College of Computer & Information Science — Fall 2007
Northeastern University — Handout 2
CSG399: Algorithmic Power Tools — 2 November 2007

Problem Set 2 (due Tuesday, November 27))

Grading. This problem set has a total of 120 points. We plan to have three problem sets in all, for a total of at least 300 points. From a grading perspective, the total points allocated for problem sets is 200 points. So you should plan on attempting at least 200 points in all, over all problem sets. Any score you earn beyond the basis of 200 points will be bonus points for you.

Collaboration. You are welcome to collaborate on solving these problems. But you must write the solutions on your own. Please cite any collaborators and/or references that help you in arriving at your solutions.

1. (25 points) Synchronized duals for set cover

A primal-dual approach that we have studied is to start with a feasible dual solution and then raise a certain subset of the dual variables until some dual constraint became tight, at which point we set the associated primal variable appropriately. For the Steiner Forest problem in particular, we studied a scheme in which we raise a set of "active" dual variables simultaneously at a uniform rate until some edge became tight.

In this problem, we analyze the same approach for set cover. Consider the following algorithm. Recall that we have a primal variable x_S for each set S in the collection of sets and a dual variable y_e for each element of the universe.

- Set all y_e to 0. Initialize set cover F to $\emptyset$.
- Repeat the following steps until all elements are covered.
  - Raise the dual variable y_e for each uncovered element uniformly until some set S becomes "tight" (i.e., $\sum_{e \in S} y_e = c(S)$).
  - Add S to F and remove any sets all of whose elements have been covered. (If multiple sets become tight, let S denote an arbitrary one tight set.)
- Return F.

Show that the above algorithm has an approximation ratio of at most f, where f is the *maximum frequency*, the maximum number of sets that contain a given element. Show that for any value of the maximum frequency f, there exists an instance for which the algorithm's approximation ratio is at least $f - \varepsilon$, where $\varepsilon > 0$ can be made arbitrarily small.

2. (25 points) Including tight edges in primal-dual algorithm for Steiner Forest

The Goemans-Williamson primal-dual algorithm we studied in class includes only one tight edge in any iteration. We discussed the primal-dual algorithm in which we include all tight edges in an iteration, and in the last step consider edges in arbitrary order, removing any edge that forms a cycle.

Show that if the order of edges in the last step is unspecified, then the above algorithm is $\Omega(n)$-approximate, where n is the number of nodes.

(*Hint:* Devise an example in which we add light-weight edges followed by several heavy-weight edges (all in a single iteration), and then in the edge deletion step, we first remove the light-weight edges since they are part of some cycle.)

3. (25 points) Initial solution for k-median local search

The running time of the local search algorithm for k-median is simply the number of iterations times the time taken for each local search iteration. Since the latter is fixed, it makes sense to keep the number of iterations small. One way toward this is to start with a reasonable initial solution. Assume for simplicity that all points have unit demand and a facility can be potentially opened at any point. Consider the following initial solution.

- $S \leftarrow \emptyset$.
- For i from 1 to k:
  - Let u be a point whose distance to the current set is largest; that is, $d(u, S) \geq d(v, S)$ for all v, where $d(v, S)$ is defined as $\min_{w \in S} d(v, w)$.
  - Add u to S.

(a) Let S be the set of points returned by the above algorithm and let T be an arbitrary set of k points. Prove that for

$$\max_v d(v, S) \leq 2 \cdot \max_v d(v, T).$$

(*Hint:* One approach is to use induction on k, the number of points in S and T. The base case is easy. For the induction step, consider two cases. One case is when there exists a point t in T for which there are at least two points in S, whose closest point in T is t. The other case is when for every point t in T, there is exactly one point in S whose closest point in T is t.)

(b) Using part (a), prove that the above algorithm yields a k-median solution that is $O(n)$-approximate.

Remark: While this is not a good approximation ratio, the fact that it is independent of the value of the distances in the metric makes it useful to show that the running time of local search, used with this procedure, is strongly polynomial time.)

4. (25 points) LPs for multicommodity flow.

Consider the demands version of the multicommodity flow problem. We are given a graph $G = (V, E)$ with capacity on edges given by $c_e, e \in E$, a set of k source-sink pairs (s_i, t_i), $1 \leq i \leq k$ with demands given by d_i, $1 \leq i \leq k$. The goal is to determine the largest f such that $f \cdot d_i$ flow of the ith commodity can be sent from source s_i to sink t_i, for $1 \leq i \leq t$, subject to the flow conservation and capacity constraints.

In our class (when we covered the Leighton-Rao result), we wrote the following dual LP for the path-flow-based LP.

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e d_e \\
\sum_{e \in p_i^j} d_e \quad \geq \quad & l_i \, \forall \; s_i - t_i \text{ path } p_i^j \\
\sum_{1 \leq i \leq k} l_i \quad \geq \quad & 1
\end{aligned} \tag{1}
$$

(a) Write down an alternative linear program for the demands multicommodity flow problem based on edge flows. In particular, have a variable $x^i_{(u,v)}$ for each edge (u,v) and commodity i that represents the flow of commodity i from u to v. Complete the LP by capturing the objective function, the flow conservation constraints, and capacity constraints in terms of these "edge flow" variables.

(b) Write a dual for the linear program from part (a). Compare this computed dual with the LP given by Equation 1.

(c) Using part (b) or otherwise (e.g., using the separation oracle method), show that LP 1 can be solved in polynomial time.

5. (25 points) Unsplittable multicommodity flow

In the multicommodity flow problem we defined in class, we assumed that a flow from a source to a sink can be split among multiple paths. In some applications, especially in the networking domain, one may demand flow of a single commodity to be sent along a single path. We call this the unsplittable multicommodity flow problem.

(a) Show that the unsplittable multicommodity flow problem is NP-hard, even with two commodities.

(*Hint:* Reduce from the partition problem.)

Consider the special case of the demands version of the unsplittable multicommodity flow problem in which the demand of each commodity as well as the capacity of each edge is 1. We study the following randomized rounding algorithm for unsplittable multicommodity flow in this case.

1. Solve the LP relaxation of the problem (which removes the unsplittability constraint).
2. Decompose the flow of each commodity i into a set P_i of at most m flow paths, where m is the number of edges. For each commodity i, select a path at random from P_i with probability equal to the flow of i along the path. Send all the flow for commodity i along this path. Call the resulting flow f.
3. Return the flow obtained by scaling f down by the largest amount of flow going across any edge.

(b) Prove that after step 2 of the above algorithm, the expected flow along any edge is at most 1.

(c) Prove that after step 2, the flow along any edge is at most $O(\log n)$ with probability at most $1/n^3$.

Hint: Use Chernoff bounds. See the theorem on relative error in the following wikipedia entry.

http://en.wikipedia.org/wiki/Chernoff_bounds

(d) Conclude that the above algorithm yields an unsplittable multicommodity flow whose value is at most an $O(\log n)$ factor less than that of an optimal unsplittable multicommodity flow, with probability at least $1/n$.