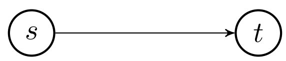
ORIE 6330 Network Flows	September 27, 2012

Problem Set 3

Due Date: October 18, 2012

1. Give an $O(mn)$ time algorithm to find a blocking flow in a directed graph $G = (V, A)$ with capacities $u(i, j)$ for $(i, j) \in A$ when there are no positive capacity cycles in the graph.

2. 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.

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

(b) Show that series-parallel graphs have no positive capacity cycles, and conclude that there is an $O(mn)$ time algorithm for finding a maximum flow in series-parallel graphs.

3. Recall that in class we showed how to find a minimum global cut in undirected graphs via MA orderings. Here we will see how to use MA orderings to compute a maximum s-t flow in a directed graph.

We compute an MA ordering in the following way. Let the source vertex s be the first vertex in the ordering; we set $v_1 = s$. In general, we choose the next vertex v_k in the ordering to maximize $u_f(\delta(W_{k-1}, v_k))$, where $W_{k-1} = \{v_1, \dots, v_{k-1}\}$, $v_k \notin W_{k-1}$, $\delta(W_{k-1}, v_k) = \{(i, v_k) \in A : i \in V_{k-1}\}$, and $u_f(X) = \sum_{(i,j)\in X} u_f(i, j)$ for $X \subseteq A$.

Suppose the sink vertex $t = v_\ell$. Then let $\alpha = \min_{k=2,\dots,\ell} u_f(\delta(W_{k-1}, v_k))$.

(a) Given the MA ordering, prove that one can augment the current flow f by α units of flow in $O(m)$ time.

(b) Show that the maximum flow in the residual graph is no more than $n\alpha$.

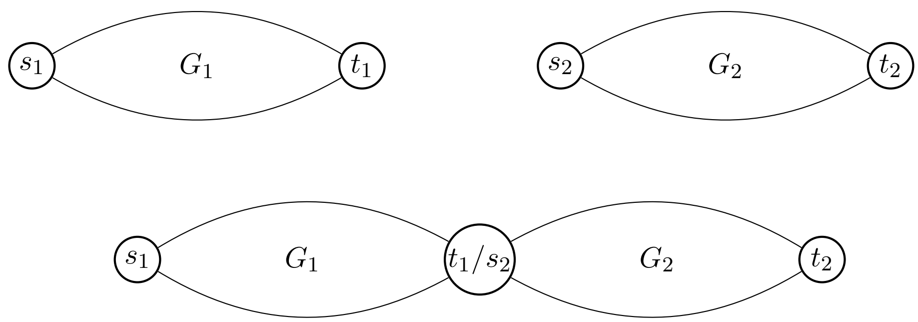


Figure 1: Series composition.

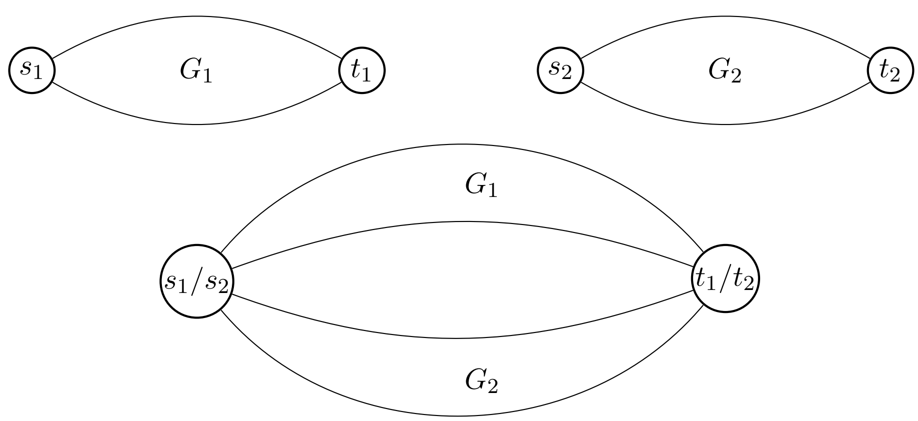


Figure 2: Parallel composition.

(c) Given an $O(m + n \log n)$ time algorithm for finding an MA ordering, use the items above to give an $O((m + n \log n)n \log(mU))$ time algorithm for finding a maximum s-t flow.

4. Let $G = (V, A)$ be a directed graph, with costs $c(i, j)$ for all $(i, j) \in A$.

(a) Using Problem 4 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, where $c(\Gamma) = \sum_{(i,j)\in\Gamma} c(i, j)$.

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

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

in $O(n^2)$ time.

5. (Bonus) In problem 1 above, 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.