Advanced algorithms

Exercise sheet #8 — Primal-Dual & approximation algorithms

November 23, 2022

Exercice 1 (Multicut Problem in Trees).

Let $G = (V, E)$ be an undirected weighted graph with integer weights $c_e \geq 0$ for each edge $e \in E$, and let $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ be k specific pairs of vertices.
A *multicut* is a set of edges F such that for all i, s_i and t_i are in different connected components of $G' = (V, E \setminus F)$. The weight $c(F)$ of F is the sum of the weight of its elements. The goal of *minimum multicut* problem is to find a multicut of minimum weight.
Recall in exercise sheet #6 we gave a 2-approximation algorithm for a slightly different problem called *multiterminal cut* where we were given a set of k vertices instead of pairs of vertices. Here, we need to deal with the fact that there may be paths connecting s_i and s_j or s_i and t_j for $i \neq j$. The multicut problem is known to be NP-hard (you need not to prove it), so there is no hope in this session to find a polynomial time algorithm to solve it. The goal of this exercise is to use a Primal-Dual method to design an approximation algorithm in the special case where G is assumed to be a tree.

(a) Let $\mathcal{P}_i$ be the set of all paths P joining s_i and t_i for some $1 \leq i \leq k$. What is the cardinality of $\mathcal{P}_i$?

Consider the following linear progam:

$$\begin{aligned} \text{minimize } & \sum_{e \in E} c_e x_e \\ \text{subject to } & \sum_{e \in P} x_e \geqslant 1, \quad \forall P \in \mathcal{P}_i, 1 \leq i \leq k, \\ & x_e \geq 0, \quad \forall e \in E, \end{aligned}$$

with the optimal value λ^* and an optimal solution (x_e^*).

(b) Let F^* be a minimum multicut of G. Show that $c(F^*) \geq \lambda^*$.

(c) We introduce a variable f_i for each pair (s_i, t_i). Give the dual of this linear program.

Suppose we root the tree G at an arbitrary vertex r. Let *depth*(v) be the number of edges on the path from v to r for any vertex v. The depth of the root is 0. Similarly, if two edges e_1 and e_2 belong to the same path from a vertex to the root, if e_1 occurs before e_2 on this path, e_1 is said to be deeper than e_2. Denote by $lca(s_i, t_i)$ the *least common ancestor* of s_i and t_i, *i.e.* the vertex v on the path from s_i to t_i whose depth is minimum.

Consider the following algorithm:

1 Set $f_i \leftarrow 0$ for all $i = 1, \dots, k$ and $F \leftarrow \emptyset$

2 For each $v \in V$, in non-increasing order of depth, do:

  (i) For each i such that $v = lca(s_i, t_i)$, increase f_i such that some inequality constraint in the dual becomes an equality (saturation of edges).

  (ii) Add to F all edges that were saturated.

3 Let $e_1, \dots, e_l$ be the list of edges in F in the order they were added.

4 **Reverse delete** For $j = l$ downto 1 do:

  (i) If $F \setminus \{e_j\}$ is a multicut in G, then remove e_j from F.

5 Output F.

(d) Show that the set F output by the algorithm is a multicut of G, and (f_i) is a feasible dual solution.

(e) Show that for every pair (s_i, t_i) such that $f_i > 0$, F contains at most two edges from the path $s_i \to t_i$.

(f) Deduce that this algorithm achieves an approximation factor of 2 for the multicut problem in trees.

Exercice 2 (Maximizing Ad-Auctions Revenue (Pâle 2015))**.**

The aim of this exercice is to analyze a system of one shot auctions used for selling advertisment slots on the web. The auction we want to study is organized by a provider that needs to decide every day to which of his n clients he will sell each of his m advertisment slots. To make his choice, the provider knows the daily budget $B(i)$ of each client $i = 1, \dots, n$ and the bids $b(i, j)$, $i = 1, \dots, n$, $j = 1, \dots, m$ made independantly by each client for each slot.

In the *offline* variant, the $B(i)$ and the $b(i, j)$ are all known from the beginning of the day and the problem is for the provider to determine the best possible assignment of the slots to the clients. Beware that it is not a standard "best bid wins" auction: with 2 clients with budget $B(1) = B(2) = 3$, if the first client offers $b(1, 1) = 3$ and $b(1, 2) = 2$ and the second offers $b(1, 1) = 2$ and $b(1, 2) = 0$, it is more interesting to sell slot 1 to client 2 than to client 1 in order to be able to sell slot 2 to client 1.

(a) Show that with 3 clients having each a budget $B(i) = 4$ and 6 slots, if the bids of clients are given by the following matrix $b(i, j)$:

$$\begin{pmatrix} 3 & 3 & 2 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 & 2 & 2 \\ 2 & 2 & 2 & 2 & 3 & 3 \end{pmatrix}$$

then in the optimal solution none of the slot is sold to the client proposing the highest bid.

(b) Give a modelization of the offline problem as an integer linear program. Use variables $y(i, j)$ in $\{0, 1\}$ to indicate whether slot j is sold to client i.

(c) Give an interpretation of the real linear program $\mathcal{P}$ in which the constraint $y(i, j) \in \{0, 1\}$ on variables is replaced by $y(i, j) \in \mathbb{R}$, $y(i, j) \geqslant 0$?

(d) Write the dual linear program $\mathcal{D}$ of the linear program $\mathcal{P}$: use variables $x(i)$ indexed by clients and variables $z(j)$ indexed by slots. You are NOT asked to give an interpretation to $\mathcal{D}$.

In the *online* variant, the $B(i)$ are known from the beginning but slots become available one after the other in the order $j = 1, \ldots, m$. When slot j becomes available, the clients make their bids $b(i, j)$, $i = 1, \ldots, n$ and the provider has to decide immediately to whom he sells the slot.

(e) Give a pair of instances of the problem with $n = m = 2$ that shows that no online algorithm can reach the optimum solution in all cases.

Consider the following algorithm, using variables $x(i)$, $y(i, j)$ and $z(j)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$:

1. Initially, for each bidder i, set $x(i) \leftarrow 0$.
2. For all $j = 1, \ldots, m$, when slot j becomes available, the provider tries to sell it to the client i_j that maximises $b(i, j)(1 - x(i))$.
   (i) if $x(i_j) < 1$, then
   - Sell the slot to client i_j at a price equal to the minimum between his bid $b(i_j, j)$ and his remaining budget, set $y(i_j, j) \leftarrow 1$ and for all $i \neq i_j$, $y(i, j) \leftarrow 0$.
   - Set $z(j) \leftarrow b(i_j, j)(1 - x(i_j))$.
   - Update

   $$x(i_j) \leftarrow x(i_j)\left(1 + \frac{b(i_j, j)}{B(i_j)}\right) + \frac{b(i_j, j)}{(c-1) \cdot B(i_j)}$$

   where $c > 1$ is a constant to be determined later.

   (ii) Otherwise the provider does not sell this slot and set: $z(j) \leftarrow 0$, $y(i, j) \leftarrow 0$ for all i.

The aim of the rest of the exercice is to prove that it is possible to choose the constant c so that this algorithm be α-competitive for some α.

(f) Show that after selling the k-th slot, variables $x(i)$, $i = 1, \ldots, n$ and $z(j)$, $j = 1, \ldots, k$ satisfy the contraints of the dual program $\mathcal{D}$.

(g) Deduce from the previous question that the final values of the variables $x(i)$ and $z(i)$ satisfy

$$O^* \leqslant \sum_i B(i)x(i) + \sum_j z(j)$$

where O^* denotes the optimal profit of the provider in the *offline* variant of the problem with the same $B(i)$ and $b(i, j)$.

(h) Show that for all i and j the values $x_j(i)$ and $x_{j+1}(i)$ of the variable $x(i)$ just before and just after the treatment of the jth slot and the final values of the variables $y(i, j)$ and $z(j)$ satisfy

$$\left(\sum_i B(i)(x_{j+1}(i) - x_j(i))\right) + z(j) \leqslant (1 + \tfrac{1}{c-1}) \sum_i b(i,j)y(i,j)$$

and conclude that

$$O^* \leqslant (1 + \tfrac{1}{c-1}) \sum_{i,j} b(i,j)y(i,j).$$

It could be tempting to conclude here that we have proved the algorithm to be $(1 - \frac{1}{c})$-competitive. However this would not be correct because in some cases the provider charges the client with a lower price than the price $b(i,j)$ he offered. The sum in the right hand side of the previous inequality is the sum of the prices that were offered by the clients for the slots they bought, while we are really interested in the actual total revenue of the provider.

(i) Show that condition $(*)$ below ensures that each client benefits at most once a day from a better price than the one he offered:

If after selling slot k to client i_k the following holds,

$$\sum_{j=1}^{k} b(i_k, j)y(i_k, j) \geqslant B(i_k) \qquad (*)$$

then, from then on, $x(i_k) \geqslant 1$.

(j) Show that if we take $c = (1 + R_{\max})^{1/R_{\max}}$ with $R_{\max} = \max_{i,j}\left(\frac{b(i,j)}{B(i)}\right)$, then Condition $(*)$ is verified. For this, show by induction that the inequality

$$x(i) \geqslant \frac{1}{c-1}\left(c^{\frac{\sum_j b(i,j)y(i,j)}{B(i)}} - 1\right)$$

remains true during all the execution of the algorithm, using the fact that, for this choice of c,

$$c^{\frac{b(i,j)}{B(i)}} \leqslant 1 + \frac{b(i,j)}{B(i)} \qquad \text{for all } i, j.$$

(k) Deduce from the previous two questions that for all i,

$$\sum_j b(i,j)y(i,j) \leqslant B(i) + \max_j(b(i,j)),$$

and conclude that the provider obtains from this client a gain at least

$$\left(\sum_j b(i,j)y(i,j)\right) \frac{B(i)}{B(i) + \max_j(b(i,j))} \geqslant \left(\sum_j b(i,j)y(i,j)\right)(1 - R_{\max})$$

(l) What is finally the competitivity factor of the algorithm?