Network Flows Take Home Final

You have 24 hours to do this exam. You are responsible for turning it in to me (Math Tower 1-106) within 24 hours of receiving it. Plan to spend 10-15 minutes discussing your exam with me, when you turn it in. Do ANY four out of the following 6 problems. All problems are weighted equally, 25 out of 100 points. On the front page write clearly which four problems you are turning in. I will NOT grade more than four problems, and then choose the best.

You are permitted to use the texts (Ahuja, Magnanti and Orlin or West), your notes and any material handed out in class. You do not need to reprove any theorem or algorithm we had covered, just state clearly what you are using. No other texts or papers are allowed. Feel free to ignore any “helpful hints”! You are expected to work on your own, without discussing the problems with others until everyone has finished taking the exam. You may consult with me, if you have trouble understanding a question. You can reach me by email: estie@ams.sunysb.edu or phone: 632-8363 office, and 474-4301 home.

Please write up your answers neatly and carefully. If I cannot understand what you are doing, I will not assign partial credit.

1. Since Dijkstra’s algorithm for finding a shortest path from node 1 to node n in a digraph with nonnegative arc lengths is very efficient, one would like to transform a digraph with possibly some negative lengths into an equivalent one in which all lengths are non negative. An equivalent digraph is one in which the shortest path from node 1 to n is the same as in the original digraph. Note that the length of the shortest path in the equivalent network may not be equal to the length of that path in the original digraph. The following algorithm accomplishes such a transformation for a digraph with n nodes:
   Step 0. t := 1 (t counts the number of iterations)
   Step 1. i := 1
   Step 2. $\tau_i := \min_j c_{i,j}$. If $\tau_i < 0$ replace $c_{i,j}$ by $c_{i,j} - \tau_i$ for each $j$, and replace $c_{k,i}$ by $c_{k,i} + \tau_i$ for each $k$.
   Step 3. If $i < n$ then increase $i$ by one and GOTO step 2.
   Step 4. If $c_{i,j} \geq 0$ for all $i, j$, STOP. (The desired equivalent network has been found.) Otherwise:
   If $t < n + 1$, increase $t$ by one and GOTO step 1. If $t = n + 1$, STOP. (The original network contained a negative cycle.)
   (a). Prove a shortest path in the original digraph is also a shortest path in the equivalent digraph.
   (Hint: consider a particular path from 1 to n and see how the lengths of its arcs change.)
   (b). Prove that if the graph has a negative cycle, that at every iteration (including after $n + 1$ iterations) there is an arc $(i, j)$ with negative cost $c_{i,j} < 0$. Note, at each iteration it may be a different arc! Another note: One can also show that if the digraph has no negative cycles, then after $n + 1$ iterations all $c_{ij} \geq 0$, but you are not asked to show this.

2. The $b$-matching problem is the following: Given a graph $G = (V, E)$ and for each node $v \in V$ a number $b_v > 0$, is there a subset $M \subseteq E$ such that for each $v \in V$, $v$ is incident upon $b_v$ edges in $M$. For example, a perfect matching is a $b$ matching where all $b_v = 1$.
   (a). Let $G$ be a bipartite graph. Give an algorithm to solve the $b$-matching on $G$. You can either describe a new algorithm, modify an algorithm from class, or rephrase the problem as some other problem we solved during the semester. Make sure to state the running time of your algorithm.
   (b). Show that the Generalized Factor Problem is NP-complete: Given a graph $G$, let $B_v$ be a subset of $\{0, 1, \ldots, \deg(v)\}$ for each node $v \in V$. ($\deg(v)$ is the degree of node $v$.) Does there exist
a subset $M \subseteq E$ such that the number of edges in $M$ incident to node $v$ is an element of $B_v$? (For example, the perfect matching problem has $B_v = \{1\}$, and $b$-matching has $B_v = \{b_v\}$.) Hint: This can be shown for bipartite graphs using a reduction from 3SAT.

3. Consider the feasible path with time windows problem. Given a directed graph with two specified nodes $a, b \in V$, travel times on the arcs $t_{ij}$ for arcs $(i, j)$, and time windows $[s_i, e_i]$ in which we are allowed to be at node $i$. Assume all the data is integer. We wish to know if there exists a path from node $a$ to node $b$ such that all nodes that the path visits, are visited during their time window. Note: You are not allowed to wait anywhere during your travel along the path, nor are you allowed to repeat nodes.

(a). Suggest an algorithm to solve this problem. One approach is to define multiple copies of each node $i$ to be $i, s_i, i, s_i + 1, ..., i, e_i$. Make sure to also state and explain the running time of your algorithm in terms of $n$ the number of nodes, $m$ the number of edges, and $W = \max_i e_i - s_i + 1$.

(b). Show that this problem is NP-complete. You may wish to use a reduction from partition, similar to the one we used to show that bicriteria shortest path problems are NP-complete.

4. Let $G = (V, E)$ be an undirected graph with weights on the edges, $w(e) > 0$. Let the weights be unique, i.e., if $e \neq f$ then $w(e) \neq w(f)$.

(a). Show that the minimum spanning tree is unique (i.e., there is only one minimum spanning tree).

(b). Given nodes $s, t \in V$ is the shortest path between $s$ and $t$ unique? Prove or give a counterexample.

5. The rural postman problem is: Given a graph $G = (V, E)$ with non-negative costs on the edges satisfying the triangle inequality, and a subset of the edges $E' \subset E$. The postman must find a cycle including all edges of $E'$ at minimum total cost. The problem is NP-complete (you don’t have to show this).

Describe an approximation algorithm for the rural postman. You should clearly describe an algorithm that runs in polynomial time, then prove that the cycle you obtain has length at most some constant times the length of the optimal cycle, for all instances of the problem.

6. The interval scheduling problem is as follows: You are given a list of $n$ tasks, each with a start time $s_i$ and end time $e_i$, where $e_i > s_i$. There are $k$ machines on which to schedule these tasks, where a machine can do at most one task at a time, and once a task is started it cannot be interrupted. It may help to think of the tasks as courses that meet once a week, and the machines as the classrooms, in which the meeting takes place. The goal is to schedule tasks (classes) to maximize the number of scheduled tasks. This is a similar problem to colouring interval graphs, where you wish to colour all the nodes (intervals) with the minimum number of colours. The interval scheduling problem is equivalent to the problem of colouring the largest number of nodes with a fixed number of colours, $k$.

(a). Suggest a method for solving the interval scheduling problem. You can either describe a new algorithm, modify an algorithm from class, or rephrase the problem as some other problem we solved during the semester. (Think of min cost flow. You may use negative costs, if this helps your formulation.)

(b). Show that the problem becomes NP-complete if we add a restriction, in which each task has a list of acceptable machines on which it can be scheduled, and we are not allowed to schedule a task (course) to an unacceptable machine (room).