CO453: Network Design – Winter 2007

Instructor: Chaitanya Swamy

Assignment 3

Due: Monday, February 12, 2007 after class

You must give a proof of correctness of any algorithm you design, and argue briefly why it runs in polynomial time. You may use any proof or algorithm covered in class directly. You may want to wait until after the lecture on Monday, February 5, before starting the assignment.

Throughout $G = (V, E)$ with $|V| = n$, $|E| = m$. Recall that a function $f : 2^V \mapsto \{0, 1\}$ is called *proper* if (i) $f(V) = 0$; (ii) for any $S \subseteq V$, $f(S) = f(V \setminus S)$; and (iii) if A, B are non-empty disjoint subsets of V, then $f(A \cup B) = 1$ implies that $f(A) = 1$ or $f(B) = 1$.

Q1: The performance guarantee of 2 proved for the primal-dual Steiner tree algorithm can be strengthened slightly to $\alpha := 2\big(1 - \frac{1}{|T|}\big)$ (recall that T is the set of terminals). Prove that using this LP-based technique, one cannot hope to prove a better approximation ratio by exhibiting an instance (G, T, c) where the cost of an (integer) optimum Steiner tree is *at least* α times the LP-optimum value, and thus α times the dual constructed, which is our lower bound. **(5 marks)**

The worst-case ratio (over all instances) of the cost of an integer optimum solution to the LP-optimum value is called the *integrality gap* of the linear program. The integrality gap of an LP is a measure of how good the LP-optimum value (or the value of any feasible dual solution) is as a lower bound. Observe that if the integrality gap is at least α, then one cannot hope to prove an approximation ratio better than α by using the value of the LP-optimal solution (or the value of any feasible dual solution) as a lower bound. This is because if we consider an instance $\mathcal{I}$ for which the ratio of the costs of the optimal integer and LP solutions is at least α, then *every* integer solution (including the one our algorithm returns) for $\mathcal{I}$ has cost at least α times our lower bound.

Observe the distinction between the analysis done in class showing that one cannot prove a factor better than $\alpha := 2\big(1 - \frac{1}{|T|}\big)$ for the primal-dual algorithm, and what the question asks you to prove. In class, we compared the cost of the Steiner tree returned against the (integer) optimum; the question here asks you to compare the cost of the integer optimum Steiner tree against the LP-optimum, and asks how good is the *lower bound* provided by the LP-optimum. In fact, notice that if $\mathcal{I}$ is a Steiner tree instance for which the ratio of costs is α, then, as mentioned above, since we also know that the primal-dual algorithm always returns a tree of cost at most α times the LP-optimum, the algorithm would in fact return an optimal integer solution on instance $\mathcal{I}$. However, we are not able to *prove* that it is an optimal Steiner tree because we are comparing our cost against an inferior lower bound, which is the value of the dual constructed.

Q2: Given a subset $T \subseteq V$, the *T-join problem* seeks to find a minimum-cost set of edges F such that nodes in T have an odd degree in F, and nodes not in T have an even degree (including 0) in F. (Such a set F is called a T-join.) For example, when $T = \{s, t\}$, a T-join is is simply an s-t path, and the T-join problem is to find the shortest s-t path.

Give a 2-approximation algorithm for this problem. **(5 marks)**

(Hint: Model the problem by a $\{0, 1\}$-proper function.)

Q3: A function $f : 2^V \mapsto \{0,1\}$ is called *downwards monotone* if (i) $f(V) = 0$; (ii) for any $T \subseteq S \subseteq V$ where $T \neq \emptyset$, $f(S) \leq f(T)$. In this question, we will show that the primal-dual schema yields a 2-approximation algorithm for the network design problem with a $\{0,1\}$ downwards-monotone cut-requirement function. The primal and dual programs are:

$$\text{(P)} \quad \min \sum_{e \in E} c_e x_e$$

$$\text{(1)} \quad \text{s.t.} \sum_{e \in \delta(S)} x_e \geq f(S) \quad \forall S \subseteq V,$$

$$x_e \geq 0 \qquad \forall e \in E.$$

$$\text{(D)} \quad \max \sum_{S} f(S) y_S$$

$$\text{(2)} \quad \text{s.t.} \sum_{S \subseteq V : e \in \delta(S)} y_S \leq c_e \quad \forall e \in E,$$

$$y_S \geq 0 \quad \forall S \subseteq V.$$

The algorithm is the same as that for $\{0,1\}$-proper functions (or Steiner tree) covered in class. We start with an empty set of edges F, and raise the dual variables y_S uniformly for all minimal violated sets (where a set S is violated if $f(S) = 1$ and $F \cap \delta(S) = \emptyset$), until constraint (2) goes tight for some edge $e \in \delta(S)$ where S is a minimal violated set (MVS). When this happens we add edge e to F, update the collection of MVSs, and repeat this process, until there are no violated sets. Then, we perform a reverse-delete step to eliminate redundant edges.

(a) Given a set F of edges, show that the minimal violated sets are precisely the components of F with $f(S) = 1$. **(3 marks)**

(b) Let F be the final solution returned, and $F' \supseteq F$ be the set of edges before the reverse-delete step. Prove that at any stage of the algorithm, if $\mathcal{V}$ is the collection of minimal violated sets, then $\sum_{S \in \mathcal{V}} \delta_F(S) \leq 2|\mathcal{V}|$.

Prove this inequality by following the same pattern as in the proof done in class. Let $F'' \subseteq F'$ be the set of edges added by the algorithm by that stage. Show that if H is the graph obtained from (V, F) by contracting each component of F'' into a single node, then H is a *forest*, that is, it is acyclic (but need not be connected). Then argue that in each connected component of H, there is at most *one* leaf node v such that $f(S_v) = 0$, where S_v is the subset of V contracted to form node v. (This is the only portion of the argument which is slightly different from the proof in class, where we showed that for a $\{0,1\}$-proper function f every leaf node v has $f(S_v) = 1$.) Thus, argue that the average degree of the nodes of H corresponding to the minimal violated subsets is at most 2. **(5 marks)**

(c) Using the inequality proved in part (b), show that the cost of the solution returned is at most twice the value of the dual constructed, and hence, the algorithm is a 2-approximation algorithm.

You may you use the result of part (b) here, even if you are unable to prove it. **(2 marks)**