ORIE 633 Network Flows	October 25, 2007

Problem Set 4

Due Date: November 8, 2007

1. Consider the cost-scaling algorithm for the minimum-cost circulation problem that we discussed in class. We used a push/relabel subroutine for converting a 2ϵ-optimal circulation to an ϵ-optimal circulation; this subroutine's running time was dominated by the $O(n^2m)$ non-saturating pushes taken by the algorithm.

   Just as we improved the running time of push/relabel for the maximum flow problem from $O(n^2m)$ to $O(n^3)$ by carefully ordering the push and relabel operations (resulting in FIFO push/relabel), we can do the same thing in this case.

   (a) Show that in $O(m)$ time one can find an ordering of the nodes such that any push operation on an admissible arc will push from a node earlier in the ordering to one later in the ordering.

   The algorithm will consider nodes in the order given by the ordering. When considering node i, we continue to push excess from node i until either there is no longer any excess at node i (after a non-saturating push) or there are no admissible arcs out of node i.

   (b) Prove that if we must relabel i, we can move i to be the first node in the ordering and have the resulting ordering satisfy the properties of the ordering in part (a).

   After relabeling i, the algorithm moves i to the beginning of the ordering. It moves back to the beginning of the ordering (with node i) and considers nodes in the new order.

   (c) Argue that if we reach the end of the ordering without a relabel operation, then we have a feasible circulation and the subroutine terminates.

   (d) Argue that the number of non-saturating pushes is at most $O(n^3)$ and that the overall running time of the subroutine is now $O(n^3)$.

2. In class, we discussed a subroutine for converting a 2ϵ-optimal circulation to an ϵ-optimal circulation via a push/relabel algorithm, which resulted in an $O(n^2m\min(\log(nC), m\log n))$ time algorithm for the minimum-cost circulation problem. In this problem, we will give a subroutine for the same problem based on blocking flows. Consider the following subroutine. Below we let G_A be the graph of currently admissible arcs (that is, $c^p_{ij} < 0$).

find-ϵ-opt-circ(f, ϵ, p)

For all $(i,j) \in A$ if $c^p_{ij} < 0$ then $f_{ij} \leftarrow u_{ij}$
While f is not a circulation
    $S \leftarrow \{i \in V : \exists j \in V$ such that $e^f_j > 0, i$ reachable from j in $G_A\}$
    For all $i \in S, p_i \leftarrow p_i - \epsilon$
    Form network N from G_A by adding source s, sink t,
        arc (s,i) of capacity e^f_i for all $i \in V$ with $e^f_i > 0$,
        arc (i,t) of capacity e^f_i for all $i \in V$ with $e^f_i < 0$
    Find blocking flow b on N
    $f \leftarrow f + b$
Return(f,p)

Given a blocking flow algorithm that runs in $O(m \log n)$ time, prove that the subroutine above is correct and runs in $O(mn \log n)$ time. This gives a $O(mn \log n \min(\log(nC), m \log n))$ time algorithm for the minimum-cost circulation problem.

3. In this problem we will consider yet another algorithm for finding a minimum-cost circulation.

    (a) Let f^* be a minimum-cost circulation, and let f be any circulation. Prove that there exists some cycle C such that if circulation f' results from cancelling C, then
    $$c \cdot f - c \cdot f' \geq \frac{1}{m}(c \cdot f - c \cdot f^*).$$
    In other words, the improvement in cost that is obtained by cancelling C is at least a $\frac{1}{m}$ fraction of the difference in cost between our solution and the optimal.

    (b) Prove that $O(m \log(mUC))$ iterations of cancelling cycles as in part (a) is sufficient to find a minimum-cost flow.

Unfortunately, finding a cycle to cancel that gives the most improvement in the objective function is an NP-hard problem. However, we can do something almost as good by finding in polynomial-time a collection $\mathcal{C}$ of node-disjoint cycles such that cancelling all cycles in $\mathcal{C}$ improves the objective function as much as the most improving single cycle. The next step in this problem is to give an algorithm that will find such a collection $\mathcal{C}$.

To give such an algorithm, it may help to consider the *minimum-cost perfect matching problem in bipartite graphs*. In this problem, we are given as input a bipartite graph $G = (A, B, E)$ with $|A| = |B|$, with costs c_{ij} for each $(i,j) \in E$, where $i \in A$ and $j \in B$. The goal is to find a minimum-cost subset of edges $F \subseteq E$ such that each node in A and each node in B is adjacent to exactly one edge of F (a *perfect matching*). Of course, there may not be any $F \subseteq E$ that meets this condition, in which case the correct output is to say "no perfect matching". There is an $O(n(m + n \log n))$ time algorithm to solve this problem, where $n = |A| = |B|$, and $m = |E|$.

(c) (Bonus problem) Let f be any circulation, and let C be the cycle in G_f that results in the greatest improvement in objective by cancelling C; that is, if circulation f^C is the circulation that results by cancelling C, then C is chosen to maximize $\delta = c \cdot f - c \cdot f^C$. Prove that in $O(mn(m + n \log n))$ time one can find a collection $\mathcal{C}$ of node-disjoint cycles such that if f' results from cancelling all cycles in $\mathcal{C}$, then $c \cdot f - c \cdot f' \geq \delta$.

(d) Conclude that one can obtain a $O(m^2 n(m + n \log n) \log(mUC))$ time algorithm for the minimum-cost circulation problem.

4. (Bonus) This problem generalizes the minimum mean-cost cycle problem. Let $G = (V, A)$ be a directed graph, with integer costs c_{ij} and time $t_{ij} \geq 0$ for each $(i,j) \in A$. Assume that for every cycle Γ, $t(\Gamma) = \sum_{(i,j)\in\Gamma} t_{ij} > 0$. Let $T = \max_{(i,j)\in A} t_{ij}$ and $C = \max_{(i,j)\in A} c_{ij}$. Give a $O(mn \log(nCT))$ time algorithm for finding a cycle that minimizes
$$\min_{\text{cycles } \Gamma \in G} \frac{c(\Gamma)}{t(\Gamma)}.$$
Note that in the case $t_{ij} = 1$ for all $(i,j) \in A$, this is just the problem of finding a minimum mean-cost cycle.