College of Computer & Information Science — Spring 2011
Northeastern University — Handout 26
CS5800: Algorithms — 21 April 2011

Sample Solution to Problem Set 6

1. (10 points) Selecting online advertisements

Major online portals like Google and Yahoo have considerable information about the individual users based on their past interactions. This allows them to post targeted advertisements to the users. Suppose a set U of n users, labeled 1 through n, visit the portal on a particular day. The portal has a set A of m ads, labeled 1 through m, to choose from. The analysis of the users has revealed k different groups (from a marketing standpoint), the ith group consisting of subset S_i of users from U. A user may be part of several groups; i.e., a user may be an element of several different S_i's. The jth ad is targeted to a subset $G_j \subseteq \{1, \ldots, k\}$ of the groups.

The portal needs to decide whether there exists a way of assigning advertisements to users such that the following conditions hold: (a) each user is shown exactly one ad; (b) ad j is shown to user i only if i is in a group k in G_j; (c) the number of times the ad j is shown is at least r_j, where r_j is a given integer.

Give a polynomial-time algorithm that takes the above input – U, A, the sets S_i's, the groups G_j's, the r_j's – and determines whether the portal can assign an ad to each user so that the above three conditions are satisfied, and if so, then returns such an assignment.

Answer: We set up a network flow problem as follows. We have a node for each user, for each ad, for each group, and a source node s, and sink node t. Thus, the set of nodes is $U \cup A \cup \{S_i : 1 \leq i \leq k\} \cup \{s, t\}$. The edges and their capacities are as follows. There is an edge from s to every user u in U, with capacity 1. There is an edge from user u to group S_i if $u \in S_i$ with capacity ∞. There is an edge from group S_i to ad j if j can be shown to users in S_i; i.e., if $i \in G_j$ with capacity ∞. Finally, there is an edge from ad j to t with capacity r_j.

We compute maximum flow and determine whether value of flow equals $\sum_j r_j$. If it is, then the flow gives an assigment of ads to users satisfying all the constraints, except for possibly not covering all users. We can then assign ads to remaining users arbitrarily, while ensuring the constraint that user is in one of the appropriate groups.

We now consider the flow in the other direction. Suppose there is a way to assign the ads so that all constraints are satisfied. Consider the flow associated with this assignment. The only capacity constraints that could possibly be violated are the ones on the edges joining ads to the sink. Consider one such violation: suppose edge from ad j to t has a flow of $f_j > r_j$ on it; we compute $f_j - r_j$ flows, each of value 1 from the users to ad j to t, and remove them. Once we have addressed all violations, we have a max flow of value $\sum_j r_j$ in the original network.

Clearly, the algorithm is polynomial time.

2. (10 points) Privacy of survey data

Last year was the year of the Census. The results of a survey such as the Census are often disclosed in aggregate form to ensure the confidentiality of the information. Revealing nothing other than the aggregates also has a downside since that may provide too little information.

Suppose a survey has produced a $m \times n$ table D, in which each entry d_{ij} is a nonnegative integer. Let $r(i)$ be the sum of the elements in the ith row of D, and let $c(j)$ denote the sum of he elements in the jth column of D. The surveyor would like to disclose all the m row sums and the n column sums. Furthermore, the surveyor wants to disclose a subset, say S, of the mn matrix elements, and yet suppress the remaining matrix elements to ensure the confidentiality of the supressed information. Unless care is exercised, the surveyor may permit someone to deduce the exact value of one or more of the suppressed elements. We say that an entry d_{ij} of the matrix is *unprotected*, if it is possible to deduce the exact value of d_{ij} from the row and column sums and the elements of S.

Given m, n, the row and column sums, and the elements of S, our problem is to identify all the unprotected elements of the matrix and their values.

Using the maximum flow problem, give a polynomial-time algorithm for the above problem. Justify the correctness of your algorithm. You need not worry about the complexity of your algorithm, as long as it is polynomial-time. For instance, you may make multiple (polynomially many) calls to maximum flow subroutines, if needed.

Answer: We construct a network flow instance G as follows. The node set is as follows: a node R_i for each row i, a node C_j for each column j, a source node s and a sink t. For each row i, let $r'(i)$ denote $r(i) - \sum_{(i,j)\in S} d_{ij}$. For each column j, let $c'(j)$ denote $c(j) - \sum_{(i,j)\in S} d_{ij}$. We have an edge from s to each row node R_i with capacity $r'(i)$, an edge from each column node C_j to t with capacity $c'(j)$, and an edge from R_i to C_j for each $(i,j) \notin S$ with capacity ∞.

We now give an algorithm to determine whether a particular element (i,j) is protected. We compute a maximum flow f of G. Note that the flow value should equal $\sum_i r'(i)$ or $\sum_j c'(j)$. Let f_{ij} be the flow on the edge from R_i to C_j. We next compute a max flow on an instance G', which is identical to G except that we set the capacity of edge (R_i, C_j) to $f_{ij} - 1$. If we find that the maxflow in G' has value equal to $|f|$, then (i,j) is protected – since d_{ij} can take value f_{ij} as well as a value less than f_{ij}. Otherwise, we compute a maxflow on an instance G'', which is identical to G except that the capacity of the edges (s, R_i) and (C_j, t) are reduced to $r'(i) - f_{ij} - 1$ and $c'(j) - f_{ij} - 1$, respectively. This checks whether d_{ij} can be greater than f_{ij}. If there is a maxflow in G'' of value at least $\sum_i r'(i) - f_{ij} - 1$, then again (i,j) is protected. Otherwise, (i,j) is not protected.

We repeat the above for all pairs (i,j) in S.

3. (10 points) Design of a salad

Problem 7.16 of text.

Answer:

We can build the following model:

$$
\begin{aligned}
t*21 + l*16 + s*371 + c*346 + o*884 &= min \\
t*0.85 + l*1.62 + s*12.78 + c*8.39 &>= 15 \\
t*0.33 + l*0.2 + s*1.58 + c*1.39 + o*100 &>= 2 \\
t*0.33 + l*0.2 + s*1.58 + c*1.39 + o*100 &<= 6
\end{aligned}
$$

$$\begin{aligned} t*4.64+l*2.37+s*74.69+c*80.7 &>= 4 \\ t*9+l*8+s*7+c*508.2 &<= 100 \\ l+s &<= t+c+o \end{aligned}$$

Notice all the variables above are non-negative by default, so we don't have to add the non-negative restriction.

We can solve that the optimal solution is $\{t, l, s, c, o\} = \{5.88, 5.84, 0.04\}$, and the value is $232.5K$

4. (10 points) Optimal strategies in pizza business

Problem 7.14 of text.

Answer: Suppose Joey's and Tony's strategies are respectively $\{x_1, x_2\}$ and $\{y_1, y_2, y_3\}$, we have the following LP:

$$\begin{aligned} &maxz \\ z &<= 2x_1 - x_2 \\ z &<= -2x_2 \\ z &<= -3x_1 + x_2 \\ x_1 + x_2 &= 1 \\ x_1, x_2 &>= 0 \end{aligned}$$

And its duality LP as follows:

$$\begin{aligned} &minw \\ w &>= 2y_1 - 3y_3 \\ w &>= -y_1 - 2y_2 + y_3 \\ y_1 + y_2 + y_3 &= 1 \\ y_1, y_2, y_3 &>= 0 \end{aligned}$$

We can solve the value of this game is -1, and the optimal strategies for Tony and Joey are $\{1/2, 1/2\}$ and $\{0, 2/3, 1/3\}$ respectively.