convex, then the set X is convex.
Proof:
Clearly $X = \cap_{i=1}^m X_i$, where $X_i := \{x \in R^n : g_i(x) \leq 0\}$. By proposition 1.1.1, it is then sufficient to prove that each set X_i is convex. Indeed, given any two elements x and y of X_i and a generic point $x = \lambda x + (1-\lambda)y, \lambda \in [0,1]$, by the convexity hypothesis of the function g_i we can write $g_i(z) = g_i(\lambda x + (1-\lambda)y) \leq \lambda g_i(x) + (1-\lambda) g_i(y) \leq 0$, where the latter inequality is valid since $g_i(x) \leq 0, g_i(y) \leq 0$, and $0 \leq \lambda \leq 1$. It follows that $g_i(z) \leq 0$, hence $z \in X_i$. Given the arbitrariness of x,y and z, one thus has that X_i is convex, as requested.

Theorem 1.1.2

Consider a convex programming problem, i.., a problem $\min\{f(x) : x \in X\}$ where $X \in R^n$ is a convex set and $f : X \to R$ is a convex function. Every locally optimal solution is also a globally optimal solution.
Proof:
Let $\tilde{x}$ be any locally optimal solution. By the lcal optimum definition, there exists then $\epsilon > 0$ such that $f(\tilde{x}) \leq f(z)$

for all $z \in I_\epsilon(\tilde{x}) := \{x \in X : |x - \tilde{x}|| \leq \epsilon\}$. We have to prove that $f(\tilde{x}) \leq f(y)$ for all $y \in X$. Given any $y \in X$, consider the point z belonging to the segment that connects $\tilde{x}$ to y and define as $z := \lambda\tilde{x} + (1-\lambda)y$, where $\lambda < 1$ is chose very close to the value 1 so that $z \in I_\epsilon(\tilde{x})$ and hence $f(\tilde{x}) \leq f(z)$. By the convexity hypothesis of f it follows that $f(\tilde{x}) \leq f(z) = f(\lambda\tilde{x} + (1-\lambda)y) \leq \lambda f(\tilde{x}) + (1-\lambda)f(y)$, from which, dividing by $(1-\lambda) > 0$, we obtain $f(\tilde{x}) \leq f(y)$, as requested.

Simplex Algorithm

Definition 4.1.1

The sets $\{x \in R^n : \alpha^T x \leq \alpha_0\}$ and $\{x \in R^n : \alpha^T = \alpha_0\}$ are called affine half.space and hyperplane, respectively, induced by (α, α_0).

Definition 4.1.2

A (convex) polyhedron is defined as the intersection of a finite number of affin ehalf-spaces and hyperplanes. The sets of feasible solutions of Linear Programming problems are hence polyhedra.

Definition 4.1.3

A bounded (i.e. there exists $M > 0$ such that $||x|| \leq M$ for all $x \in P$) polyhedron P is called polytope.

Definition 4.1.4

A point x of a polyhedron P is said to be an extreme point or a vertex of P if it cannot be expressed as a strict convex combination of other two points of the polyhedron, i.e, if there exist no $y, z \in P, y \neq z$ and $\lambda \in (0, 1)$ such that $x = \lambda y + (1 - \lambda)z$.

Theorem 4.1.1 Minkowski-Weyl theorem

Every point of a polytope can be obtained as the convex combination of its vertices.

Theorem 4.1.2

If the set P of the feasible solutions of the linear programming problem $min\{c^T x : x \in P\}$ is bounded, then there exists at least one optimal vertex of P.
Proof:
Let $x^1, ..., x^k$ be the vertices of P and $z^* := min\{c^T x x^i : i = 1, ..., k\}$. Given any $y \in P$, we need to prove that $c^T y \geq z^*$. Indeed, $y \in P$ implies the existence of multipliers $\lambda_1, ..., \lambda_k \geq 0, \Sigma_{i=1}^k \lambda_i = 1$, such that $y = \Sigma_{i=1}^k \lambda_i x^i$. Hence we have $c^T y = c^T \Sigma_{i=1}^k \lambda_i x^i = \Sigma_{i=1}^k \lambda_i (c^T x^i) \geq \Sigma_{i=1}^k \lambda_i z^* = z^*$.

Definition 4.1.5

A collection of m linearly independent columns of A is said to be a basis of A. The x_j variables associated with the basic columns are called basic variables; the remaining variables are called non-basic variables.

Definition 4.1.6

The solution obtained imposing $x_F = 0$ and $x_b = B^{-1}b$ is said to be the basic solution associated with basis B. The basic solution (and by extension, basis B itself) is said to be feasible if $x_B = B^{-1}b \geq 0$.

Definition 4.1.7

A basis B is said to be degenerate if $B^{-1}b$ has one or more zero components.

Theorem 4.1.3

A point $x \in P$ is a vertex of the not empty polyhedron $P := \{x \geq 0 : Ax = b\}$ if and only if x is a basic feasible solution of the system $Ax = b$.
Proof:
Let us prove the implication x is a basic feasible solution then x is a vertex. Let $x = [x_1, ..., x_k, 0, ..., 0]^T$ be any basic feasible solution associated with basis B of A, where $k \geq 0$ is the number of non-zero components of x. It follows that columns $A_1, ..., A_k$ must be part of B, possibly together with other columns. Let us assume by contradiction that x is not a vertex. There exist thus $y = [y_1, ..., y_k, 0, ..., 0]^T \in P$ and $z = [z_1, ..., z_k, 0, ..., 0]^T \in P$ with $y \neq z$ such that $x = \lambda y + (1 - \lambda)z$ for any $\lambda \in (0, 1)$, which implies that $k \geq 1$. Note that both y and z must have the last components set to zero, otherwise their convex combination cannot give x. For the hypothesis, we then have $y \in P \Rightarrow Ay = b \Rightarrow A_1y_1 + ... + A_ky_k = b$ and $z \in P \Rightarrow Az = b \Rightarrow A_1z_1 + ... + A_kz_k = b$. By subtracting the second equation from the first we obtain $(y_1 - z_1)A_1 + ... + (y_k - z_k)A_k = \alpha_1A_1 + ... + \alpha_kA_k = 0$, where $\alpha_i = y_i - z_i, i = 1, ..., k$. Hence there exist $\alpha_1, ..., \alpha_k$ scalars not all zero such that $\Sigma_{i=1}^{k}\alpha_iA_i = 0$, thus columns $A_1, ..., A_k$ are linearly dependent and cannot be part of the basis B ($\Rightarrow$ contradiction).
We will now prove the implication x is a vertex then x is a basic feasible solution; the fact that the basic solution is also feasible obviously derives from the hypothesis that $x \in P$. Writing, as before, $x = [x_1, ..., x_k, 0, ..., 0]^T$ with $x_1, ..., x_k > 0$ and $k \geq 0$, we have that $x \in P \Rightarrow Ax = b \Rightarrow A_1x_1 + ... + A_kx_k = b$. From here two cases can occur:

1. Columns $A_1, ..., A_k$ are linearly dependent (or k = 0): by arbitrarily selecting other m-k linearly independent columns (which, as is well known, is always possible), we obtain basis $B = [A_1, ..., A_k, ...]$ whose basic associated solution is indeed x (which satisfies Ax=b and has non-basic components all equal to zero), thus concluding the proof.

2. columns $A_1, ..., A_k$ are linearly dependent: we will prove that this case cannot actually happen. Indeed, if the columns were linearly dependent, then there would exist $\alpha_1, ..., \alpha_k$ not all zero such that $\alpha_1A_1 + ... + \alpha_kA_k = 0$. With some math manipulation we can obtain, with $\epsilon > 0$, the following $(x_1 + \epsilon\alpha_1)A_1 + .. + (x_k + \epsilon\alpha_k)A_k = b$ and $(x_1 - \epsilon\alpha_1)A_1 + .. + (x_k - \epsilon\alpha_k)A_k = b$. By defining $y := [x_1 - \epsilon\alpha_1, ..., x_k - \epsilon\alpha_k, 0, ..., 0]^T$ and $z := [x_1 + \epsilon\alpha_1, ..., x_k + \epsilon\alpha_k, 0, ..., 0]^T$, we would have Ay=b and Az=b, while choosing a sufficient small ϵ we would have $y, z \geq 0$ and thus $u, z \in P, y \neq z$. But since by construction we have $x = \frac{1}{2}y + \frac{1}{2}z$, thus would mean that vertex x can be expressed as the strict convex combination of two distinct points of P($\Rightarrow$ contradiction).

Corollary 4.1.1

Every problem $min\{c^Tx : Ax = b, x \geq 0\}$ defined on a polytope $P = \{x \geq 0 : Ax = b\} \neq \emptyset$ has at least one optimal solution coinciding with a basic feasible solution.
Proof:
According to theorem 4.1.2, there always exists an optimal solution coinciding with a vertex of P and thus, according to theorem 4.1.3, with a basic feasible solution.

Definition 4.2.1

The vector $\bar{c}^T := c^T - c_B^TB^{-1}A = [c_B^T - c_B^TB^{-1}B = 0, c_F^T - c_B^TB^{-1}F]$ is called the reduced cost vector with respect to basis B.

Theorem 4.2.1

Let B be a feasible basis. If $\bar{c}^T := c^T - c_B^T B^{-1} A \geq 0^T$, then the basic solution associated with B is optimal.
Proof:
Rewriting the objective function as $c^T x = c_B^T B^{-1} b + \bar{c}^T x$, in the hypothesis $\bar{c}^T \geq 0^T$ we have that $c^T x \geq c_B^T B^{-1} b$ for all $x \geq 0$ (and thus for all $x \in P$), where $c_B^T B^{-1} b = [c_B^T, c_F^T] \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix}$ is the value of the objective function corresponding to the basic feasible solution associated with B.

Definition 4.2.2

Given basis B and $z = c^T x$, system $\begin{cases} x_B = B^{-1}b - B^{-1}Fx_F \\ z = c_B^T B^{-1} b + (c_F^T - c_B^T B^{-1} F) x_F \end{cases}$ is said to be in canonical form with respect to B.

Bland's rule

Whenever it is possible to choose, always choose the entering/leaving variable x_j with the smallest j. In particular, we have to:

- choose variable x_h to enter the basis defining $h := argmin\{j : \bar{c}_j < 0\}$;
- among all rows t with $\bar{b}_t / \bar{a}_{th} = \theta$ that are eligible for the pivot operation, choose the one with minimum $\beta[t]$ so as to force the smallest-index variable $x_{\beta[t]}$ to leave the basis.

Theorem 4.2.2

Using Bland's rule, the simplex algorithm converges after, at most, $\binom{n}{m}$ iterations.
Proof:
Let us suppose by contradiction that the thesis is false and let us consider as counterexample the smallest LP problem for which there is no convergence. As already seen, in this case the simplex algorithm has to go through a cyclic sequence $B_1, B_2, ..., B_k = B_1$ of bases. During this sequence, pivot operations are performed on all rows and all columns of the tableau, otherwise eliminating the not involved rows/columns we would obtain a smaller counterexample. It follows that all variables enter and leave, in turn, the current basis. Moreover, we must have $\bar{b}_t = 0$ for all rows $t \in \{1, ..., m\}$, otherwise in the iteration in which the pivot operation is performed on row t, we would have $\theta > 0$, hence the value of the objective function would change- preventing cycling in the sequence of bases.
Consider now tableau T in which variable x_n leaves the current basis to let a given non-basic variable x_h enter the basis. Let us indicate with $x_{\beta[i]}$ the basic variable in row $i \in \{1, ..., m\}$, and with t the row in which x_n is in the basis. The main features of this tableau are:

- $\bar{b}_i = 0$
- $\bar{c}_h < 0$ since x_h enters the basis
- $\bar{c}_{\beta[i]} = 0 \forall i$, the reduced cost of the basic variables
- $\bar{a}_{th} > 0$, the pivot element
- $\bar{a}_{ih} \leq 0 \forall i \neq t$, due to Bland's rule

Note that if there existed $i \neq t$ with $\bar{a}_{ih} > 0$, then Bland's rule would have certainly have preferred to let variable $x_{\beta[i]}$ leave the basis. Let us now consider tableau $\tilde{T}$ in which x_n re-enters the basis; indicating with $\tilde{c}_j$ the reduced costs in row 0 of $\tilde{T}$, we must have:

- $\tilde{c}_n < 0$ since x_n enters the basis

- $\tilde{c}_j \geq 0 \forall j \neq n$ due to Bland's rule.

Now, $\tilde{T}$ has been obtained from T by means of a sequence of pivot operations, hence there exist appropriate multipliers $\mu_1, ..., \mu_m$ such that $[row\ 0\ of\ \tilde{T}] = [row\ 0\ of\ T] + \Sigma_{i=1}^{m} \mu_i [row\ i\ of\ T]$, but then:

- $\tilde{c}_{\beta[t]} = \tilde{c}_n < 0$, $\bar{c}_{\beta[t]} = 0 \Rightarrow \tilde{c}_{\beta[t]} = \bar{c}_{\beta[t]} + \mu_t \Rightarrow \mu_t < 0$
- $\tilde{c}_{\beta[i]} \geq 0, \bar{c}_{\beta[i]} = 0 \Rightarrow \tilde{c}_{\beta[i]} = \bar{c}_{\beta[i]} + \mu_i \Rightarrow \mu_i \geq 0 \forall i \neq t$

hence there is a contradiction: $\underbrace{\tilde{c}_h}_{\geq 0} = \underbrace{\bar{c}_h}_{< 0} + \Sigma_{i \neq t} \underbrace{\bar{a}_{ih}}_{\leq 0} \underbrace{\mu_i}_{\geq 0} + \underbrace{\bar{a}_{th}}_{> 0} \underbrace{\mu_t}_{< 0} < 0.$

Duality in LP

Definition 5.1.1

Given a set $X \subseteq R^n$, an inequality $c^T x \geq c_0$ satisfied by all $x \in X$ is said to be valid for X.

Theorem 5.1.1 Farkas' Lemma

The inequality $c^T x \geq c_0$ is valid for the non-empty polyhedron $P := \{x \geq 0 : Ax = b\}$ if and only if $\mu \in R^m$ exists such that $c^T \geq u^T A$ and $c_0 \leq u^t b$.
Proof:
As already seen, the fact that the condition is sufficient is trivially true, given that for all $x \geq 0$ such that Ax=b we have: $c^T x \geq u^T Ax = u^T b \geq c_0$. We will now prove that the condition is also necessary, i.e, that $c^T x \geq c_0$ valid for $P \neq \emptyset \Rightarrow \exists u \in R^m : c^T \geq u^T A, c_0 \leq u^T b$. By the hypothesis validity, we have that $c_0 \leq z^* := min\{c^T x : Ax = b, x \geq 0\}$, which includes $z^* = -\infty$. Let then x^* be an optimal feasible solution found by the simplex algorithm applied to the aforementioned problem. This solution exists by the convergence property of the simplex algorithm. In addition, let B be an optimal basis associated with x^*, and let as usual $A = [B, F], c^T = [c_B^T, c_F^T]$ and $x^* = (x_B^*, x_F^*)$ with $x_B^* = B^{-1}b$ and $x_F^* = 0$. We will prove that vector $u^T := c_B^T B^{-1}$ verifies the two conditions, hence that the thesis is valid. Recalling the reduced cost expression computed in correspondence of the optimal basis B we have: $\bar{c}^T := c^T - \underbrace{c_B^T B^{-1}}_{u^T} A \geq 0^T \Rightarrow c^T \geq u^T A$, and thus it satisfies the first condition. In addition, for the optimization problem we have that: $c_o \leq z^* = c^T x^* = c_B^T x_B^* + c_F^T x_F^* = c_B^T B^{-1} b = u^T b$, and thus also the second condition is verified.

Dual Problem

The considerations above show that, if the problem $min\{c^T x : Ax = b, x \geq 0\}$ has an optimal finite value, then:
$min_x\{c^T x : Ax = b, x \geq 0\} = max_{c_0}\{c_0 : c^T x \geq c_0 \; valid \; for \; P\}$
$= max_{c_0,u}\{c_0 : c_0 \leq u^T b, c^T \geq u^T A\}$ by Farkas' lemma
$= max_u\{u^T b : c^T \geq u^T A\}$ since, at the optimal solution $c_0 = u^T b$.

Proposition 5.3.1

The dual of the dual problem coincides with the primal problem.

Theorem 5.3.1 Strong Duality

Let $P := \{x \geq 0 : Ax \geq b\} \neq \emptyset$ with $min\{c^T x : x \in P\}$ finite. Then $min\{c^T x : Ax \geq b, x \geq 0\} = max\{u^T b : c^T \geq u^T A, u \geq 0\}$
Proof:
The proof is the result of the considerations mentioned above.

Theorem 5.3.2 Weak Duality

Let $P := \{x \geq 0 : Ax \geq b\} \neq \emptyset$ and $D := \{u \geq 0 : c^T \geq u^T A\} \neq \emptyset$. For all pairs of points $\bar{x} \in P$ and $\bar{u} \in D$ we have that $\bar{u}^T b \leq c^T \bar{x}$.
Proof:
Given $\bar{x} \in P$ and $\bar{u} \in D$, we have $A\bar{x} \geq b, \bar{x} \geq 0, \bar{u} \geq 0$ and $c^T \geq \bar{u}^T A$, from which $\bar{u}^T b \leq \bar{u}^T A\bar{x} \leq c^T \bar{x}$.

Corollary 5.3.1

Consider the pair of primal $min\{c^T x : Ax \geq b, x \geq 0\}$ and dual $max\{u^T b : c^T \geq u^T A, u \geq 0\}$ problems. There are only 4 cases:

1. Both problems have a finite optimum, with $min\{c^T x : Ax \geq b, x \geq 0\} = max\{u^T b : c^T \geq u^T A, u \geq 0\}$;
2. The primal problem is unbounded and the dual is unfeasible;
3. The dual problem is unbounded and the primal is unfeasible;
4. Both problem are unfeasible

Proof:
Case 1 derives from the strong duality theorem, while 2 and 3 from the weak duality, given that the existence of a feasible solution in one of the two problems prevents the other from being unbounded. Case 4 is not excluded from any property and indeed occurs.

Theorem 5.5.1 Optimality Conditions

Two vectors $\bar{x} \in R^n$ and $\bar{u} \in R^m$ are optimal for problems $min\{c^T x : Ax \geq b, x \geq 0\}$ and $max\{u^T b : c^T \geq u^T A, u \geq 0\}$, respectively, if and only if the following optimality conditions hold:

- $A\bar{x} \geq b, \bar{x} \geq 0$, primal feasibility
- $c^T \geq \bar{u}^T A, \bar{u} \geq 0$, dual feasibility
- $\bar{u}^T(A\bar{x} - b) = 0$
- $(c^T - \bar{u}^T A)\bar{x} = 0$

The last two conditions are the complementary slackness conditions.

Bland's Rule(Dual)

Whenever it is possible to choose, always choose the entering/leaving variable x_j with the smallest index j

- Choose variable $x_{\beta[t]}$ leaving the basis with minimum index $\beta[t]$, i.e., the pivot row t with $\bar{b}_t < 0$ and minimum $\beta[t]$.
- among all variables x_h with $\bar{a}_{th} < 0$ and $\bar{c}_h/|\bar{a}_{th}| = min\{\bar{c}_j/|\bar{a}_{tj}| : \bar{a}_{tj} < 0\}$ that are eligible for entering the basis, choose the one with minimum index h.

Integer Linear Programming

Definition 6.1.1

Given a set $S \subseteq R^n$, a convex hull of S is the smallest convex set conv(S) that contains S.

Definition 6.2.1

An integer matrix A of size mxn with $m \leq n$ is said to be unimodular if for any of its mxm submatrices B, $det(B) \in \{-1, 0, 1\}$.

Theorem 6.2.1

Let A be unimodular and b integer. Then the polyhedron $P := \{x \geq 0 : Ax = b\}$ has only integer vertices.
Proof:
Let x^* be any vertex of P. As already proven, there exists a basis B of A such that $x^* = (B^{-1}b, 0)$. Since B is a nonsingular mxm submatrix of A, we have $|det(B)| = 1$, which implies the integrality of B^{-1} and of $B^{-1}b$ for all integers b.

Definition 6.2.2

An integer matrix A of size mxn is said to be totally unimodular (TUM) if $det(Q) \in \{-1, 0, 1\}$ for any of its square submatrices Q of any order.

Theorem 6.2.2

Let A be totally unimodular and b integer. Then polyhedron $P := \{x \geq 0 : AX \geq b\}$ has only integer vertices.
Proof:
Let x^* be any vertex of polyhedron P. First of all, we prove that $(x^*, s^* := Ax^* - b)$ is a vertex of polyhedron $P' := \{(x, s) \geq 0 : Ax - s = b\}$. If that was not the case, there would exist two distinct points (x^1, s^1) and (x^2, s^2) of P' such that $(x^*, s^*) = \lambda(x^1, s^1) + (1 - \lambda)(x^2, s^2)$ for some λ with $0 < \lambda$ 1. Note that x^1 and x^2 belong to P, given that $s^1 = Ax^1 - b \geq 0$ and $s^{"} = Ax^2.b \geq 0$. In addition $(x^1, s^1) \neq (x^2, s^2)$, hence $x^* = \lambda x^1 + (1 - \lambda)x^2$ cannot be a vertex of P. Since A is TUM, we have that $A' := [A, -I]$ is unimodular. By theorem 6.2.1, (x^*, s^*) is integer, hence x^* is integer as well.

Theorem 6.2.3

Let A be a matrix with $a_{ij} \in \{-1, 0, 1\} \forall i, j$. A is totally unimodular if the following conditions hold:

1 Every columns of A has no more that two non-zero elements

2 there exists a partition (I_1, I_2) of the rows of A such that each column with two non-zero elements has these two elements belonging to rows on different sets if and only if the two elements have the same sign.

Proof:
We have to prove that $det(Q) \in \{-1, 0, 1\}$ for any submatrix Q of A of order k (k=1,...,m). The proof is by induction on k. If k = 1, then $Q = [a_{ij}]$ and hence $det(Q) = a_{ij} \in \{-1, 0, 1\}$, as requested. Let us suppose now that $det(Q') \in \{-1, 0, 1\}$ for any submatrix Q' of order k', where $k' \geq 1$ is a fixed value. Let us consider any submatrix Q of order $k := k' + 1$. By condition (1) only three cases can occur:

- Q has one column of zeros: in this case det(Q)=0.
- Q has a column with only one element different from zero: in this case, barrring permutations of rows and/or columns, Q is of the type $Q = \begin{bmatrix} \pm 1 & | & *...* \\ - & - & - \\ 0 & | & \\ . & | & \\ . & | & Q' \\ . & | & \\ 0 & | & \end{bmatrix} \Rightarrow det(Q) = \pm 1 det(Q')$
  where $det(Q') \in \{-1, 0, 1\}$ given that Q' has order k-1=k'.
- Each column of Q has exactly two non-zero elements: in this case, let I(Q) be the set of rows of Q. We then have $\Sigma_{i \in I_1 \cap I(Q)}[row\ i\ of\ Q] - \Sigma_{i \in I_2 \cap I(Q)}[row\ i\ of\ Q] = [null\ row]$ given that in every column the two non-zero elements cancel out. It follows that the rows of Q are linearly dependent, hence det(Q)=0.

In each of the three cases ab ove we therefore have $det(Q) \in \{-1, 0, 1\}$, hence this property applies to all submatrices Q of order k=k'+1. Applying the same reasoning, the result can be inductively extended to matrices Q of any order.

Proposition 6.2.1

Matrix A is TUM is and only if:

- A^T is TUM;
- matrix A' obtained from A permuting and/or changing sign to some columns and/or rows is TUM;
- matrices $\begin{bmatrix} \pm 1 & | & \\ 0 & | & \\ . & | & \\ . & | & A \\ . & | & \\ 0 & | & \end{bmatrix}$ and $\begin{bmatrix} 0 & | & \\ 0 & | & \\ . & | & \\ . & | & A \\ . & | & \\ 0 & | & \end{bmatrix}$ are TUM.

Definition 6.3.1

Given $x^* \in P$ a cut is an inequality of the type $\alpha^T x \leq \alpha_0$ such that:

1. $\alpha^T x \leq \alpha_0, \forall x \in X := P \cap Z^n$
2. $\alpha^T x^* > \alpha_0$

These two conditions imply that the inequality $\alpha^T x \leq \alpha_0$ is valid for X, but violated by point x^*.

Chvàtal's inequalities

For all $r \in R$, let $\lfloor r \rfloor$ be the truncation of r to the nearest lower integer,i.i. $\lfloor r \rfloor := max\{i \in Z : i \leq r\}$. Given system Ax=b, $x \geq 0$, choose randomly $u \in R^m$ and obtain by linear combination the valid equation for P: $u^T Ax = u^T b$. By means of truncation, we can obtain the "weakened" inequality $\alpha^T x := \lfloor u^T A \rfloor x \leq u^T b$, where coefficients α_j are defined as $\alpha_j := \lfloor u^T A_j \rfloor := \lfloor \Sigma_{i=1}^m u_i a_{ij} \rfloor$for all j=1,...,n. At least, defining $\alpha_0 := \lfloor u^T b \rfloor$ we obtain the valid inequality for X but not necessarily for P $\alpha^T x \leq \alpha_0$.

It is possible to prove that, given a fractional vertex x^* of P, there always exists a vector $u \in R^m$ for which the corresponding inequality $\alpha^T x \leq \alpha_0$ is violated by x^*. With the Chvàtal procedure it is therefore always possible to cut any fractional vertex of P, by choosing vector u appropriately.
However, it would be wrong to think that all valid inequalities for X are obtained directly starting from the initial system Ax=b: there exist valid inequalities for X that are not obtained as $\lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor$ for any $u \in R^m$. Indicating with $A^{(1)}x \leq b^{(1)}$ the families of inequalities $\lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor, u \in R^m$, we can define the first Chvàtal's closure as $P_1 := \{x \geq 0 : Ax = b, A^{(1)}x \leq b^{(1)}\} \subseteq P$, where $P_1 = P$ if and only if P does not have fractional vertices. Although system $A^{(1)}x \leq b^{(1)}$, in principle, contains ∞^m inequalities, it is possible to prove that only a finite number of its inequalities is necessary to describe P_1, hence P_1 is a polytope. The additional constraint $A^{(1)}x \leq b^{(1)}$ allows one to improve the initial formulation Ax=b, $x \geq 0$. If conv(X)=P_1, this formulation is ideal. Otherwise there exists fractional vertices of P_1, that can be cut generating new Chvàtal's inequalities. We keep going till we have a solution completely integer, i.e, we keep iterating till $P_1 \supset P_2 \supset ... \supset P_K = conv(X)$.

Gomory's cut

The idea is to generate a cut using the information associated with the basis B corresponding to the optimal basic solution x^* of the current relaxation. If x^* is integer, it is not necessary to generate any cut. Otherwise, there exists at least one fractional component x_h^*. Variable x_h must then be a basic variable on a certain row t of the optimal tableau associated with x^*. The equation associated with row t of the optimal tableau is the following: $x_h + \Sigma_{j \in F} \bar{a}_{tj} x_j = \bar{b}_t (= x_h^*)$, where F is the set of indexes of non-basic variables. Row t is said to be the generating row for the cut. The previous equation is, by construction, valid for P. By applying Chvàtal's truncation procedure starting from this equation, we immediately obtain Gomory's cut: $x_h + \Sigma_{j \in F} \lfloor \bar{a}_{tj} \rfloor x_j \leq \lfloor \bar{b}_t \rfloor$. By construction, this inequality is valid for X, but violated by x^*. Indeed, given that $x_j^* = 0$ for all $j \in F$ and that $\bar{b}_t = x_h^*$ is fractional, we have: $(x_h + \Sigma_{j \in F} \lfloor \bar{a}_{tj} \rfloor x_j)_{x=x^*} = x_h^* = \bar{b}_t > \lfloor \bar{b}_t \rfloor$. The previous equation is said to be the integer form of the cut. There exists an equivalent formulation, called fractional form, which is obtined by subtracting the two equations: $\Sigma_{j \in F} (\bar{a}_{tj} - \lfloor \bar{a}_{tj} \rfloor) x_j \geq \bar{b}_t \lfloor \bar{b}_t \rfloor$, i.e., $\Sigma_{j \in F} \phi(\bar{a}_{tj}) x_j \geq \phi(\bar{b}_t)$, where for all $r \in R$ we indicate with $\phi(r) := r - \lfloor r \rfloor \geq 0$. Converting the first equation to standard form, we obtain $x_h + \Sigma_{j \in F} \lfloor \bar{a}_{tj} \rfloor x_j + s = \lfloor \bar{b}_t \rfloor$, where the slack variable $s \geq 0$ will be integer for all integer x, given that all coefficients are integer. We van then obtain the final equation as:$\Sigma_{j \in F} - \phi(\bar{a}_{tj}) x_j + s = -\phi(\bar{b}_t), s \geq 0$ integer.

Graph Theory

Definition 7.3.1

The node-edge incidence matrix D of an undirected graph G=(V,E) is the $|V|x|E|$ matrix with elements d_{ij} $\begin{cases} 1\ if\ the\ j-th\ edge \\ 0\ otherwise. \end{cases}$

Definition 7.3.2

The node-edge incidence matrix D of a directed graph G=(V,A) is the $|V|x|A|$ matrix with elements d_{ij} $\begin{cases} 1\ if\ the\ j-th\ leaves\ ve \\ -1\ if\ the\ j-th\ enters \\ 0\ otherwise. \end{cases}$

Definition 7.3.3

The adjacency matrix Q of a simple graph $G = (V, E)$ is the symmetric $|V|x|V|$ matrix with elements q_{ij} $\begin{cases} 1\ if\ [i,j] \in E \\ 0\ otherwise. \end{cases}$

Definition 7.3.4

The adjacency matrix Q of a simple directed graph $G = (V, A)$ is the symmetric $|V|x|V|$ matrix with elements q_{ij} $\begin{cases} 1\ if\ (i,j) \in E \\ 0\ otherwise. \end{cases}$

Definition 7.4.1

Given two real functions f and g defined on the same domain, we have that f(x)=O(g(x)) if there exists two constants k_1 and k_2 such that $f(x) \leq k_1 g(x) + k_2$ for all x in the domain.

Definition 7.4.2

An algorithm is said to be polynomial if it need, in the worst case, a computing time $f(d) = O(d^k)$, where d is the size of the instance and k is a fixed constant.

Definition 7.4.3

A problem is said to be polynomial (or easy) if there exists a polynomial algorithm solving it.

Definition 7.4.4

A polynomial certificate is defined as an auxiliary piece of information that can be used to verify, in time polynomial in the size of the instance, the correctness of the answer for a given instance.

Definition 7.4.5

A decision problem belongs to the NP (Non-deterministic Polynomial) class if every of its instances with "yes" answer admits a polynomial certificate.

Definition 7.4.6

A decision problem belongs to the co-NP (NP complete) class if every of its instances with "no" answer admits a polynomial certificate.

Definition 7.4.7

Problem P_1 can be reduced in polynomial time to problem $P_2 (P_1 \propto P_2)$ if there exists an algorithm to solve P_1 that:

- uses as black-box procedure any algorithm for P_2;
- is polynomial in the hypothesis that the algorithm for P_2 needs O(1) time.

Definition 7.4.8

A problem $P_2 \in NP$ is said to be NP-complete if $P_1 \propto P_2$ for all $P_1 \in NP$.

Theorem 7.4.1

The problem: Given a directed graph D=(N,A), does there exists an Hamiltonian circuit D'=(N',A') in D? is NP-complete.
Proof:
The problem belongs to the NP class, given that one of its instances with "yes" answer can be certified by exhibiting a Hamiltonian circuit in D.

Minimum Spanning Tree

$$\begin{cases} min \underbrace{\Sigma_{e \in E} c_e x_e}_{total\ cost} \\ \underbrace{\Sigma_{e \in E} x_e = n - 1}_{n-1\ chosen\ edges} \\ \underbrace{\Sigma_{e \in E} x_e \leq |S| - 1}_{cycle\ elimination\ condition} \quad \forall S \subseteq V : S \neq \emptyset \\ x_e \geq 0\ integer, e \in E \end{cases}$$

Theorem 7.5.1

Let $G_{T^*} = (V, T^*)$ be a minimum spanning tree. For any $e \in E$ we have that $e \in T^*$ if and only if exists an $S \subset V$ such that $e = argmin\{c_f : f \in \delta(S)\}$
Proof:

First, we will prove that the condition is sufficient,i.e.m that "$e = argmin\{c_f : f \in \delta(S)\}$" for any cut $\delta(S) \Rightarrow e \in T^*$. Let us assume by contradiction that $e \notin T^*$. The set of edges $T^* \cup \{e\}$ contains a cycle $C[T^*, e]$ that crosses the cut $\delta(S)$ at least twice. Choose then any edge $f \in C[T^*, e] \cap \delta(S), f \neq e$ and observe that $c_e < c_f$ given that $f \in \delta(S) \setminus \{e\}$. But then the set of edges $T := (T^* \setminus \{f\}) \cup \{e\}$ defines a tree $G_T = (V, T)$ with cost $c(T) = c(T^*) - c_f + c_e < c(T^*)$, which implies that T^* is a minimum spanning tree.
Let us prove now that the condition is also necessary, i.e., that $e \in T^* \Rightarrow$ there exists $S \subset V$ *s.t.* $e = argmin\{c_f : f \in \delta(S)\}$. To that end, it is sufficient to choose as set S one of the two connected components of the partial graph $G' = (V, T^* \setminus \{e\})$, and observe that if there existed $f \in \delta(S) \setminus \{e\}$ with $c_f < c_e$, then $G_{T^*} = (V, T^*)$ would not be a minimum spanning tree, given that the alternative tree $G_T = (V, T)$ with set of edges $T := (T^* \setminus \{e\}) \cup \{f\}$ would have cost $c(T) = c(T^*) - c_e + c_F < c(T^*)$.
In a similar way, it is also possible to prove that a given tree $G_{T^*} = (V, T^*)$ is a minimum spanning tree if and only if, for all $e \notin T^*$ and for all $f \in C[T^*, e] \setminus \{e\}$, we have $c_f \leq c_e$.

Theorem 7.6.1

The problem of identifying a simple path with minimum cost in a graph is NP-hard.
Proof:
It is sufficient to observe that a simple path cannot have more than n arcs, and that a simple path has n arcs if and oly if it is a Hamiltonian circuit. It follows that the problem of identifying a Hamiltonian circuit in G can be expressed as the problem of identifying the simple path from s:=1 to t:=1 containing the largest possible number of arcs, i.e., the simple path with minimum cost when each arc $(i, j) \in A$ is given a cost $c_{ij} = -1$. There exists hence an NP-complete problem that can be reduced i polynomial time to the minimum-cost simple path problem, which is thus NP-hard.

Shortest Path Problem

$$\begin{cases} min \underbrace{\Sigma_{(i,j)\in A} c_{ij} x_{ij}}_{\text{Path cost}} \\ \underbrace{\Sigma_{(h,j)\in\delta^+(h)} x_{hj}}_{\text{n. leaving arcs}} - \underbrace{\Sigma_{(i,h)\in\delta^-(h)} x_{ih}}_{\text{n. entering arcs}} = \begin{cases} 1 \; if \; h = s \\ -1 \; if \; h = t \\ 0 \; \forall h \in V \setminus \{s, t\} \end{cases} \\ \underbrace{\Sigma_{(i,j)\in A(S)} x_{ij}}_{\text{n. arcs in S}} \leq |S| - 1, \forall S \subseteq V : S \neq \emptyset \\ 0 \leq x_{ij} \leq 1 \text{ integer}, (i, j) \in A \end{cases}$$

Theorem 7.6.2 Correctness of Dijkstra's Algorithm

Assume that the cost L_i of the shortest paths from s to each vertex i belonging to a given set $S \subset V$ with $s \in S(L_s := 0)$ is known, and let $(v, h) := argmin\{L_i + c_{ij} : (i, j) \in \delta^+(S)\}$. If $c_{ij} \geq 0$ for all $(i, j) \in A$, then $L_v + c_{vh}$ is the cost of the shortest path from s to h.
Proof:
$L_v + c_{vh}$ is the cost of a path form s to h formed by the shortest path from s to v (with cost L_v) followed by arc (v,h). We have to prove that any other path P from s to h has cost c(P)$\geq L_v + c_{vh}$. Let (i,j) be the first arc in $P \cap \delta^+(S)$, and as partition P in $P_1 \cup \{(i, j)\} \cup P_2$, where P_1 and P_2 are two paths from s to i and from j to h, respectively. We have then $c(P) = \underbrace{c(P_1)}_{\geq L_i} + c_{ij} + \underbrace{c(P_2)}_{\geq 0} \geq L_i + c_{ij} \geq L_v + c_{vh}$.

Theorem 7.6.3

Consider matrix d obtained performing the triangular operation on vertices 1,...,h. In the absence of negative cycles, for all $i, j \in V$, value $d[i, j]$ is the cost of the shortest path from i to j in the subgraph induced by the set of vertices

$\{1, ..., h\} \cup \{i, j\}$.
Proof:
We are gonna perform an induction on h. For h=0, the matrix d coincides with the initial matrix of the costs of the arcs, hence $d[i, j] = c[i, j]$ represents the cost of the shortest path from i to j in the subgraph induced by $\{i, j\}$, as claimed. Let us now suppose that the property is verified at iteration h-1. Consider a shortest path P_{ij} from i to j in the subgraph induced by $\{1, .., h\} \cup \{i, j\}$, and let $c(P_{ij})$ be its cost. Two cases may occur:

- P_{ij} does not apss through vertex h: in this case $c(P_{ij}) = d[i, j]$ by the inductive hypothesis.
- P_{ij} passes thorugh h. We can partition P_{ij} in this case in $P_{ih} \cup P_{hj}$, where P_{ih} is a shortest path from i to h in the subgraph induced by $\{1, ..., h-1\} \cup \{i, h\}$, and P_{hj} is a shortest path form h to j in the subgraph induced by $\{1, ..., h-1\} \cup \{h, j\}$. By the inductive hypothesis, we have that $c(P_{ih}) = d[i, h]$ and $c(P_{hj}) = d[h, j]$, and hence $c(P_{ij}) = d[i, h] + d[h, j]$.

Therefore it holds that $c(P_{ij}) = min\{\underbrace{d[i, j]}_{\text{case 1}}, \underbrace{d[i, h] + d[h, j]}_{\text{case 2}}\}$. Thus, after the triangular operation on h, we have $d[i, j] = c(P_{ij})$, as requested.

Definition 7.6.1

A project is a set of activities $A_i (i = 1, .., m)$ each having a known duration $d_i \geq 0$. Among some activities, possible precedence relations of the type $A_i \prec A_j$ can specified (meaning that activity A_j can begin only after the completion of activity A_i).

Definition 7.7.1

Givn a flow network, a feasible flow form s to t is a function $x : A \to R$ such that

$$0 \leq x_{ij} \leq k_{ij}, (i, j) \in A \ (7.3)$$
$$\underbrace{\Sigma_{(h,j)\in\delta^+(h)} x_{hj}}_{\text{flow leaving from h}} - \underbrace{\Sigma_{(i,h)\in\delta^-(h)} x_{ih}}_{\text{flow entering into h}} = 0, h \in V \setminus \{s, t\} \ (7.4)$$

Max Flow

$max : \{\phi_0 := \Sigma_{(s,j)\in\delta^+(s)} x_{sj} - \Sigma_{(i,s)\in\delta^-(s)} x_{i,s} : \text{constraints (7.3)-(7.4)}\}$

Min cost Flow

$min\{\Sigma_{(i,j)\in A} c_{ij} x_{ij} : \text{constraints (7.3)-(7.4)}\}$

Definition 7.7.2

We call cut a partition $(S, V \setminus S)$ of the set of vertices such that $s \in S$ and $t \in V \setminus S$.

Definition 7.7.3

Given a feasible flow x, we call flow through a cut $(S, V \setminus S)$ the quantity $\phi(S) := \underbrace{\Sigma_{(i,j)\in\delta^+(S)} x_{ij}}_{\text{flow leaving from S}} - \underbrace{\Sigma_{(i,j)\in\delta^-(S)} x_{ij}}_{\text{flow entering into S}}$
The quantity $\phi_0 := \phi(\{s\}) = \Sigma_{(s,j)\in\delta^+(s)} x_{sj} - \Sigma_{(i,s)\in\delta^-(s)} x_{is}$ is said to be the value of flow x.

Definition 7.7.4

The capacity of the cut $(S, V \setminus S)$ is the quantity: $k(S) := \Sigma_{(i,j)\in\delta^+} k_{ij}$

Theorem 7.7.1

Let x be a feasible flow. For every cut $(S, V \setminus S)$ one has $\phi(S) = \phi_0$,i.e., the flow though every cut is a constant.
Proof:
Consider an arbitrary cut $(S, V \setminus S)$. Then:
$\phi_0 := \phi(\{s\}) =$
$\Sigma_{(s,j)\in\delta^+(s)} x_{sj} - \Sigma_{(i,s)\in\delta^-(s)} x_{is} = \Sigma_{h\in S} \underbrace{[\Sigma_{(h,j)\in\delta^+(h)} x_{hj} - \Sigma_{(i,h)\in\delta^-(h)} x_{ih}]}_{=0 \text{ for all } h\neq s} =$
$\Sigma_{h\in S}\Sigma_{(h,j)\in\delta^+(h)} x_{hj} - \Sigma_{h\in S}\Sigma_{(i,h)\in\delta^-(h)} x_{ih} =$
$[\underbrace{\Sigma_{(i,j)\in A(S)} x_{ij}}_{=} + \Sigma_{(i,j)\in\delta^+(S)} x_{ij}] - [\underbrace{\Sigma_{(i,j)\in A(S)} x_{ij}}_{=} + \Sigma_{(i,j)\in\delta^-(S)} x_{ij}] =: \phi(S)$

Theorem 7.7.2

For every feasible flow x and every cut $(S, V \setminus S)$, one has $\phi(S) \leq k(S)$.
Proof:

$$\phi(S) := \sum_{(i,j)\in\delta^+(S)} \underbrace{x_{ij}}_{\leq k_{ij}} - \sum_{(i,j)\in\delta^-(S)} \underbrace{x_{ij}}_{\geq 0} \leq \sum_{(i,j)\in\delta^+(S)} k_{ij} =: k(S).$$

MAX-FLOW/MIN-CUT

A feasible flow x is optimal for the MAX-FLOW problem if and only if there exists a cut $(S^*, V \setminus S^*)$ with $\phi(S^*) = k(S^*)$. In this case, $(S^*, V \setminus S^*)$ is a cut of minimum capacity in the given network.

Definition 7.7.5

Let x be a feasible flow. An arc (i,j) is said to be saturated if $x_{ij} = k_{ij}$, empty if $x_{ij} = 0$.

Definition 7.7.6

Let x be a feasible flow. The residual network $\bar{G} = (V, \bar{A})$ associated with x is obtained from the original network G=(V,A) by replacing each arc $(i, j) \in A$ with two arcs:

- a forward arc (i,j) of residual capacity $\bar{k}_{ij} = k_{ij} - x_{ij} \geq 0$
- a backward arc (i,j) of residual capacity $\bar{k}_{ij} = x_{ij} \geq 0$

and then eliminating all arcs with zero residual capacity.

Theorem 7.7.3

A feasible flow x is optimal for the $MAX - FLOW$ problem if and only if vertex t is not reachable from vertex s in the residual network $\bar{G} = (V, \bar{A})$ associated with x.
Proof:
Let ϕ_0 be the value of flow x. If t is reachable form s in $\bar{G}$, there exists then an augmenting path P from s to t in $\bar{G}$. Setting $\delta := min\{\bar{k}_{uv} : (u, v) \in P\} > 0$, for all $(u, v) \in P$ it is possible to update $x_{uv} := x_{uv} + \delta$ if (u,v) is a forward arc; $x_{vu} := x_{vu} - \delta$ if (u,v) is a backward arc. It is easy to verify that the new vector x defines a feasible flow of value $\phi_0 + \delta$ in the original network, which proves that the starting solution x was not optimal for the MAX-FLOW problem. Let us now assume that t is not reachable from s in $\bar{G}$. There hence exists a cut $(S^*, V \setminus S^*)$ in the residual network $\bar{G}$ such that $\delta^+_{\bar{G}}(S^*) = \emptyset$ is saturated. By the definition of residual network we have that, in the original network G:

- each arc $(i,j) \in \delta_g^+(S^*)$ is saturated
- each arc $(i,j) \in \delta_g^-(S^*)$ is empty

It follows that

$$\phi(S^*) := \underbrace{\sum_{(i,j)\in\delta_G^+(S^*)} x_{ij}}_{\text{all saturated}} - \underbrace{\sum_{(i,j)\in\delta_G^-(S^*)} x_{ij}}_{\text{all empty}} = \sum_{(i,j)\in\delta_G^+(S^*)} k_{ij} =: k(S^*)$$

hence the optimality of x derives form theorem 7.7.2, which also guarantees that $(S^*, V \setminus S^*)$ is a cut of minimum capacity in the original network.

Some NP-hard Problems

Knapsack Problem

$$x_j = \begin{cases} 1 \text{ if item j is selected} \\ 0 \text{ otherwise} \end{cases}$$

The ILP problem is:

$$\begin{cases} z^* := \max \sum_{j=1}^{n} p_j x_j \\ \sum_{j=1}^{n} w_j x_j \leq W \\ 0 \leq x_j \leq 1 \text{ integer}, j \in \{1, ..., n\} \end{cases}$$

Traveling salesman problem

$$x_{ij} = \begin{cases} 1 \text{ if arc } (i,j) \in A \text{ is chosen in the optimal circuit} \\ 0 \text{ otherwise} \end{cases}$$

We obtain the following model

$$\begin{cases} \min \underbrace{\sum_{(i,j) \in A} c_{ij} x_{ij}}_{\text{circuit cost}} \\ \underbrace{\sum_{(i,j) \in \delta^-(j)} x_{ij}}_{\text{one arc entering j}} = 1, j \in V \\ \underbrace{\sum_{(i,j) \in \delta^+(i)} x_{ij}}_{\text{one arc leaving i}} = 1, i \in V \\ \underbrace{\sum_{(i,j) \in \delta^+(S)} x_{ij}}_{\text{reachability from 1}} \geq 1, S \subset V : 1 \in S \\ x_{ij} \geq 0, \text{ integer}, (i,j) \in A \end{cases}$$

Steiner tree problem

$$x_{ij} = \begin{cases} 1 \text{ if arc } (i,j) \in A \text{ is selectedt} \\ 0 \text{ otherwise} \end{cases}$$

We obtain the following model

$$\begin{cases} min \underbrace{\sum_{(i,j)\in A} c_{ij}x_{ij}}_{\text{Solution cost}} \\ \underbrace{\sum_{(i,j)\in\delta^-(j)} x_{ij}}_{\text{n. of arcs entering j}} = \begin{cases} 1\ \forall j \in T \\ -0\ if\ j = r \\ \leq 1\ \forall j \in V \setminus (T \cup \{r\}) \end{cases} \\ \sum_{(i,j)\in\delta^+(S)} x_{ij} \geq \sum_{(i,j)\in\delta^-(t)} x_{it} \text{ for all } S \subset V : r \in S, \text{ and for all } t \in V \setminus S \\ x_{ij} \geq 0 \text{ integer}, (i,j) \in A \end{cases}$$

Plant location

$$x_{ij} = \begin{cases} 1 \text{ if user i connects to location j} \\ 0 \text{ otherwise} \end{cases} \qquad y_j = \begin{cases} 1 \text{ if location j is activated} \\ 0 \text{ otherwise} \end{cases}$$

We obtain the following model

$$\begin{cases} min \underbrace{\sum_{j=1}^{m} d_j y_j}_{\text{fixed cost}} + \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}}_{\text{connection cost}} \\ \underbrace{\sum_{j=1}^{m} x_{ij}}_{\text{each user connects to exactly one location}} = 1, i \in \{1,...,n\} \\ \underbrace{x_{ij} \leq y_j}_{\text{connection i-j}\Rightarrow\text{location j activated}} , i \in \{1,...,n\}, j \in \{1,...,m\} \\ x_{ij} \geq 0 \text{ integer}, i \in \{1,...,m\}, j \in \{1,...,m\} \\ 0 \leq y_j \leq 1 \text{ integer }, j \in \{1,...,m\} \end{cases}$$

Set covering

$$\begin{cases} min c^T x \\ Ax \geq 1 \\ 0 \leq x \leq 1 \text{ integer} \end{cases}$$

Algorithms

Simplex

Let $\beta[1], ..., \beta[m]$ be the indexes of the columns of an initial feasible basis;
unbounded = false;
optimal = false;
while optimal = false and unbounded = false:
    let $B := [A_{\beta[1]}|...|A_{\beta[m]}]$ be the current feasible basis;
    compute B^{-1} and set $u^T := c_B^T B^{-1}$;
    compute the reduced cost $\bar{c}_h := c_h - u^T A_h$ of non-basic variables x_h;
    if $\bar{c}_h \geq 0 \forall x_h$ non-basic then optimal = true;
    else
        choose a non-basic variable x_h with $\bar{c}_h < 0$;
        compute $\bar{b} := B^{-1}b$ and $\bar{A}_h := B^{-1}A_h$;
        if $\bar{a}_{ih} \leq 0 \forall i \in \{1, ..., m\}$ then unbounded = true;
        else
            compute t=argmin$\{\bar{b}_i/\bar{a}_{ih}, i \in \{i, ..., m\} : \bar{a}_{ih} > 0\}$;
            update the current basis by setting $\beta[t] = h$;

Pivot operation

save=$\bar{a}[t, h]$; #value of the pivot element;
$for\ j = 0\ to\ n$: $\bar{a}[t, j] = \bar{a}[t, j]/save$;
$for\ i = 0\ to\ m$:
    if $(i \neq t)$ and $(\bar{a}[i, h] \neq 0)$
        save=$\bar{a}[i, h]$;
        for j=0 to n: $\bar{a}[i, j] = \bar{a}[i, j] - save * \bar{a}[t, j]$

Two phase method simplex

Let $\beta[1], ..., \beta[m]$ be the indexes of the columns of an initial feasible basis;
Create the initial tableau $\bar{A} = \{\bar{a}[i, j] : i = 0, ..., m; j = 0, ..., n\}$, in canonical form w.r.t. the initial basis;
unbounded = false;
optimal = false;
while optimal = false and unbounded = false:
    if $\bar{a}[0, j] \geq 0 \forall j = 1, ..., n$ then optimal = true;
    else
        choose a non-basic variable x_h with $\bar{a}[0, h] < 0$;
        if $\bar{a}[i, h] \leq 0 \forall i = 1, ..., m$ then unbounded = true;
        else
            compute t=argmin$\{\bar{a}[i, 0]/\bar{a}[i, h] : i \in \{i, ..., m\}, \bar{a}[i, h] > 0\}$;
            pivot(t,h);
            $\beta[t] = h$;

Results of the two phase method

1. $w^* > 0$. In this case, there exist no solution to the artificial problem with $y_i = 0 \forall i \in \{1, ..., m\}$(such solution would have cost w=0), hence the system Ax=b, $x \geq 0$, does not admit a solution: the original problem is hence infeasible.

2. $w^* = 0$. In this case we have $y^* = 0$, since $y^* \geq 0$ *and* $\Sigma_{i=1}^{m} y_i^* = 0$, hence x^* is a feasible solution of the original problem. With respect to the optimal tableau for the artificial problem, two sub cases may occur:

   a All the artificial variables are non-basic variables: eliminating the columns of the tableau corresponding to the artificial variables, we have then a tableau for the original problem, already in canonical form w.r.t to a basis. Now the artificial objective function is converted in canonical form, and the original problem is solved by the simplex algorithm.

   b There exists a basic variable y_h on a row i: in this case we have $y_h^* = 0$ (since $w^* = 0$), hence we are in the presence of degeneracy. If there exists a value $\bar{a}_{ij} \neq 0$ in correspondence to a variable $x_j (j \leq n)$, then it is sufficient to perform a pivot operation on element (i,j). Since $\bar{b}_i = 0$, this operation is admissible even if $\bar{a}_{ij} < 0$ and does not entail variations to the objective function w. In this way, y_h leaves the basis. Repeating the procedure for all the basic y_h leads back to the non-degenerate case (a).
   One last possibility remains: all values $\bar{a}_{i1}, ..., \bar{a}_{in}$ are equal to zero. In this case, eliminating the artificial columns we obtain a tableas equivalent to the original one, but with a zero row. This implies that the matrix A does not have full row rank m, and thus the i-th row of the current tableau may be eliminated without problems.

Cutting Plane Algorithm

solve the continuous relaxation $min\{c^T x : Ax = b, x \geq 0\}$, and let x^* be the optimal basic solution found;
while x^* not integer:
    solve the separation problem of x^* from X, identifying an inequality $\alpha^T x \leq \alpha_0$ valid for X but violated by x^*;
    add constraint $\alpha^T x \leq \alpha_0$ to the current continuous relaxation;
    solve the current continuous relaxation (with dual simplex) and let x^* be the new optimal basic solution identified.

Branch and Bound

m:=1;parent[1] := 0; $q := \emptyset$; z_{OPT} := heuristic solution value (possibly $+\infty$)
solve the continuous relaxation $min\{c^T x : Ax = b, x \geq 0\}$, and let x^* be the optimal solution found;
$LB[1] := c^T x^*$;
if (x^* integer) and ($c^T x^* < z_{OPT}$):
    $x_{OPT} = x^*; z_{OPT} := c^T x^*$
if $LB[1] < z_{OPT}$:
    choose the fractional branching variable x_h^*;
    $vbranch[1] := h; value[1] := x_h^*$;
    $Q := \{1\}$
while $Q \neq \emptyset$:
    choose a node $t \in Q$, and set $Q := Q \setminus \{t\}$;
    $h := vbranch[t]; val := value[t]$;
    for child:=1 to 2:
        m:=m+1;
        if child = 1:
            $parent[m] := t$;
            else $parent[m] := -t$;
        define problem PL_m associated with node m;
        solve problem PL_m and let x^* be the optimal solution found;
        $LB[m] := c^T x^*$;
        if (x^* integer) and ($c^T x^* < z_{OPT}$):

$x_{OPT} := x^*; z_{OPT} := c^T x^*;$
$Q := Q \setminus \{j \in Q : LB[j] \geq z_{OPT}\}$
if $LB[m] < z_{OPT}$:
choose the fractional branching variable x_k^*;
$vbranch[m] := k; value[m] := x_k^*$
$Q := Q \cup \{m\}$

Minimum Spanning Tree

$T^* := \emptyset$
while $|T^*| \neq n-1$:
identify a set $S \subset V$ s.t. $\delta(S) \cap T^* = \emptyset$, and let $e = argmin\{c_f : f \in \delta(S)\}$;
$T^* := T^* \cup \{e\}$

Prim-Dijkstra's Algorithm $O(n^2)$

$flag[1] := 1; pred[1] := 1;$
for $j = 2$ *to* n:
$flag[j] := 0;$
$L[j] := c[i, j];$
$pred[j] := 1;$
for k:=1 to n-1:
$min = +\infty$
for j:=2 to n:
if $(flag[j] = 0)$ *and* $(L[j] < min)$:
$min := L[j];$
$h := j$
$flag[h] := 1$ (We extend S)
for j:=2 to n:
if $(flag[j] = 0)$ *and* $(c[h, j] < L[j])$:
$L[j] := c[h, j];$
$pred[j] := h;$

Kruskal's Algorithm

sort the edges of G according to the increasing costs, obtaining: $Edge[1].cost \leq ... \leq Edge[m].cost$;
k:=0; h:=0;
for i:=1 to n:
$comp[i] := i;$
while $(k < n-1)$ and $(h < m)$:
h:=h+1;
$i := Edge[h].From;$
$j := Edje[h].To;$
$C_1 := comp[i];$
$C_2 := comp[j];$
if $C_1 \neq C_2$:
k:=k+1;
$Tree[k] := Edge[h];$
for q:=1 to n:
if$comp[q] = C_2$:
$comp[q] := C_1;$
if $k \neq k-1$: "Graph is not connected"

Path finding

for j:=1 to n
    $pred[j] := 0$
$pred[s] := s$;
$Q := \{s\}$
while $Q \neq \emptyset$:
    choose a vertex $h \in Q$ and set $Q := Q \setminus \{h\}$
    for each $(h, j) \in \delta^+(h)$:
        $if pred[j] = 0$:
            $pred[j] := j$;
            $Q := Q \cup \{j\}$;

Dijkstra $O(n^2)$

for j:=1 to n:
    $flag[j] := 0;\ pred[j] := s;\ L[j] := c[s, j]$;
$flag[s] := 1$;
$L[s] := 0$;
for k:=1 to n-1:
    $min := +\infty$;
    for j:=1 to n:
        $if\ (flag[j] = 0)\ and\ (L[j] < min)$:
            $min := L[j]$;
            $h := j$;
    $flag[h] := 1$;
    for j:=1 to n:
        $if\ (flag[j] = 0)\ and\ (L[h] + c[h, j] < L[j])$:
            $L[j] := L[h] + c[h, j]$;
            $pred[j] := h$;

Floyd-Warshall's algorithm

for i:=1 to n:
    for j:=1 to n:
        initialize $d[i, j] := c[i, j]\ and\ pred[i, j] := i$;
for h:=1 to n:
    for i:=1 to n:
        for j:=1 to n:
            $if\ d[i, h] + d[h, j] < d[i, j]$:
                $d[i, j] := d[i, h] + d[h, j]$;
                $pred[i, j] := pred[h, j]$;
    for i:=1 to n:
        $if\ d[i, i] < 0$ then STOP; (We have a negative cycle)

Critical path method CPM

$TMIN[1] := 0$; sort the vertices topologically;
for h:=2 to n:
    $TMIN[h] := max\{TMIN[i] + d[i, h] : (i, h) \in \delta^-(h)\}$;
$TMAX[n] := TMIN[n]$; (minimum duration of the project)
for h:=n-1 down to 1:
    $TMAX[h] := min\{TMAX[j] - d[h, j] : (h, j) \in \delta^+(h)\}$;

Ford-Fulkerson algorithm

```
Ford-Fulkerson algorithm ;
  begin
1.   for each (i, j) ∈ A do x[i, j] := 0; /* initial null flow */
     φ0 := 0;
2.   repeat /* MAIN LOOP */
3.      for each j := 1 to n do pred[j] := 0 ;
        ε[s] := +∞ ; pred[s] := s ;
        Q := {s} ; /* queue of labeled yet unprocessed vertices */
        while (Q ≠ ∅) and (pred[t] = 0) do
           begin
4.            pop a vertex h ∈ Q from the queue (e.g., h := min{i : i ∈ Q}) ;
              Q := Q \ {h} ;
5.            for each (h, j) ∈ δ+G(h) : x[h, j] < k[h, j] do /* unsaturated forward arcs */
                 if pred[j] = 0 then
                    begin
                       ε[j] := min{ε[h], k[h, j] − x[h, j]} ; pred[j] := h ;
                       Q := Q ∪ {j}
                    end ;
6.            for each (i, h) ∈ δ−G(h) : x[i, h] > 0 do /* non-empty backward arcs */
                 if pred[i] = 0 then
                    begin
                       ε[i] := min{ε[h], x[i, h]} ; pred[i] := −h ;
                       Q := Q ∪ {i}
                    end
           end ;
7.      if pred[t] ≠ 0 then /* augmenting path found */
           begin
8.            δ := ε[t] ; φ0 := φ0 + δ ;
              j := t ;
9.            while j ≠ s do
                 begin /* backward reconstruction of the augmenting path */
                    i := pred[j] ;
                    if i > 0
                       then x[i, j] := x[i, j] + δ
                       else x[j, −i] := x[j, −i] − δ ;
                    j := |i|
                 end
           end
10.  until pred[t] = 0 ;
11.  "the current flow x is optimal, and a cut (S*, V \ S*) of minimum capacity
       is the one associated with S* := {j ∈ V : pred[j] ≠ 0}"
  end .
```