Introduction to Algorithms **Solution Set 6**
CS 4820, Spring 2014

(1) *(5 points)* Design a linear-time algorithm that tests whether a given flow is a maximum flow. The input to the algorithm is a flow network G and a flow f in G; the output should be an answer to the question, "Is f a maximum flow?" You may assume that G has integer edge capacities and that f is an integer-valued flow. Your algorithm's running time should be $O(m+n)$, where m and n are the number of edges and vertices in G, respectively.

Solution

Build the residual graph G_f; this takes $O(m+n)$ time because G_f has n vertices and at most $2m$ edges, and the residual capacity of each edge can be computed in $O(1)$ time. Check whether the residual graph contains an augment path. This takes $O(m)$ time using BFS or DFS. If there is no augmenting path then f is a maximum flow: output "yes". Otherwise output "no". The correctness of the algorithm is due to the following two observations, both of which were proven in class.

- If f is a flow with no augmenting path, then f is a maximum flow. (This was the central claim in the proof of correctness of the Ford-Fulkerson algorithm.)
- If f is a flow whose residual graph contains an augmenting path P with bottleneck capacity b, then augmenting P yields a flow whose value is $\text{value}(f) + b$. In particular, the value of this flow is greater than the value of f, so f is not a max flow.

(2) *(10 points)* Design a linear-time algorithm that recomputes a maximum flow in a network after the capacity of one edge is decremented. The input to the algorithm consists of:

- a flow network G with integer edge capacities,
- a integer flow f that is a maximum flow in G,
- an edge $e = (u, v)$.

If G' denotes the flow network obtained from G by decreasing the capacity of edge e to $c(u, v) - 1$, then your algorithm should output a maximum flow in G', and it should run in time $O(m+n)$, where m and n are the number of edges and vertices in G, respectively.

Solution

If $f(u, v) < c(u, v)$, then output f itself. Otherwise, $f(u, v) = c(u, v)$. In this case, build the residual graph G_f and let G_f^+ denote the set of edges with strictly positive residual capacity. Test whether G_f^+ has a directed cycle containing edge (v, u). If so, perform the operation $\text{send}(a, b, 1)$ on every edge of this directed cycle. *(Translation for those of you who didn't attend the lectures on maximum flow: if (b, a) is an edge of the flow network G and $f(b, a) > 0$, then decrease the value $f(b, a)$ by 1. Otherwise, increase $f(a, b)$ by 1.)* In class we proved that f continues to satisfy the capacity and flow conservation constraints after this operation: it continues to satisfy the capacity constraints because we are only performing $\text{send}(a, b, 1)$ on edges whose residual capacity is at least 1. It continues to satisfy flow conservation because every vertex of the cycle is the head or exactly one edge on the cycle and the tail of another; thus it participates in two "send" operations, one of which increases its net outflow by 1 and the other decreases its net outflow by 1.

Finally, if $f(u,v) = c(u,v) > 0$ and G_f^+ does not contain a directed cycle through (v,u), then the following are true. **(Proofs omitted; to be filled in later.)**

1. The subgraph of the residual graph consisting of edges with positive residual capacity, G_f^+, contains directed paths from u to s and from t to v, but not from u to v.
2. If A denotes the set of vertices reachable from u in G_f^+, and B denotes the complement of A, then A, B constitute a minimum s-t cut.
3. The capacity of this cut decreases by 1 when we decrement the capacity of (u,v), so the max-flow value decreases by at least 1 as well.
4. Let P_u and P_v denote paths in G_f^+ from u to s and from t to v, respectively. These paths are vertex-disjoint. If we modify f by performing send$(a,b,1)$ for every edge $(a,b) \in P_u \cup P_v$ while reducing $f(u,v)$ to $f(u,v)-1$, then we obtain a max-flow in the new graph where $c(u,v)$ has been decreased to $c(u,v)-1$.

Proof of item 1. Since we are in the case that G_f^+ does not contain a directed cycle through (v,u), there is no directed path from v to u. By homework problem 5.2, we can write f as a nonegative combination of path and cycle vectors. One of these paths or cycles has to contains the edge (u,v) (because $f(u,v) > 0$). It cannot be a cycle because in this case the residual graph G_f^+ would contain a directed cycle containing the edge (v,u). Therefore there exists a directed path from s to t through (u,v) with positive flow along all of its edges. The reversal of this path is a directed path from t to s through (v,u) in G_f^+ (because all back edges have positive residual capacity). It follows that G_f^+ contains both a directed path from t to v and a directed path from u to s.

Proof of item 2. By item 1, the set A contains s but not t. Hence, (A,B) is a valid s-t cut. To show that it has minimum capacity, it's enough to show that its capacity is equal to the value of the flow. Since A contains *all* vertices reachable from u in G_f^+, every edge (p,q) that goes out of A has residual capacity 0, which means $f(p,q) = c(p,q)$, and every edge (p,q) that goes into A has residual capacity $c(p,q)$, which means $f(p,q) = 0$. It follows that

$$\text{value}(f) = \underbrace{f^{out}(A)}_{=c(A,B)} - \underbrace{f^{in}(A)}_{=0} = c(A,B).$$

Proof of item 3. The capacity of any s-t cut is an upper bound on the maximum value of a s-t flow.

Proof of item 4. The paths are vertex disjoint because otherwise the graph G_f^+ would contain a directed cycle through (v,u). Since we perform the send operation along a directed simple path from t to s through (v,u), flow conservation and capacity constraints will be maintained, the value of the flow decreases by 1, and the flow through (u,v) decreases by 1. Hence, the resulting flow satisfies the capacity constraints even after we decrement the capacity of the edge (u,v). By item 3, the resulting flow is a maximum flow in the new flow network G'.

(3) *(5 points) In this problem — as in the lectures this semester, but contrary to the textbook — a flow network may have edges entering the source and/or leaving the sink.*

A flow network is called *Eulerian* if the combined capacity of the incoming edges at any vertex is equal to the combined capacity of the outgoing edges at that vertex. If a and b are any two distinct vertices of an Eulerian flow network, prove that the maximum flow from a to b is equal to the maximum flow from b to a.

Solution

For any two vertex sets X, Y, let

$$c(X,Y) = \sum_{x \in X} \sum_{y \in Y} c(x,y)$$

denote the total capacity of edges from X to Y. (Note that $c(X,Y)$ is defined by the above formula even when X and Y are not disjoint.) By abuse of notation, for a vertex v we'll use $c(v,Y)$ to denote $c(\{v\},Y)$ and $c(X,v)$ to denote $c(X,\{v\})$.

The Eulerian condition can thus be expressed by the property

$$\forall u \in V \; c(V,u) = c(u,V)$$

which we will use as follows. Consider any two vertex sets A, B such that $B = V - A$, i.e. A and B form a partition of V. Then we have

$$\begin{aligned} c(A,B) - c(B,A) &= c(A,B) + c(A,A) - c(A,A) - c(B,A) \\ &= c(A,V) - c(V,A) \\ &= \sum_{u \in A} c(u,V) - c(V,u) = 0, \end{aligned}$$

where the last step used the Eulerian property.

Thus, for any sets A, B such that $B = V - A$, we have $c(A,B) = c(B,A)$. It follows that the minimum capacity of an a-b cut equals the minimum capacity of a b-a cut. By the max-flow min-cut theorem, the same equality holds for the maximum flow values from a to b and from b to a.

(4) *(10 points)* At Ford-Fulkerson University (motto: "I would find a network whose maximum flow value is different from its minimum cut capacity, but there is no such network.") there are many committees that need to be staffed with professors. The university has p professors organized into d departments; each professor belongs to only one department. There are q committees, and the following constraints must be satisfied when staffing the committees.

1. The required number of professors on committee k is specified by a positive integer r_k.
2. No professor is allowed to serve on more than c committees.
3. No committee is allowed to have more than one professor from the same department.
4. For each professor j, there is a list L_j of the committees on which he or she is qualified to serve. Professor j is not allowed to serve on committee k unless $k \in L_j$.

Design a polynomial-time algorithm to determine whether it is possible to staff each committee without violating any of the constraints listed above. If it is possible to staff the committees, your algorithm should output an assignment of professors to committees that satisfies all of the constraints. The input to the problem is specified by the numbers $c, d, p, q, r_1, \ldots, r_q$, and the lists $L_1, \ldots, L_p$.

Solution

Construct a flow network G with the following vertices.

- source s, sink t
- professor nodes u_j for $j = 1, \ldots, p$.
- middle nodes v_{ki} for all $k = 1, \ldots, q$ and $i = 1, \ldots, d$.
- committee nodes w_k for $k = 1, \ldots, q$.

Our flow network G has the following edges.

- Edges (s, u_j) of capacity c for $j = 1, \ldots, p$.
- Edges (u_j, v_{ki}) of capacity 1 for all k, i such that professor j belongs to department i and $k \in L_j$.
- Edges (v_{ki}, w_k) of capacity 1 for all j, k.
- Edges (w_k, t) of capacity r_k for $k = 1, \ldots, q$.

Compute an integer-valued maximum flow in G, denoted by f. If there exists an edge (w_k, t) such that $f(w_k, t) < r_k$ for any k, report the it is impossible to staff the committees. Otherwise, assign professors to committees as follows. For every edge (u_j, v_{ki}) such that $f(u_j, v_{ki}) = 1$, assign professor j to committee k.

Running time. The flow network has $O(p + qd)$ vertices and $O(pq)$ edges. The bound on the number of vertices is obvious; the bound on the number of edges follows by counting edges in each layer, and the only non-obvious step is to notice that since each professor belongs to at most one department, the number of edges leaving a given u_j is at most q, which bounds the number of edges in that layer of the graph by $O(pq)$. (Another observation that contributes to the $O(pq)$ bound on the number of edges is that $p \geq d$ because each professor belongs to exactly one department.) Constructing the network takes $O(1)$ per vertex and edge, and finding a maximum flow using Ford-Fulkerson takes $O(mV)$ where m is the number of edges and V is the maximum flow value. In this problem, the maximum flow value is bounded by pq since there are at most pq edges from the "u layer" to the "v layer" and each has capacity 1, so the cut which has $\{s, u_1, \ldots, u_p\}$ on one side and all other vertices on the opposite side has capacity pq. Substituting this upper bound for the max-flow value into the Ford-Fulkerson running time bound, and recalling that the number of edges is $O(pq)$, we find that our algorithm's running time is bounded by $O(p^2q^2)$.

(Remark: Other running time analyses are acceptable. For example, if using the preflow-push algorithm from Section 7.4 of the textbook whose worst-case running time is $O(n^3)$, one would obtain a running time bound of $O((p + qd)^3)$.)

Correctness. We will show that for every valid committee assignment, there exists a flow with value $\sum_k r_k$, and that for every integer-valued flow with at least this value, the last step of the algorithm reconstructs a valid committee assignment.

For any valid committee assignment, we consider the following flow f. Choose $f(u_j, v_{ki}) = 1$ if professor j in department i serves on committee k. Choose $f(s, u_j) = c_j$ if professor j serves in c_j committees. Choose $f(v_{ki}, w_k)$ if some professor from department i serves on committee k. Choose $f(w_k, t)$ to be the number of professors serving on committee k (this number is equal to r_k in any valid committee assignment). Finally, choose f to be 0 on every other edge. By construction, the flow f satisfies all flow conservation and capacity constraints. It's value is equal to the net amount of flow going into t, which is equal to $\sum_k r_k$.

Now, let f be any integer-valued flow with value $\sum_k r_k$. The flow entering t is equal to this value. Since the edges (w_k, t) have capacities r_k, each of these edges are saturated by the flow. (Therefore, the flow passes the checks of the algorithm.) Consider the committee assignment defined by the last step of the algorithm: assign professor j in department i to committee k if and only if $f(u_j, v_{ki}) = 1$. By the construction of the flow network, professor j can be assigned to committee k only if $k \in L_j$ (constraint 4). Since the flow is integer valued, we have $f(u_j, v_{ki}) = 0$ if professor j in department i is not assigned to committee k. By flow conservation for vertex v_{ki}, the flow on (v_{ki}, w_k) is equal to the number of professors from department i assigned to committee k. Since the capacity of these edges is equal to 1, it follows that at most 1 professor from department i is assigned to committee k (constraint 3). By flow conservation for vertex w_k, the flow on (w_k, t) is equal to the number of professors assigned to committee k. We established before that this flow is equal to r_k (constraint 1). It remains to verify that no professor serves on more than c committees (constraint 3). By flow conservation for vertex u_j, the flow on (s, u_j) is equal to the number of committees that professor j is assigned to. Since the capacity of this edge is c, it follows that no professor is assigned to more than c committees.