ORIE 633 Network Flows September 20, 2007

Problem Set 2

Due Date: October 4, 2007

1. In discussing the push/relabel algorithm, we showed that if we can find a set of nodes S with $s \in S$ such that the current preflow f saturates all arcs in $\delta^+(S)$, and there is no active node in $V - S$, then S is a minimum s-t cut, and the maximum flow value will be $u(\delta^+(S))$. In practical implementations of the push/relabel algorithm, it is often useful to detect if the algorithm has reached this state, then run another algorithm to convert the current preflow f into a flow. One heuristic often used is the *gap heuristic*; it checks to see whether there is some distance k such that there are no nodes i with $d_i = k$, and all nodes j with distance $d_j < k$ are inactive.

   (a) Prove that the gap heuristic correctly detects if we have reached the state such that there is a set S with $s \in S$, all arcs in $\delta^+(S)$ saturated, and all nodes in $V - S$ inactive.

   (b) Given a preflow f in the state above, give an $O(mn)$ time algorithm for converting f into a flow f' of the same value. (Hint: although several algorithms will work, one idea is to use a variant of the shortest augmenting path algorithm from Problem Set 1).

2. Consider the problem of finding shortest paths in a directed graph:

   Shortest paths in directed graphs

   - **Input:**
     - A directed graph $G = (V, A)$
     - Lengths $l(i, j)$ for all $(i, j) \in A$, integer, possibly negative; however, no negative length cycles
     - Source vertex $s \in V$.
   - **Goal:** Find length $d(v)$ of shortest path from s to v for all $v \in V$.

   The Bellman-Ford shortest path algorithm computes $d(i)$ by computing $d_k(i)$, the shortest walk (i.e. a path in which vertices can be repeated) between s and i using exactly k arcs.

   (a) Prove that $d_k(j)$ can be computed by the recurrence

   $$d_k(j) = \min_{(i,j)\in A} [d_{k-1}(i) + l(i, j)].$$

   (b) Let $h_l(i) = \min_{k=1,\ldots,l} d_k(i)$. Prove that if the graph has no negative-length cycle then $h_{n-1}(i) = d(i)$ for all $i \in V$. Moreover, show that the graph has no negative-length cycle iff for all $i \in V$, $d_n(i) \geq h_{n-1}(i)$. (Hint for the if part: prove that if $h_n(i) = h_{n-1}(i)$ for all i then $h_c(i) = h_n(i)$ for all $c \geq n$.)

(c) Prove that this algorithm runs in $O(mn)$ time.

3. In undirected graphs it seems that there potentially could be $\binom{n}{2}$ distinct minimum s-t cut values, one for each possible pair of vertices s and t. In this problem, we will show that in fact there can be at most $n-1$ distinct cut values. Let $\lambda(G,i,j)$ denote the value of the minimum i-j cut in an undirected graph G.

   (a) Prove that for any sequence $i_1, i_2, \ldots, i_k$ of distinct nodes that
   $$\lambda(G, i_1, i_k) \geq \min(\lambda(G, i_1, i_2), \lambda(G, i_2, i_3), \ldots, \lambda(G, i_{k-1}, i_k)).$$

   Now consider an auxiliary graph G' that has the same node set V, and has edge set E' of all possible edges. Let the weight $w'_{ij} = \lambda(G,i,j)$ for all edges $(i,j) \in E'$. Let T be a maximum-weight spanning tree in G'.

   (b) For any distinct $i, j \in V$, let P_{ij} be the unique path in T between i and j. Prove that $\lambda(G,i,j) = \min_{(k,l)\in P_{ij}} w_{kl}$. (Hint: you may need the property of a maximum weight spanning tree that for any edge $(a,b) \notin T$, $w_{ab} \leq \min_{(c,d)\in P_{ab}} w_{cd}$.)

   From this we can conclude that there are at most $n-1$ distinct cut values, corresponding to the $n-1$ edges of T.

4. A flow f is called a *blocking flow* in G if for every s-t path P in G, there is some arc at capacity; that is, there is some $(i,j) \in P$ such that $f_{ij} = u_{ij}$. Note that a blocking flow is not necessarily a maximum flow, since there might exist an s-t path in the residual graph G_f.

   (a) Give an $O(mn)$ time algorithm for finding a blocking flow in an acyclic graph. (Partial credit for an $O(m^2)$ time algorithm).

   In some graphs, a blocking flow is also a maximum flow. This is true in *series-parallel* graphs. Series-parallel graphs can be constructed inductively. A graph with a single arc from s to t is the simplest series-parallel graph.

   Two series-parallel graphs G_1 and G_2 can be combined into a new series-parallel graph through either a *series composition* or a *parallel composition*. In a series composition, the t node of G_1 is identified with the s node of G_2, and the s node of G_1 becomes the s node of the new graph, while the t node of G_2 becomes the t node of the new graph. See Figure 1.

   In a parallel composition, the s nodes of G_1 and G_2 are identified, and the t nodes of G_1 and G_2 are identified. The identified s nodes are the s node of the new graph, and the identified t nodes are the t node of the new graph. See Figure 2.

   (b) Prove that a blocking flow in a series-parallel graph is also a maximum flow.

   (c) Show that series-parallel graphs are acyclic, and conclude that there is an $O(mn)$ time algorithm for finding a max flow in series-parallel graphs.

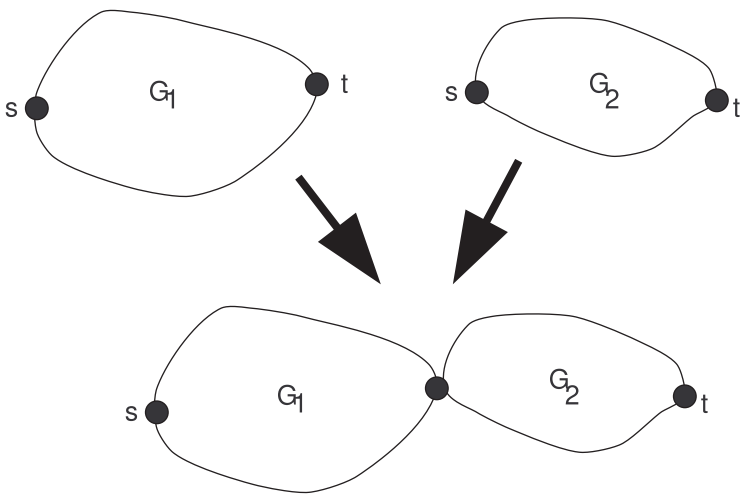

Figure 1: A series composition

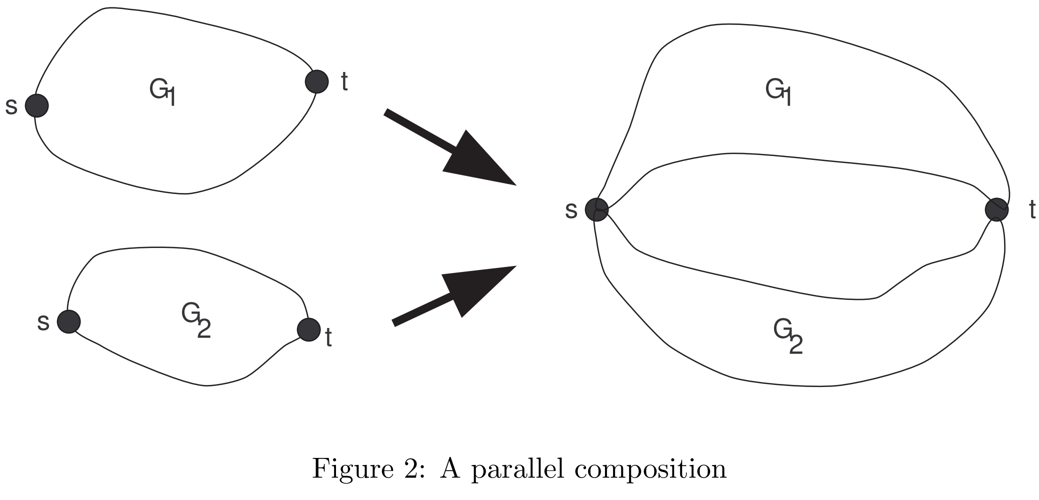

Figure 2: A parallel composition