ORIE 633 Network Flows October 11, 2007

Problem Set 3

Due Date: October 25, 2007

1. Given an undirected graph G, let λ be the value of the global minimum cut. An *α-approximate global min-cut* is a set of vertices $S \subset V$, $S \neq \emptyset$, such that $u(\delta(S)) \leq \alpha\lambda$. We will be thinking about applying the random contraction algorithm from class to the problem of finding α-approximate global min-cuts.

   Suppose that α is a half-integer greater than 1 (e.g. $\frac{3}{2}$, 2, $\frac{5}{2}$, ...), and is a fixed constant.

   (a) Prove that the probability that a given α-approximate global min-cut survives contraction of the graph from n vertices down to 2α vertices is at least $1/\binom{n}{2\alpha}$.

   (b) Use the previous item as the basis for an algorithm that outputs an α-approximate global min-cut with probability at least $1/\left((2\alpha)^{2\alpha}\binom{n}{2\alpha}\right)$.

   (c) Argue that there can be at most $O((2n)^{2\alpha})$ α-approximate global min-cuts.

   (d) (Bonus problem) Give an algorithm that finds all α-approximate global min-cuts. Get the best running time that you can (more points for lower running times).

2. When we discussed the maximum flow problem, we showed that any s-t flow could be decomposed into at most m s-t paths and cycles. In this problem, you will show that any circulation f can be decomposed into at most m cycles. In particular, given a circulation f, show that there exists a set $\mathcal{C}$ of cycles and weights $w : \mathcal{C} \rightarrow \Re^+$ such that

   (a) $f_{ij} = \sum_{C \in \mathcal{C}:(i,j) \in C} w(C)$ for all $(i,j) \in A$ with $f_{ij} > 0$;

   (b) $|\mathcal{C}| \leq m$.

3. Let $G = (V, A)$ be a directed graph, with costs c_{ij} for all $(i,j) \in A$.

   (a) Using Problem 2 of Problem Set 2, show that the value

   $$\mu = \min_{\text{cycles } \Gamma \in G} \frac{c(\Gamma)}{|\Gamma|}$$

   can be computed in $O(mn)$ time.

   (b) Given the computation above, show how to find the cycle Γ for which

   $$\mu = \frac{c(\Gamma)}{|\Gamma|}$$

   in $O(n^2)$ time.

4. In this problem we will consider an alternate algorithm for the minimum-cost circulation problem. Recall from class that we have a minimum-cost circulation f iff G_f has no negative cost cycles. But some negative cost cycles are easier to find than others. For example, consider a cycle in which each edge has negative cost. It can be checked in $O(n)$ time whether there is such a cycle for any cost function c. Call an arc (i, j) admissible with respect to potentials p if $c^p_{ij} < 0$. A cycle is admissible if it consists entirely of admissible arcs. Consider the following algorithm:

Admissible Cycle Cancelling

```
1    Initialize potentials p and find a feasible circulation f
2    While f is not a minimum-cost circulation
3        Repeatedly find and cancel admissible cycles until none exist
4        Update potentials p so that c^p_ij ≥ −ε(f) for all (i, j) ∈ A_f
             for the current circulation f.
```

(a) We need to give a procedure for updating the potentials in step 4 of the algorithm. Recalling the proof that $\mu(f) = -\epsilon(f)$, give such a procedure.

(b) Prove the following claim: when updating the potentials, $\epsilon(f)$ has decreased by a factor of $(1 - 1/n)$ since the last update.

(c) Prove the following claim: in each pass through step 3, at most m cycles are cancelled.

(d) Prove the following claim: at most $O(n \log(nC))$ iterations of the while loop of step 2 are needed to obtain an optimal circulation f.

(e) Prove that the overall running time of the algorithm is $O(mn^2 \log(nC))$.

5. (Bonus) In the previous problem set, one problem gave a $O(mn)$ time algorithm for finding a blocking flow. We can derive a faster algorithm assuming the existence of a special data structure called *dynamic trees*. The data structure maintains a vertex-disjoint set of rooted trees. Each vertex has a real-valued cost. The data structure can perform each of the following operations in $O(\log n)$ amortized time:

- *maketree*(i): Create a new tree containing the single vertex i of cost zero.
- *findroot*(i): Return the root of the tree containing vertex i.
- *findcost*(i): Return (j, x), where x is the minimum cost of a vertex on the tree path from i to `findroot`(i) and j is the last vertex on this path of cost x.
- *addcost*(i, x): Add x to the cost of every vertex on the path from i to `findroot`(i).
- *link*(i, j): Combine the two trees containing vertices i and j by adding the edge (i, j). i must be the root of a tree.
- *cut*(i): Divide the tree containing vertex i into two trees by deleting the edge out of i. i must not be the root of a tree.

Show that by using the dynamic trees data structure one can obtain an $O(m \log n)$ time algorithm for finding a blocking flow in an acyclic directed graph.