Thomas Rothvoß
Location: ELD120 Discussion: 12.05.10

Exercises

Approximation Algorithms

Spring 2010

Sheet 10

Exercise 1
We consider the SET COVER problem, where we are given sets $S_1,\dots,S_m \subseteq U$ and each set S_i has cost $c(S_i)$. The goal is to find a set $I \subseteq \{1,\dots,m\}$ such that $\bigcup_{i\in I} S_i = U$ and $\sum_{i\in I} c(S_i)$ is minimized. We assume that each element is contained in at most f many different sets. The primal LP relaxation and the corresponding dual are:

$$\begin{array}{rcll} \min \sum_{i=1}^{m} x_i c(S_i) & & & (P) \\ \sum_{i: j\in S_i} x_i & \geq & 1 & \forall j \in U \\ x_i & \geq & 0 & \forall i = 1,\dots,m \end{array} \qquad \begin{array}{rcll} \max \sum_{j\in U} y_j & & & (D) \\ \sum_{j\in S_i} y_j & \leq & c(S_i) & \forall i = 1,\dots,m \\ y_j & \geq & 0 & \forall j \in U \end{array}$$

We say an element j is *covered*, if $j \in \bigcup_{i:x_i=1} S_i$. A set S_i is called *tight* w.r.t. y, if its dual constraint is satisfied with equality, i.e. $\sum_{j\in S_i} y_j = c(S_i)$. Consider the following primal-dual algorithm

(1) $x := \mathbf{0}, y := \mathbf{0}$

(2) WHILE not all elements covered DO

  (3) Choose an arbitrary uncovered element j

  (4) Increase y_j until a set S_i becomes tight

  (5) Set $x_i := 1$

Let x, y be the values at the end of the algorithm. Perform the following tasks

i) Show that x is a feasible solution for (P) and y is a feasible solution for (D).

ii) Argue why $x_i > 0 \Rightarrow \sum_{j\in S_i} y_j = c(S_i)$.

iii) Show that
$$\sum_{i=1}^{m} x_i c(S_i) \leq f \cdot \sum_{j\in U} y_j$$
Hint: This is a special case of the ”Relaxed Complementary Slackness lemma” from the lecture.

iv) Argue why the algorithm gives an f-approximation.

Solution:

i) The WHILE loop ends only, when all elements are covered. Then x is a feasible solution to (P). After setting $x_i := 1$, all elements in S_i are covered and their dual variable will never raised again. At the beginning, y is a feasible solution. We never raise y so that a constraint is violated. Hence y is also feasible.

ii) We set $x_i = 1$ only if the set becomes tight, i.e. $\sum_{j\in S_i} y_j = c(S_i)$ at this point in the algorithm. But then, all elements in S_i are covered, i.e. no y_j will be raised anymore for any $j \in S_i$. That means $\sum_{j\in S_i} y_j = c(S_i)$ still holds at the end of the algorithm.

iii) We have either $x_i = 0$ or $\sum_{j\in S_i} y_j = c(S_i)$ (by ii))

$$\sum_{i=1}^{m} x_i c(S_i) = \sum_{i=1}^{m} x_i \sum_{j\in S_i} y_j = \sum_{j\in U} y_j \underbrace{\sum_{i:j\in S_i} x_i}_{\leq f} \leq f \cdot \sum_{j\in U} y_j$$

iv) The algorithm gives an f-apx because

$$APX = \sum x_i c(S_i) \leq f \cdot \sum y_j \overset{y \text{ is feasible for}(D)}{\leq} f \cdot OPT_f \leq f \cdot OPT.$$

Exercise 2

For the MULTICUT ON TREES problem, we are given a tree T on nodes V with edge costs $c : T \to \mathbb{Q}_+$ and terminal pairs $(s_1,t_1),\ldots,(s_k,t_k)$ $(s_i,t_i \in V)$. The goal is to find a subset $D \subseteq T$ of minimal cost, such that any pair (s_i,t_i) is separated. Let $P_i \subseteq T$ be the unique s_i-t_i path. Then

$$OPT = \min_{D\subseteq T} \left\{ \sum_{e\in D} c_e \mid \forall i = 1,\ldots,k : |D \cap P_i| \geq 1 \right\}$$

A primal LP relaxation and the corresponding dual look as follows:

$$\begin{array}{lrcll} \min \sum_{e\in T} x_e c_e & & & & (P) \\ & \sum_{e\in P_i} x_e & \geq & 1 & \forall i = 1,\ldots,k \\ & x_e & \geq & 0 & \forall e \in T \end{array} \qquad \begin{array}{lrcll} \max \sum_{i=1}^{k} f_i & & & & (D) \\ & \sum_{i:e\in P_i} f_i & \leq & c_e & \forall e \in E \\ & f_i & \geq & 0 & \forall i = 1,\ldots,k \end{array}$$

We root the tree T at an arbitrary node $r \in V$ and denote by $a(s_i,t_i)$ that node of the s_i-t_i path that is closest to the root r. Assume that the pairs are sorted such that the distances of $a(s_i,t_i)$ to r are **non-increasing** in i (for example in the figure below one has $j < i$). Consider the following primal-dual algorithm:

(1) $x := \mathbf{0}$, $f := \mathbf{0}$

(2) FOR $i = 1,\ldots,k$ DO

  (3) Increase f_i until some edge e becomes tight (i.e. $\sum_{i:e\in P_i} f_i = c_e$)

  (4) FOR all edges e that became tight DO $x_e := 1$

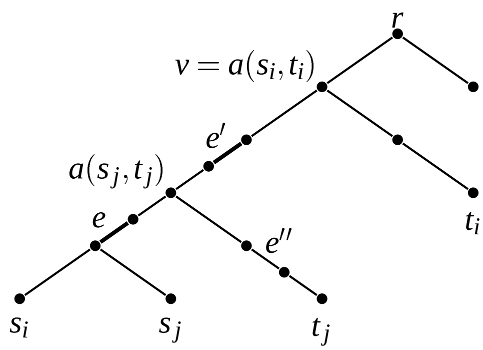
(5) *Reverse delete:* FOR all e with $x_e = 1$, in reverse order, in which they were set to 1 DO

(6) IF x remains feasible THEN set $x_e := 0$

We now analyse the algorithm (x and f are the vectors at the *end* of the algorithm):

i) Argue why x is a feasible primal and f is a feasible dual solution.

ii) Let $D = \{e \mid x_e = 1\}$. Consider a pair (s_i, t_i) with $f_i > 0$. Show that at most one edge e from the s_i-$a(s_i, t_i)$ path is in D.
Hint: In fact, this is a bit tricky. Assume for contradiction that you have two edge $e, e' \in D$ on the s_i-$a(s_i, t_i)$ path.



Then argue that there is an s_j-t_j pair, where $a(s_j, t_j)$ lies between e and e'. Finally say why at the time, when e was *not* deleted, there was an edge e'' in D with $e'' \in P_j$, $e'' \notin P_i$. This should give a contradiction.

iii) Show: If $f_i > 0$, then $\sum_{e \in P_i} x_e \leq 2$.

iv) Show that $\sum_{e \in T} x_e c_e \leq 2 \sum_{i=1}^{k} f_i$.
Hint: This is a special case of the ”Relaxed Complementary Slackness lemma” from the lecture.

v) Which approximation factor does the algorithm give?

vi) Suppose you want to approximate the INTEGRAL MULTI COMMODITY FLOW PROBLEM IN TREES, where a tree T with integral edge capacities $c(e) \in \mathbb{N}$ and terminal pairs s_i-t_i is given. The goal is to route an maximum amount of integer flow that does not violate the capacities. In other words, the aim is to find

$$OPT = \max_{f_1, \ldots, f_k \in \mathbb{Z}_+} \left\{ \sum_{i=1}^{k} f_i \mid \forall e \in T : \sum_{i : e \in P_i} f_i \leq c_e \right\}$$

How would you approximate this problem?

Solution:

i) It suffices to show that x is feasible after the FOR loop in (2): Consider any (s_i,t_i) pair. In iteration i, f_i is raised, until an edge on P_i becomes tight. Then for this edge (and maybe others) one has $x_e := 1$. In other iterations maybe x is more increased, but in any case $\sum_{e\in P_i} x_e \geq 1$. The dual solution f is feasible at the beginning. We also don't increase the values more than allowed. Hence f is dual feasible.

ii) Suppose for contradiction, that 2 edges e, e' are on the s_i-$a(s_i,t_i)$ path, say e is below e'. Edge e was not removed, hence there must be a pair s_j-t_j such that e is the only picked edge on P_j. But $f_i > 0$ was only raised in iteration i if $e \notin D$ at that point. Hence the growth of f_j was stopped because another edge $e'' \in P_j$ was tight that is not in P_i. But in the deletion phase we check e before e'' and e was not deleted because it should be the only edge in $P_j \cap D$. Contradiction.

	Iteration	
Phase I	j	(2) some edge $e'' \in P_j$ is in D (potentially already added before). But $e'' \notin P_i$ since $f_i > 0$ later.
	$\vdots$	
	i	(1) $f_i > 0$, hence $e \notin D$
	$> i$	(3) $e \in D$ was added
Phase II	$> i$	(4) check e, but e is not removed since via assumption at this point e is only edge from D on P_j. Contradiction because there is still e''.
	$\vdots$	
	j	check e'' for deletion

iii) For any path with $f_i > 0$, we pick at most one edge from the s_i-$a(s_i,t_i)$ path and one from the t_i-$a(s_i,t_i)$ path. Hence $\sum_{e\in P_i} x_e \leq 2$.

iv) We obtain:

$$\sum_{e\in D} x_e c_e \overset{\text{either } x_e=0 \text{ or } e \text{ is tight}}{=} \sum_{e\in D} x_e \sum_{i:e\in P_i} f_i = \sum_{i=1}^{k} f_i \underbrace{\sum_{e\in P_i} x_e}_{\leq 2 \text{ or } f_i=0} \leq 2\sum_{i=1}^{k} f_i.$$

vi) Run the primal-dual algorithm as it is. If the cost/capacities $c(e)$ are integer, the primal-dual algorithms increase the values f_i each time by an integer amount. Hence $f_i \in \mathbb{Z}_+$ at the end. For (D) is the primal for the flow problem and (P) is the dual. Then f is a feasible primal solution and x is a feasible dual solution which is a factor 2 more expensive. Hence f is a 2-approximation.