

CS 222 Fall 2007 HW 3, Due Oct. 26.

In our formulation of maximum flow, we only have upper capacities C_e on the flow in each edge e . However, it is possible to put a lower capacity L_e on each edge e , requiring that $f(e)$ satisfy the constraint that $L_e \leq f(e) \leq C_e$. The network-flow problem can be modified to determine if such a flow is possible, and to find a maximum flow if so (this is detailed in the book). For the next problem, you do not need to know the details of that modification, but you can assume that such a modification is possible, and use it in your answers, if that is helpful.

1. Table Rounding problem: You are given an n by m table of numbers between 0 and 1 along with row and column totals. Your objective is to round each entry to either 0 or 1 (you have complete freedom in this) so that each resulting row and column total is itself rounded to one of its two nearest integers, and so that the table total is rounded to one of its two nearest integers. Any number which was originally an integer should not change. For example,

.2	.4	0	.2		0.8		0	0	0	0	0
.4	0	.2	.4		1		0	0	1	0	1
.8	.8	.8	0		2.4	====>	1	1	1	0	3
.6	0	.2	.2		1		1	0	0	0	1

2	1.2	1.2	.8		5.2		2	1	2	0	5

It turns out that such a rounding is always possible, but you don't have to prove that.

Devise an algorithm, based on network flow, that takes in a table (as described above) and returns a rounding of it (as described above), or declares that no rounding is possible (that outcome should not actually ever happen). The solution must be general, not just be described for the example above. You might be able to prove that a rounding is always possible, by making some observations about your algorithm. That is not required, but you might try it out.

Answer: Build a bipartite graph with node sets A and B, with one node in A for each row and one node in B for each column. Direct an edge from

each node in A to each node in B , and put a capacity of 1 on each of these edges. Add a source node s and a directed edge to each node in A , and add a node v and a directed edge from each node in B to v , and add a terminal node t and a directed edge from v to t . For each node i in A , let $r(i)$ denote the given row total for row i . Then set the lower bound on edge (s, i) equal to $r(i)$ rounded down to the nearest integer of $r(i)$, and set the capacity of (s, i) equal to $r(i)$ rounded up to its nearest integer. Similarly, for each node j in B , let $c(j)$ denote the given column total for column j . Then set the lower bound on edge (j, v) equal to $c(j)$ rounded down to the nearest integer of $c(j)$, and set the capacity of (j, v) equal to $c(j)$ rounded up to its nearest integer. Note that if any $r(i)$ or $c(j)$ is an integer, then both lower and upper bounds will be that integer. Let T denote the given total for the table. Then set the lower bound on edge (v, t) equal to T rounded down to the nearest integer of T , and set the capacity of (v, t) equal to T rounded up to its nearest integer. Again, if T is an integer, then both bounds will be T . Then compute an integral maximum flow (we know that if there is a max. flow, then there is an integral one). If there is a max flow, then all the constraints have been satisfied, and we use the flow values on the various edges as values in the rounded table.

2. Suppose we have the height and weight of every person in the class, and we organize this information into a two dimensional table T where there are n height intervals, and m weight intervals. In the table, the (i, j) entry $T(i, j)$ indicates how many people fall into height category i and weight category j . Suppose that some of the particular cell entries are kept secret (so they are empty), but the number of people falling into each height category is public, as is the number that fall into each weight category. That is, the n row sums and the m column sums of the table are published, along with some, but not all, of the individual cell values of the table.

Devise an algorithm, based on network flow, that finds a way to assign values to the empty cells so that every row now adds up to its given row total, and every column now adds up to its given column total.

Answer: Similar to problem 1. Set up a bipartite graph (A, B) with one node in A for each row and one node in B for each column, and an edge (i, j) between the node for row i and the node for column j , if cell (i, j) is empty. Put infinite capacity on each such edge. Now add a source node s with a directed edge to each node i in A , with capacity equal to the given row total

for i . Add a terminal node t with a directed edge to it from each node j in B , with capacity equal to the column total for j . An s-t maximum flow with value equal to the sum of the row totals, describes valid values to use for the missing cells.

3. The node cover problem

Let G be an undirected graph with each node i given weight $w(i) > 0$. A set of nodes S is a *node cover* of G if every edge of G is incident to at least one node of S . The *weight* of a node cover S is the summation of the weights, denoted $w(S)$, of the nodes in S ; the weighted node cover problem is to select a node cover with minimum weight.

If G is a bipartite, then the minimum weight node cover can be found by network flow. Explain how to do this. Hint: The minimum cut is the key, rather than the maximum flow.

Answer: Add a node s and an edge from s to each node i on the left side of G . Give that edge capacity $w(i)$. Similarly, add a node t and an edge from each node j on the right side to t . Give that edge capacity $w(j)$. Direct each original edge in G from left to right and give each such edge infinite capacity. Then find a minimum cut in that edge. If the edge from s to i crosses the cut, then take node i into the node cover. If the edge from j to t crosses the cut, then take node j into the node cover. This set of nodes must be a node cover because for any edge (i, j) in the bipartite graph, at least one of the edges (s, i) or (j, t) must cross the cut, or else t would be on the s side of the cut, which is not possible. Now the weight of the node cover defined in this way equals the capacity of the minimum cut, so the minimum node cover has total weight less than or equal to the minimum cut capacity. Conversely, if we have a node cover, cut the exterior edges corresponding to the nodes in the cover, creating a cut of capacity equal to the weight of the node cover. Hence minimum cut capacity must be less than or equal to the minimum node cover weight. Putting the two sides together we have that the minimum cut capacity must equal the minimum node cover weight, hence the minimum cut defines a minimum weight node cover.

4. Let f and f' be two different maximum s-t network flows in a graph G , where the total flow into any node v is the same in f as in f' (this may be the case for example in your solution to problem 2). Let $e = (u, v)$ be a directed edge where $f'(e) > f(e)$. Show that in the residual graph G_f for flow f , there must be a directed cycle that includes the forward edge (u, v) .

Similarly, in the residual graph $G_{f'}$ for f' , there must be a directed cycle that includes the backward edge (v, u) .

Answer: For each edge $g = (p, q)$, where $f'(g) \neq f(g)$, define $F(p, q) = f'(g) - f(g)$, which could be positive or negative. Note that for any node $q \neq s$ (even t), $\sum_p F(p, q) = 0$, since the total flow into q is the same in f and f' . Similarly for any node $p \neq t$ (even s), $\sum_q F(p, q) = 0$, since the total flow out of p is the same in f and f' . Also, note that if $F(p, q) > 0$ then (p, q) is a forward edge in G_f since $f(p, q)$ cannot be at its capacity. In fact, the residual capacity of edge (p, q) in G_f is at least $F(p, q)$. Similarly, if $F(p, q) < 0$ then $f(p, q) > 0$, so edge (q, p) is a backward edge in G_f with residual capacity $f(p, q) \geq f(p, q) - f'(p, q) = |F(p, q)|$. So in G_f , if $F(p, q) > 0$ then assign $F(p, q)$ to the forward edge (p, q) ; if $F(p, q) < 0$, assign $|F(p, q)|$ to the backward edge (q, p) . But don't assign anything to edge (u, v) . I claim that the result is a flow from v to u of total value $F(u, v)$. To see this, note that the assigned values obey the capacity constraints on the edges in G_f , and for every node other than v or u , the total of the assigned values into the node equals the total of the assigned values out of the node. This is even true for nodes s and t . The total out of v is greater than the total in by exactly $F(u, v)$ and the total in to u is greater than the total out of u by exactly $F(u, v)$. Thus this assignment satisfies the definition of a flow from v to u , of value $F(u, v)$. Now since $F(u, v) > 0$, there certainly must be a directed path in G_f from v to u , and when we include the edge $e = (u, v)$ (which exists in G_f since $f(u, v) < f'(u, v) \leq c_e$), we have shown that e is in a directed cycle in G_f . Note that this flow in G_f does not put any flow on the backward edge (v, u) , so in fact there is a directed cycle in G_f that contains e and that has at least three nodes.

Now suppose we want to know if there is another s-t flow f'' in G where the total flow into any node is the same as in f , and where $f''(e) > f'(e)$, assuming that $f'(e) < C_e$. We claim we can solve this question with the following method: Let $G_{f'}$ be the residual graph for flow f' . Then remove the edge (v, u) from $G_{f'}$ and find the maximum flow value in that $G_{f'}$, from v to u . (That is, let v be the source node, and u be the terminal node in that flow computation.)

Problem: How does the flow value computed in the above method answer the question of whether such an f'' exists? That is, complete the description of the above method, and explain why the method is correct.

Answer: To finish the algorithm find a maximum flow from v to u in $G_{f'}$. There is the claimed flow f'' if and only if the maximum v to u flow has non-zero value.

The proof of this is the follows. By the answer to the first part of problem 4, if there is such an f'' , there is a directed path from v to u in $G_{f'}$, so there is some non-zero $v - u$ flow in $G_{f'}$.

Conversely, if the maximum flow value is greater than 0, then let b be the minimum non-zero flow on any edge in the flow. Hence there must be a single path from v to u in $G_{f'}$ with minimum non-zero capacity of b . As in the Ford-Fulkerson algorithm, we can increase the flow on any forward edge in that path by b , and decrease the flow by b on any edge (i, j) where (j, i) is a backward edge on the path. Then if we increase the flow on (u, v) by b , we will have another $s - t$ maximum flow, but where the flow on e is larger than in f' . So if there is a $v - u$ flow in $G_{f'}$, there is a flow f'' as described above.