Combinatorial Online-Optimization in Practice

Jörg Rambau / Luis M. Torres

Modeling

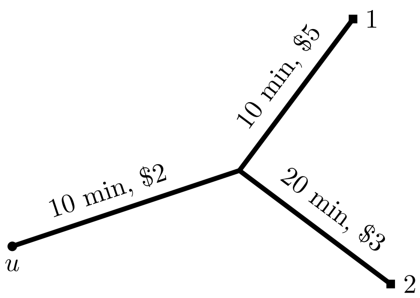
1. Consider this more general version of the "ADAC problem": there is a set requests $V = \{1, \dots n\}$ located at some points in the plane. Some of them, like in the case presented in the lecture, are served just by moving a unit to them. However, there are another requests demanding a damaged vehicle to be transported to a garage (which may be different for each request). Model this problem in graph theoretical terms for the following cases:

   (a) There is only service one unit and no time windows.

   (b) More than one units and no time windows.

   (c) More than one units and hard time windows.

2. The picture shows a (trivial) instance of a vehicle routing problem. Located on the road map are one service unit u and two events 1 and 2. For each road segment, the traveling time (in minutes) and traveling cost (in dollars) is given. The first event needs to be attended within 50 minutes, the second one within one hour. Suppose, at first, that the service of an event occurs "instantly" when the unit arrives to it, i.e., it takes no time and has no cost. We want to find a feasible route of minimum cost for u.



   (a) Model the problem as a hamiltonian shortest path problem with resource constraints in an undirected graph. Characterize (in words) which are the feasible solutions and find the best by inspection.

   (b) Construct the time expanded network for this graph. For this purpose, discretize the time in intervals of 10 minutes. Again, characterize the feasible solutions and find the optimum by inspection. How is the condition "every event hast to be visited exactly once" stated in this expanded network?

(c) Suppose that the time intervals in the problem data are increased, or, equivalently, that we need to choose a more precise discretization of the time (5 minutes, for example). What happens in both cases to the time expanded network? This is why the algorithms that use this approach are called only *pseudopolynomial*.

(d) How can the model be improved to handle the service time and service costs of the events?

Easy and hard problems

1. Let $D = (V, A)$ be a directed graph and $c : A \to \mathbb{R}_+$ a nonnegative cost function on the arcs.

   (a) Suppose a set of labels $\{y_v : v \in V\}$ has been found for all the vertices in V such that
   $$y_v + c_{vw} \geq y_w \tag{1}$$
   holds for all $vw \in A$. Let s and t be two arbitrary nodes in V. Prove that any path from s to t has a cost larger or equal than $y_t - y_s$.

   (b) Prove that the distance labels found by Djikstra's algortihm satisfy equation (1) for all arcs in A.

   (c) Use the last observation to prove the correctness of Djikstra's algortihm and show that it can be implemented to run in time $O(n^2)$.

2. A *topological sort* of a digraph $D = (V, A)$ is a set $\{\ell_v : v \in V\}$ of labels for its nodes having the property that $\ell_v < \ell_w$ holds for all arcs $vw \in A$. A digraph has a topological sort if an only if it doesn't contain any (directed) cycle.

   (a) Give an algorithm for finding a topological sort in an acyclic digraph and analyze its complexity.

   (b) Suppose that D is stored as a list $L^+(v)$ of successors and a list $L^-(v)$ of predecessors for each node v in such a way, that it is possible to remove a node in time $O(|L^+(v)| + |L^-(v)|)$. Improve (if necessary) your last algorithm and use it to solve the shortest path problem in time $O(m)$ (where m is the number of arcs of the digraph).

3. The following problem is known as the *knapsack problem.* There are given n distinct objects of values $\{a_1, \ldots, a_m\}$ and weights $\{w_1, \ldots, w_m\}$. You have a knapsack of limited capacity W (with $W < \sum w_i$) and want to choose as much value as possible to transport, i.e., you are looking for a subset of objects such that the sum of their weights is not larger than than W and the sum of their values is maximal. This problem is known to be NP-hard.

   Show that this problem can be (polynomially) transformed into a shortest path problem with one resource constraint and a negative cost function in an acyclic digraph. This proves that the resource constrained shortest path problem is NP-hard, (why?) even for the case when there is only *one* resource.

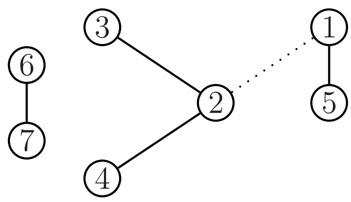
Hint: Introduce two nodes for each of the objects, asociated to the decision "take it" or "do not take it".

4. We will next consider Kruskal's algorithm for the MST. (We keep the same notation used in the lecture). The first step of the algorithm (sorting the nodes according to their weights) can be implemented to run in time $O(m \log m)$ using well-known sorting procedures (for example HEAPSORT) which we will not discuss here. Let us focus on how long it takes to execute the "main loop",i.e., adding new edges of the tree and testing if they create circuits.

   (a) A very simple implementation is the following: we maintain for each node v a label $C(v)$ indicating the connected component of T to which v belongs. (All these labels are kept in a vector). Initially, $C(v) = v$ for all $v \in V$. To prove if adding an edge $e_i = uv$ creates a circuit in T, we can compare $C(u)$ and $C(v)$. After adding an edge uv, however, we have to take care to update the vector C: either change every $C(u)$ with $C(v)$ or the other way around. The figure shows C and T before and after inserting a new edge.

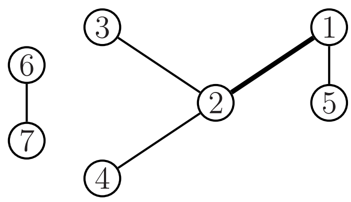
   Before inserting edge 12**:**

v	C
1	2
2	1
3	1
4	1
5	2
6	3
7	3



After inserting edge 12**:**

v	C
1	**1**
2	1
3	1
4	1
5	**1**
6	3
7	3



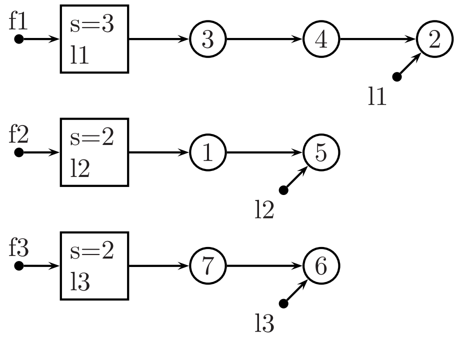
Show that this implementation runs in time $O(n^2)$.

(b) The algorithm can be accelerated by using a more advanced data structure to store the connected components: every connected component will be maintained as a linked list. For every node v of the graph, we keep a pointer $f(v)$ to a linked list associated with its component. The first element of the this list contains information about the size s of the list, and a pointer to the last element.

When an edge is added, the lists corresponding to the two components being connected are merged (check that this can be done in constant time) and the pointer information is changed *only for the nodes in the smaller component*. The next figure shows these data structures for the same example from above.

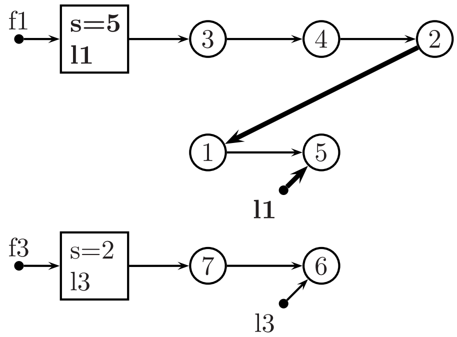
Before inserting edge 12**:**

v	f
1	f2
2	f1
3	f1
4	f1
5	f2
6	f3
7	f3



After inserting edge 12**:**

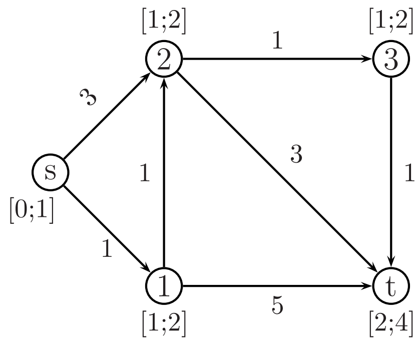
v	f
1	**f1**
2	f1
3	f1
4	f1
5	**f1**
6	f3
7	f3



Show that with this new implementation, the main loop of the algorithm runs in time $O(m \log n)$. This gives an overall running time of $O(m \log m)$.

5. Consider the instance of the shortest path problem with time windows given in the following picture. The interval at each node shows the time window

when a vehicle is allowed to visit that node. The numbers showed on the arcs are the travel costs. Assume all travel times are equal to one.



Find the shortest path from s to t that respects all the time windows. To do this, follow the idea of the time expanded network: construct a new digraph with one copy of each node for each unit of time of its time windows. "Translate" the original arcs into this time expanded network. Finally, solve the problem using Djisktra's algorithm.

Basics: Polynomial algorithms revisited

Throughout this session, we will discuss some topics on the matching problem and the well-known Edmond's algorithm. In all exercises, let $G = (V, E)$ be an undirected graph and M a matching in G.

1. Let A a subset of nodes. A connected component of the graph $G \setminus A$ which has an odd number of vertices will be called an *odd component* of $G \setminus A$. Moreover, let us denote by $\operatorname{oc}(G \setminus A)$ the number of such odd components in $G \setminus A$.

   Prove that, for any matching M in G, the following holds:

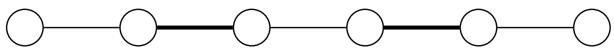
   $$|M| \leq \frac{1}{2}\left(|V| - \operatorname{oc}(G \setminus A) + |A|\right).$$

   In particular, this means that if there is a set of nodes A with $\operatorname{oc}(G \setminus A) > |A|$, then there is no matching that can cover all nodes in V.

   Hint: Count the M-exposed nodes (i.e., the nodes not covered by the matching).

   In fact, if M is a matching of maximum cardinality, then there is a set of nodes A for which the last expression holds with equality. This is known as the Tutte-Berge Formula.

2. A (simple) path in G is called an M-alternating path if it consists alternately of edges belonging to M and not, like in the figure (the bold edges are matching edges).

   An M-alternating path between two M-exposed nodes u and v is called an M-*augmenting* path. Show that M is a matching of maximum cardinality if and only if there are no M-augmenting paths in G.

3. A matching is called perfect if it covers all the nodes of the graph. In the next exercises, we are going to design a polynomial algorithm for constructing perfect matchings. Let us start with the case when G is a bipartite graph.

   Suppose we are given some non-perfect matching in G and consider the following marking procedure CreateTree.

Algorithm 1 CreateTree

Input: A graph $G = (V, E)$, a matching M in G.
Output: An M-alternating tree T.

Choose an M-exposed node r and mark it with B;
Set $V(T) = \{r\}$, $E(T) = \emptyset$;
while there exists $vw \in E$ with v marked as B and w not marked **do**
    if w is M-exposed **then**
        return T. STOP;
    end if
    if w is M-covered **then**
        Let wz be the matching edge covering w;
        Mark w with A and z with B;
        Add w and z to $V(T)$, and vw, wz to $E(T)$;
    end if
end while
return T. STOP.

(a) Show that CreateTree is correctly defined, i.e., that it ends after finitely many steps, that it does not mark any node more than once, and that it constructs a subgraph T of G which is a tree. Note that since M is non-perfect, at least one node is marked (Why?) The tree T is called an M-*alternating* tree, since the paths going from the root to any other node of the tree are alternating paths. Observe that no edge of the tree connects two nodes having the same mark.

(b) Show that after CreateTree has ended one of the two following stop conditions must hold:

Either: The neighbours of every node marked as B are all marked as A

or: There is an M-augmenting path in G.

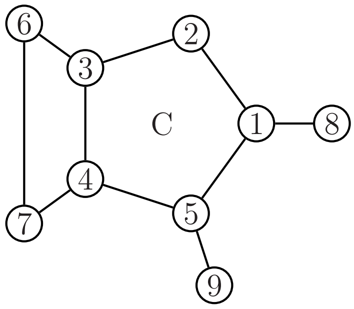
Hint: Remember that G is bipartite.

(c) Using the results of exercise one, show that if the first stop condition ("Either") holds, then G has no perfect matching.

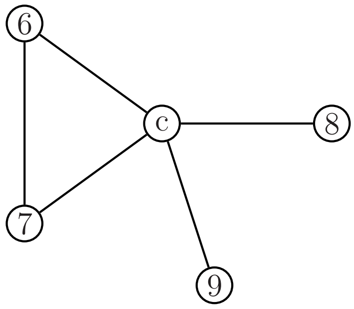
(d) Put all these ideas together to construct an algorithm for finding a perfect matching in a given bipartite graph G or stating that G has no perfect matching. Where does the algorithm fail when G is not bipartite?

4. Given an odd circuit C (i.e., a circuit with an odd number of edges), we define the following operation on G: we delete from the graph all nodes in

$V(C)$ and replace them by a new node c, then we connect this node to all the neighbours of the vertices of C. The next figure illustrates this procedure.



G



$G \times C$

We call this operation *to shrink the odd circuit* C. The new graph will be denoted by $G \times C$ and the new node c is usually called a *pseudonode*, to differentiate it from the "original" nodes in G.

(a) Show that any perfect matching M' in $G \times C$ can be used to construct a perfect matching in G. We call this operation *to extend a matching*.

(b) Show with a counterexample that the converse is not true: a perfect matching in G cannot always be "shrinked" to a perfect matching in $G \times C$. In particular, this means that it is not possible just to shrink all odd circuits of G until we obtain a bipartite graph G', then solve the matching problem in G' with the last algorithm and finally extend the solution to G.

5. Despite of the last observation, the idea of shrinking odd circuits and extending matchings can be used to formulate an algorithm for finding perfect matchings on general graphs. This is the idea of the famous BlossomAlgorithm due to Edmonds described below.

   (a) Show that the shrinking step is well defined, i.e., that it preserves the structure of T as an M-alternating tree.

   (b) Prove that the BlossomAlgorithm terminates after a finite number of steps.

   (c) Show that after the BlossomAlgorithm has ended, it either finds a perfect matching in G or gives a proof that such a matching does not exist. (**Hint:** Observe that none of the nodes marked as B can be a pseudonode. Why?)

Algorithm 2 BlossomAlgorithm

Input: A graph $G = (V, E)$,
Output: A perfect matching M in G, if it exists.

Set $M = M' = \emptyset$, $G' = G$;
Choose an M'-exposed node r and mark it with B;
Set $V(T) = \{r\}$, $E(T) = \emptyset$;
while there exists $vw \in E$ with v marked as B and w not marked as A **do**
  if w is M'-exposed **then**
    Use the M'-augmenting path from r to w to enlarge M';
    Extend M' to a matching M in G;
    Clear all node marks;
    Set $M' = M$, $G' = G$;
    if there is no M'-exposed node in G' **then**
      Return perfect matching M' and STOP;
    else
      Choose an M'-exposed node r and mark it with B;
      Set $V(T) = \{r\}$, $E(T) = \emptyset$;
    end if
  end if
  if w is M'-covered and w is not marked as B **then**
    Let wz be the matching edge covering w;
    Mark w with A and z with B;
    Add w and z to $V(T)$, and vw, wz to $E(T)$;
  end if
  if w is M'-covered and w is marked as B **then**
    Let C be the circuit obtained in T by adding vw to $E(T)$;
    Set $G' = G' \times C$, $T = T \times C$;
    Mark the new pseudonode c with B;
  end if
end while
STOP. G has no perfect matching.

Geometric Descriptions of Optimization Problems

We will study in this session valid inequalities for polytopes associated to various combinatorial problems.

1. Let $x_1, \dots, x_k$ be some points in $\mathbb{R}^n$ that satisfy a given linear inequality H of the form $a^T x \leq b$. Furthermore, let

$$P := \operatorname{conv}\{x_1, \dots, x_k\}$$

be the polytope defined by the convex hull of these points. Show that any $y \in P$ also satisfies H. What happens with equalities?

2. Given a graph $G = (V, E)$, the perfect matching polytope $\mathrm{PMP}(G)$ of G is defined as:

$$\mathrm{PMP}(G) := \operatorname{conv}\big\{x \in \mathbb{R}^E \,:\, x \text{ is the characteristic vector of a perfect matching in } G\}$$

Prove that any $x \in \mathrm{PMP}(G)$ satisfies the following set of equalities and inequalities:

$$\sum_{e \in \delta(v)} x_e = 1, \forall v \in V$$
$$\sum_{e \in \delta(S)} x_e \geq 1, \forall S \subseteq V \text{ with } |S| \text{ odd},$$
$$0 \leq x_e \leq 1, \forall e \in E.$$

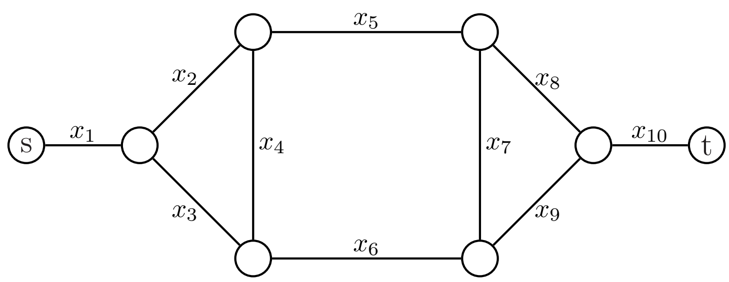
3. In the lectures, we presented a set of inequalities defining the *Path-and-Cycles-Polytope* associated to an undirected graph G.
   (a) Show that every incidence vector of a hamiltonian (s,t)-path satisfies all the inequalities of the system.
   (b) It is known that the Path-and-Cycles-Polytope is *integral*, i.e., that its vertices have only integral coordinates. Since the problem of finding a hamiltonian (s,t)-path in a graph is NP-hard, it follows that the Path-and-Cycles-Polytope must contain some other integral points which are not the incidence vectors of hamiltonian paths. (Why?) Find some examples.

(c) Following an usual approach in polyhedral combinatorics, we will now try to refine our linear description of the polytope by finding some new valid inequalities that *separate* invalid solutions. Show that the incidence vector x of an (s,t)-path satisfies the following set of inequalities:

$$\sum_{e \in \delta(S)} x_e \geq 2, \forall S \subset V \setminus \{s,t\}\,.$$

(d) Show that after adding the last inequalities, any vector x satisfying the new linear system and *having integral coordinates* is the incidence vector of an (s,t)-path. Let us call this new polytope the *Path-No-Subtour-Polytope.*

(e) Unfortunately, by adding new inequalities, we have also created new vertices with fractional coordinates in the polytope (so-called *fractional vertices*). To see this, consider the Path-No-Subtour-Polytope associated to the following graph:



(The labels indicate how the edges will be encoded in a characteristic vector x). Show that the point:

$$x_0^T = (1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1)$$

belongs to the polytope. On the other side, consider the hyperplane given by the linear function

$$H : x_1 + 2x_2 + 2x_3 + 2x_4 + x_5 + x_6 + 2x_7 + 2x_8 + 2x_9 + x_{10} = 10.5$$

Prove that all integral points of the Path-No-Subtour-Polytope (i.e., all incidence vectors of (s,t)-paths lie on one side of H and that x_0 lies on the other side. What can you conclude from the result of exercise one?

Basic concepts on Polyhedrons

1. Consider the $\mathcal{V}$-polytope P in $\mathbb{R}^2$ defined by $P := \text{conv}\{Y\}$, where

$$Y = \left\{(-4,2)^T, (-3,-2)^T, (0,4)^T, (0,2)^T, (3,-2)^T, (4,0)^T, (5,2)^T\right\}$$

   (a) Draw the points of Y and the polytope P in a diagram.

   (b) Apply one step of the *Double Description Method* to find a set of points $Y^{/2}$ such that:

$$\text{conv}Y^{/2} = \text{conv}Y \cap \left\{x \in \mathbb{R}^2 : x_2 = 0\right\}$$

   Do not calculate all coordinates. Just do it for the first point and find the other ones graphically. Observe that the algorithm produces some redundant points. Getting rid of them is not a trivial task.

2. Let us now consider the reciprocal problem: given the $\mathcal{H}$-polytope P in $\mathbb{R}^2$ defined by the following inequalities,

$$\begin{aligned} -x_1 + 2x_2 &\leq 10 \\ x_1 - x_2 &\leq 4 \\ x_1 + 3x_2 &\leq 15 \\ -x_1 - 4x_2 &\leq 2 \\ -x_1 &\leq 4 \end{aligned}$$

   apply one step of the *Fourier-Motzkin elimination* method to find a set of inequalities defining the projection of P on the hyperplane $\{x \in \mathbb{R}^2 : x_2 = 0\}$. Draw a diagram illustrating how the algorithm works.

3. Consider the $\mathcal{H}$-polytope P defined by the following system of inequalities:

$$\begin{aligned} -x_1 + x_2 &\leq 2 \\ x_1 + 2x_2 &\leq 4 \\ -3x_1 - 2x_2 &\leq 6 \\ 3x_1 - 4x_2 &\leq 12 \end{aligned}$$

   (a) Introduce four new variables $w_1, \ldots, w_4$ and write P as the *intersection* of a new $\mathcal{H}$-polytope

$$C_0(A) := \left\{ \begin{pmatrix} x \\ w \end{pmatrix} \in \mathbb{R}^6 : Ax \leq w \right\}$$

   and four hyperplanes.

(b) Using the following identity

$$\begin{pmatrix} x \\ w \end{pmatrix} = \sum_{i=1}^{2} |x_i| \operatorname{sign}(x_i) \begin{pmatrix} e_i \\ Ae_i \end{pmatrix} + \sum_{j=1}^{4} (w_j - (Ax)_j) \begin{pmatrix} 0 \\ e_j \end{pmatrix}$$

write $C_0(A)$ as a conic combination of a set of vectors in $\mathbb{R}^6$. This gives a description of $C_0(A)$ as a $\mathcal{V}$-polytope. How would you proceed to obtain from this a description of P as a $\mathcal{V}$-polytope?

4. Again, let's look at the converse problem. Given the $\mathcal{V}$-polytope in $\mathbb{R}^2$,

$$P := \operatorname{conv}\left\{(-2,2)^T, (1,-2)^T, (4,4)^T, (5,1)^T\right\}$$

find a $\mathcal{H}$-polytope P' in $\mathbb{R}^6$ such that P is a projection of P'.

Linear Programming

1. Consider the following pair of dual linear programs:

$$
\text{(LP)} \quad \begin{aligned} \max\ c^T x \quad & \text{s.t.} \\ Ax &\le b \\ x &\ge 0 \end{aligned}
\qquad\qquad
\text{(DP)} \quad \begin{aligned} \min\ y^T b \quad & \text{s.t.} \\ y^T A &\ge c^T \\ y &\ge 0 \end{aligned}
$$

where $A \in \mathbb{R}^{m \times n}$; $x, c \in \mathbb{R}^n$; and $y, b \in \mathbb{R}^m$.

(a) Prove that every feasible solution of DP is an upper bound on the value of the optimal solution for LP and, conversely, every feasible solution of LP is a lower bound on the value of the optimum for DP. This property is called *weak duality.*

(b) Knowing that

$$\max\left\{c^T : Ax \le b\right\} = \min\left\{y^T b : y^T A = c^T,\, y \ge 0\right\}$$

(provided both sets are non-empty) prove that the optimal values of LP and DP do in fact coincide. This property is called *strong duality.*

(c) Prove the *complementary slackness conditions* for any pair of optimal solutions $\tilde{x}$ and $\tilde{y}$ to LP and DP:

$$
\begin{aligned}
(c^T - \tilde{y}^T A)\tilde{x} &= 0 \\
\tilde{y}^T (b - A\tilde{x}) &= 0
\end{aligned}
$$

2. Given an undirected graph $G = (V, E)$, a famous result of Edmonds says that the problem of finding a perfect matching of minimum weight in G can be stated as the following linear program:

$$\min \sum_{e \in E} x_e\, c_e$$

s.t.

$$
\begin{aligned}
\sum_{e \in \delta(v)} x_e &= 1 && \forall v \in V, \\
\sum_{e \in \delta(S)} x_e &\ge 1 && \forall S \subseteq V \text{ with } |S| \text{ odd}, \\
x_e &\ge 0 && \forall e \in E.
\end{aligned}
$$

(a) Interpret this linear program in graph-theoretical terms.

(b) Show that if G is bipartite, then the inequalities referring to odd sets of nodes are redundant.

(c) Write down the dual linear program for the case when G is bipartite. Explain what the duality theorem implies for this problem.

(d) Use complementary slackness to derive optimality conditions for a pair of primal and dual solutions. Can you turn this conditions into the idea for an algorithm to solve the minimum weight perfect matching problem?

3. Given the directed graph $D = (V, A)$ with cost function c on the arcs, the problem of finding a shortest path between two nodes s and t can be stated as the following linear program:

$$\min \sum_{a \in A} x_a \, c_a$$

s.t.

$$\begin{aligned}
&\sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 \quad && \forall v \in V \setminus \{s, v\}\,, \\
&\sum_{a \in \delta^-(s)} x_a = 1, && \\
&\sum_{a \in \delta^+(t)} x_a = 1, && \\
&x_a \geq 0 && \forall a \in A.
\end{aligned}$$

Explain the restrictions in graph-theoretic terms. Formulate the dual problem and interpret its meaning. Analize the complementary slackness conditions.

Large Scale Integer Programming

Polyhedral studies play an important role when solving large scale integer programs. The knowledge about a specific polytope can be exploited to generate *good* cutting planes and tighten the linear bounds that appear during the branch-and-bound process. This enhanced method is sometimes called *Branch-and-Cut*.

As an example of possible cutting planes, we will consider in this session two kinds of linear inequalities present in the stable set polytope associated to an undirected graph $G = (V, E)$.

Recall that the problem of finding a stable set of maximal cardinality in G can be stated as the following integer program:

$$\begin{aligned}
&\max \sum_{v \in V} x_v \\
&s.t. \\
&\quad x_v + x_w \leq 1 \quad \forall vw \in E, \qquad (2)\\
&\quad x_v \in \{0, 1\}.
\end{aligned}$$

The constraints (2) are usually called *edge inequalities.* The feasible solutions to this problem are incidence vectors of stable sets in G. Their convex hull is called the *stable set polytope* and usually denoted by STAB(G).

1. A *clique* is a set of nodes of a graph which are all pairwise adjacent. Associated to each clique Q in G is the *clique inequality*:

$$\sum_{v \in Q} x_v \leq 1$$

   (a) Prove that the clique inequality is valid for the stable set polytope, i.e., that the incidence vector of any stable set in G satisfies this inequality.

   (b) Show that this inequality cannot be obtained from the edge inequalities.

   (c) Recall the definition of *facet*: an inequality $a^T x \leq \alpha$ is said to define a facet of a polyhedron P if the following holds

$$\dim \left(\left\{ x \in P : \, a^T x = \alpha \right\} \right) = \dim(P) - 1$$

   Facet-defining inequalities are very important in a linear description of a polyhedron. In fact, it can be shown that the set of all these inequalities forms a *complete and non redundant* linear description of the polyhedron. Find conditions for clique inequalities to be facet defining.

Hint: Prove at first that $\dim(\mathrm{STAB}(G)) = |V|$ and try to find $|V|$ affine independent vectors that satisfy the clique inequality with equality.

(d) Suppose you are given a fractional vector $x^* \in \mathbb{R}^{|V|}$. How would you detect if x^* violates some clique inequality? This is called the *separation* problem.