University of Washington
Department of Computer Science and Engineering
CSE 521, Spring 1998

May 13, 1998

Applications of network flow, **May 13, 1998**

This material is not covered in our text. Some of it is from the book *Network Flows*, by Ahuja, Magnanti, and Orlin.

Problem 1. Maximum Independent Set in a Bipartite Graph:

Given a bipartite graph $G = (U, V, E)$, find an independent set $U' \cup V'$ which is as large as possible, where $U' \subseteq U$ and $V' \subseteq V$. A set is independent if there are no edges of E between elements of the set.

Solution:

Construct a flow graph on the vertices $U \cup V \cup \{s, t\}$. For each edge $(u, v) \in E$ there is an infinite capacity edge from u to v. For each $u \in U$, there is a unit capacity edge from s to u, and for each $v \in V$, there is a unit capacity edge from v to t.

Find a *finite* capacity cut S, T, with $s \in S$ and $t \in T$. Let $U' = U \cap S$ and $V' = V \cap T$. The set $U' \cup V'$ is independent since we are assuming that there are no infinite capacity edges crossing the cut. The size of the cut is $|U - U'| + |V - V'| = |U| + |V| - |U' \cup V'|$. Thus, in order to make the independent set as large as possible, we make the cut as small as possible.

Problem 2. Optimal Strip Mining:

The strip mining problem is to determine the material to remove from the mine which maximizes the value of the ore less the cost of the removal of the material. The area to be mined is divided into cubes of a fixed size, and we must decide which cubes are to be removed. Each cube has a cost of removal as well as a value of the ore that it contains. If a cube is removed, the cubes above it, and diagonally above it must also be removed. (The latter restriction is to ensure that the sides of the mine are not too steep.)

This problem can now be modelled with a graph, $G = (V, E)$. We have a directed acyclic graph, where the nodes correspond to the cubes of earth to be removed. A cube can only be removed if all of its predecessors have been removed. Cube v_i has cost c_i and benefit b_i. We can assume that $c_i \geq 0$, $b_i \geq 0$ and one of c_i or b_i is 0. We wish to find a set of vertices R which is closed with respect to predecessors (i.e., with the property that if $(u, v) \in E$ and $v \in R$ then $u \in R$) and maximizes $\sum_{v_i \in R} b_i - c_i$.

Solution:

We begin by constructing a flow graph on $V \cup \{s, t\}$. For $(u, v) \in E$ we have an infinite capacity edge between u and v. If $c_i > 0$ we put an edge from s to v_i with capacity c_i, and if $b_i > 0$ we put an edge from v_i to t with capacity b_i.

Find a *finite* capacity cut S, T, with $s \in S$ and $t \in T$. We will take our solution R to be $T - \{t\}$. Since there are no infinite capacity edges from S to T, if u is a predecessor of v and $v \in R$ then $u \in R$, so R is closed. The edges crossing the cut are edges from the source, or edges to the sink. A source edge crossing the cut gives a cost c_i for an element in R. A sink edge crossing the cut gives a benefit b_i for an element not in R. Let the maximum possible benefit B be $B = \sum_i b_i$ and the profit P_R for a solution R be $P_R = \sum_{v_i \in R} b_i - c_i$. The capacity of the cut is:

$$\sum_{v_i \in R} c_i + \sum_{v_i \in V-R} b_i = \sum_{v_i \in R} c_i + \sum_{v_i \in V-R} b_i + \sum_{v_i \in R} b_i - \sum_{v_i \in R} b_i = B + \sum_{v_i \in R} c_i - \sum_{v_i \in R} b_i = B - P_R.$$

Thus, to maximize the profit, we minimize the size of the cut. (Note that there are trivial solutions which guarantee that there is a finite sized cut, which correspond to either having $S = \{s\}$ or $T = \{t\}$.)

Problem 3. Rounding elements of a table:

Give ranges for elements of a table and the row sums and columns sums, the question is whether or not there is an assignment of values to the ranges that satisfy the row and column sums. To be more precises, we have two $n \times m$ arrays, L and U which specify lower and upperbounds on matrix elements. We also have a vector R of length n which specifies the column sums and a vector C of length m which specifies the column sums.

We want to find a matrix A such that $l_{ij} \leq a_{ij} \leq u_{ij}$ and $\sum_j a_{ij} = r_i$ and $\sum_i a_{ij} = j$.

Solution:

We construct a flow graph which sends flows from "rows" to "columns" via the matrix entries, such that there is a feasible flow iff the equations can be satisfied.

We will need to assume that the *vertices* of the graph have capacities with upper and lower bounds on the flow. The trick to introduce vertex capacities is to split the vertex v into vertices v_{in}, v_{out}, with an edge $v_{in} \to v_{out}$ with the given upper and lower bounds on the vertex capacity. All edges originally going into v go to v_{in} and all edges originally leaving v, leave from v_{out}.

The flow graph has vertices ${v^r}_i$ (for the rows), ${v^c}_j$ (for the columns), and v_{ij} (for the array entries). The capacities are:

Vertex	Lower bound	Upper bound
${v^r}_i$	r_i	r_i
${v^c}_j$	c_j	c_j
v_{ij}	l_{ij}	u_{ij}

We have edges ${v^r}_i \to v_{ij}$ for all j and edges ${v^c}_j \to v_{ij}$ for all i. These edges can have infinite capacity (since we already restrict the flow with capacities on the vertices).

If there is a feasible solution, then each row sends its flow to the entries in its column, and then the entries of each column send their flow to the column. This gives the correct values for the row and column sums.

Problem 4. Can the Mariners win the pennant?:

The final problem is to determine, given the current standings in a sports league, whether or not a given team has the possibility of finishing first.

At some point in the season we have the current standings, e.g:

Team	Wins	Losses
Army Ants	8	4
Banana Slugs	7	5
Cockroaches	6	6
Dead Snails	3	9

as well as the number of remaining games for each team. Let g_{ij} denote the number of games that team i has with team j. In this example, lets assume each team has two remaining games with each other team.

We ask the question, can the Dead Snails finish first in the league. Since they have six games left, they can finish with nine wins which would put them in first place, assuming that the other teams lost all of their remaining games. However, since the other teams will play each other, we can't assume they will all lose all games.

Solution:

The flow problem we set up has vertices t_i for each team (other than the team we are evaluating), and vertices for the v_{ij} for the games that team i plays with team j.

Let W be the maximum number of wins that the Dead Snails can have assuming they win the rest of their games, and w_i the number of wins that team i currently has. We have an edge of capacity $W - w_i$ from s to t_i. We have infinite capacity arcs from t_i to v_{ij} for each j. The edge from v_{ij} to t has capacity g_{ij}. We ask if there exists a flow of size $\sum_{ij} g_{ij}$ in this network.

The flow corresponds to having a winner for each of the remaining games. The flow along the edges $v_{ij} \to t$ accounts for teams t_i and t_j playing the right number of times. The flow from s to the t_i's assigns winners to the games in a way that no one has more wins than the Dead Snails.