CMSC 641, Design and Analysis of Algorithms, Spring 2010

Sandipan Dey,
Homework Assignment - 5

March 22, 2010

Dynamic Programming

Formulation 1

(a)

$MinCost(i, n) =$
Minimum cost to ship n bottles of vitamins using some of the boxes from the boxes $i \ldots m$ only, where $1 \leq i \leq m$.

(b)

$$MinCost(i, n) = min \begin{cases} MinCost(i+1, n), \text{ without using the } i^{th} \text{ box} \\ MinCost(i+1, n - x_i) + c_i, \text{ if } n > x_i, \text{ using the } i^{th} \text{ box} \\ c_i, \text{ if } n \leq x_i, \text{ using the } i^{th} \text{ box, no further box to be considered} \end{cases}$$

Base cases:

$$MinCost(m, n) = \begin{cases} c_m & \text{if } n \leq x_m \\ \infty & \text{otherwise} \end{cases}$$

$$MinCost(i, 0) = 0, \ \forall i = 1 \ldots m.$$

(c)

Runtime $= \theta(mn)$, since we have to fill a table of size $m(n+1) = \theta(mn)$ and computing each entry corresponding to a cell of the table takes constant time.

Formulation 2

(a)

$MinCost(i)$ = Minimum cost to ship i bottles of vitamins using some of the boxes from the boxes $1 \ldots m$.

(b)

$$MinCost(i) = \min_{1 \le j \le m} \begin{cases} MinCost(i - x_j) + c_j & \text{if } i > x_j \text{, using the } j^{th} \text{ box} \\ c_j & \text{if } i \le x_j \text{, no further box to be considered} \end{cases}$$

Base case:

$$MinCost(0) = 0$$

(c)

Runtime $= \theta(m.n)$, since we have to fill a table of size $n + 1 = \theta(n)$ and computing each entry corresponding to a cell of the table takes $\theta(m)$ time.

This problem is a slight variation of the coin change problem, where one is supposed to find minimum number of coins required to have a change for n dollars with coins of denominations $x_1 \ldots x_m$ and we are interested in minimizing the number of coins, i.e, $c_j = 1, \forall j = 1 \ldots m$, with the base cases differ only.

Network Flow

(a)

The network flow (with lower bounds) graph is presented in the figure 1 as a solution of this problem.

(b)

Apart from source (L_1) on the left and sink (L_5) on the right, we have 3 other different layers of node(s) in the flow network, let's descirbe them from left to right:

1. The 2^{nd} layer (L_2) consists of n nodes $s_i \ldots s_n$, i^{th} node represents student i.
   Since each student has to make k dishes, each student node s_i connected by an edge to exactly k of the nodes representing dishes $d_{il},\ l = 1 \ldots k$ on the right (in L_3).
   Each such edge (s_i, d_{il}) has both capacity lower and upper bounds equal to 3, since each of the dishes must be tested by 3 different students from L_4.
   Also, each student s_i is connected with the source node s on the left by an edge with capacity of both lower and upper bounds equal to $3k$.

2. The 3^{rd} layer (L_3) consists of $n \times k$ nodes that represent all the dishes prepared by n students.

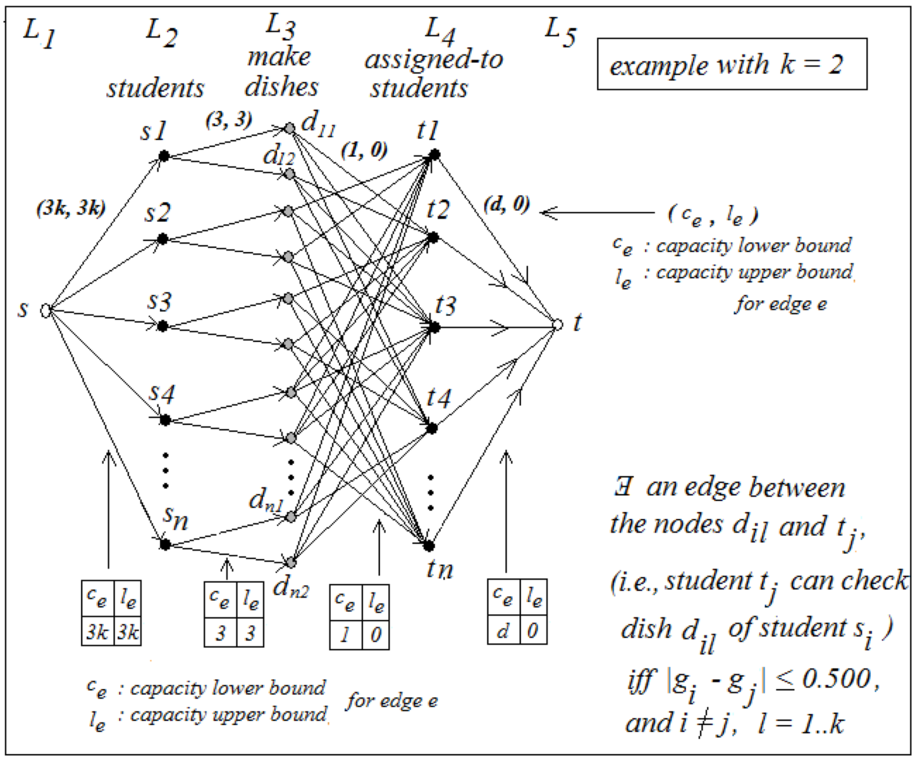

Figure 1: Network flow for the final exam problem at CCI

3. The 4^{th} layer (L_4) consists of n nodes $t_i \dots t_n$ that again represent the students that are going to check the dishes.
   Since a student can check only the dishes of the similar students (with GPA difference at most 0.500), a dish d_{il} will have an edge with a student t_j on the right (which means that the student can test that dish) iff s_i and s_j are similar, i.e., $|g_i - g_j| \leq n$ and $i \neq j$ (a student can't test his own dish). Moreover, such an edge will have capacity upper bound 1 and lower bound 0.
   Finally, since a student can evaluate at most d dishes, the edge from any t_j to the sink node t on the right will have capacity upper bound d but lower bound can be anything.

Alternative simpler design of the flow graph

(a)

The network flow (with lower bounds) graph is presented in the figure 2 as an alternative solution of this problem.

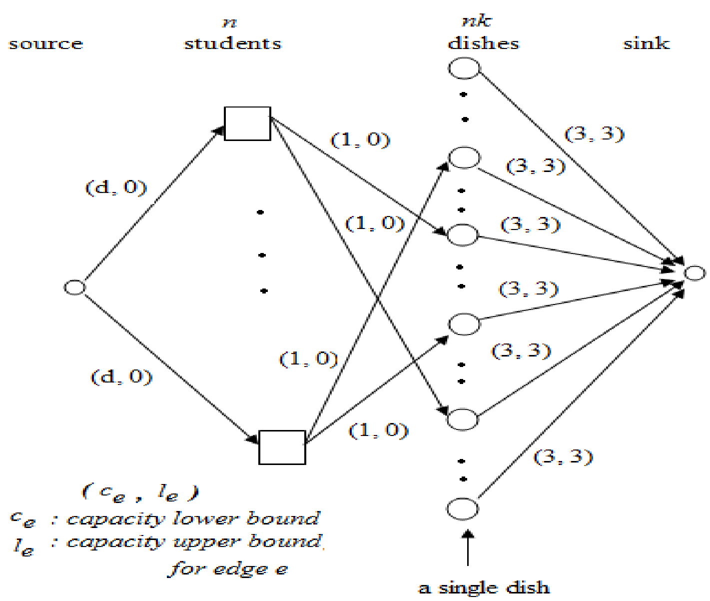

Figure 2: Network flow for the final exam problem at CCI

(b)

1. Each student can taste at most d dishes, hence each edge from the source s to a student has capacity upper bound d. However, the minimum capacity

can be 0, since the student may not be required to taste a dish.

2. As before, a student can check a dish iff it's not his own dish and the dish is prepared by a student with similar grade, only in that case there will be an edge between the student and the dish. The upper bound of the capacity will be 1 and lower bound will be 0, as obvious, for every such possible edge.

3. Finally, each edge from a dish to the sink must have minimum and maximum capacity 3, since a dish has to be evaluated by 3 different students.

This solution can still be modified to the following more compact network flow graph as presented in figure 3, by defining set D_i as a set of k dishes possessed by student s_i (s.t. $|D_i| = k$ and $|\bigcup_{i=1}^{n} D_i| = n.k$) and each node will denote a set of k dishes instead of a single dish.

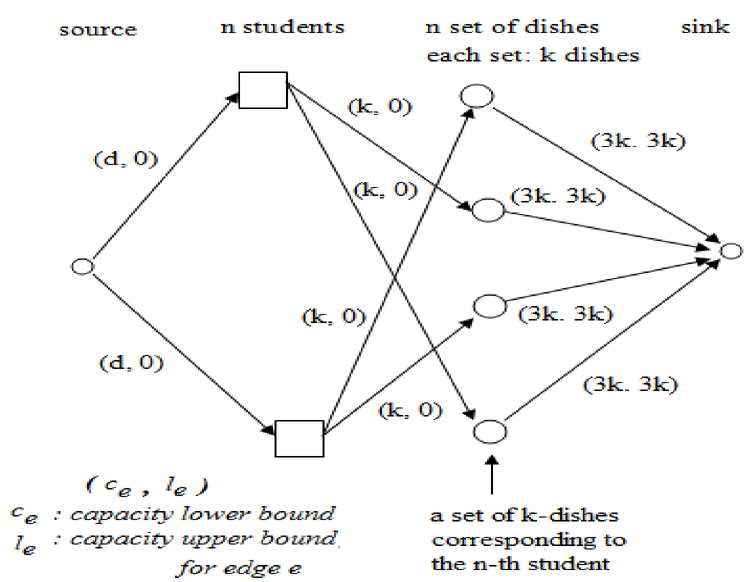

Figure 3: Network flow for the final exam problem at CCI

The problem to be solved here is nothing but a matching problem with a set of constraints. With all the above, it can be seen that the above flow network (with lower bounds) satisfies all the constraints and represents a contrained matching between the students and the dishes. Hence, maximum (feasible) flow in this network corresponds to a (feasible optimal) solution to the final exam problem.

(c)

The second solution can be easily modified by just changing the capacity upper bound on the edges in between a student and a k-dish set, to accomodate the additional constraint, as presented in the following figure 4.
Earlier a student was allowed to taste any number of dishes from 0 (none) to k, all the dishes corresponding to another student with similar grade. But now if we upperbound the capacity of this edge by $k-1$ instead, each evaluator is restricted to tasting at most $k-1$ dishes prepared by a single student.

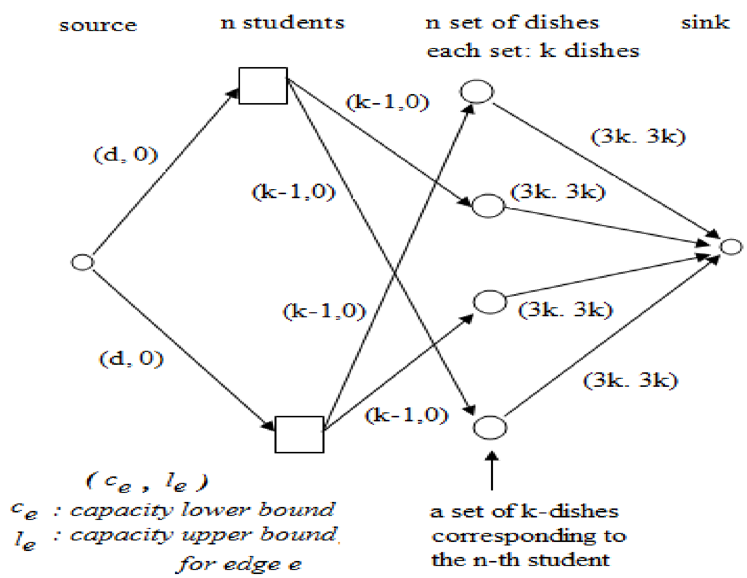

Figure 4: Network flow for the final exam problem at CCI, with the additional constraint

Problem 29-2.2

1. Let's denote the shortest path from node s to node y in the connected graph $G(V, E)$ as p_{sy}.
2. Any arbitrary edge $(u, v) \in E(G)$ may or may not be on the shortest path. Let's define the indicator variable $x_{uv} = \begin{cases} 0, & (u,v) \in p_{sy} \\ 1, & ow \end{cases}$, $\forall (u, v) \in E(G)$. Hence, we have $x_{uv} \geq 0, \forall u, v \in V(G)$.
3. s is the starting vertex and we are interested in finding the shortest path that is a simple path, i.e., with no repeated vertex $\Rightarrow$ there is no vertex

$u \in V(G)$ s.t. there is an edge $(u,s) \in p_{us}$ entering into $s \Rightarrow x_{us} = 0,\ \forall u \in V(G) \Rightarrow \sum_{u \in V(G)} x_{us} = 0$ (no self loops).
Also, there must be an (exactly one) edge $(s,w) \in E(G)$ going out from s that is on the shortest path, i.e., s is on the path $p_{sy} \Rightarrow \exists w \in V(G)\ :\ x_{sw} = 1$. Putting it together, $\sum_{u \in V(G)} x_{us} - \sum_{w \in V(G)} x_{sw} = -1$.

4. Similarly, there must be an (exactly one) edge $(w,y) \in E(G)$ going into y which is on the shortest path. Since y the end vertex of the path p_{sy} which is of non-zero length (since G is connected) $\exists w \in V(G)\ :\ x_{wy} = 1 \Rightarrow \sum_{w \in V(G)} x_{wy} = 1$ but no outgoing edge from w on the shortest path $\Rightarrow \sum_{u \in V(G)} x_{wy} - \sum_{w \in V(G)} x_{yw} = 1$.

5. Any vertex $v \in V(G)$ other than s and y either ill be on the path or will not be, in either cases, we shall have $\sum_{u \in V(G)} x_{uv} - \sum_{w \in V(G)} x_{vw} = 0,\ \forall v \neq s, y$.

6. Now, with all the above constrainst we shall be interested to find the shortest path, i.e., solve the minimization problem $min \sum_{(u,v) \in E(G)} w(u,v).x_{uv}$, where w_{uv} is the weight of the edge (u,v).

7. Hence our linear program is:

$$\begin{aligned} min \sum_{(u,v) \in E(G)} w(u,v).x_{uv} \\ \text{s.t.} \sum_{u \in V(G)} x_{us} - \sum_{w \in V(G)} x_{sw} = -1, \\ \sum_{u \in V(G)} x_{wy} - \sum_{w \in V(G)} x_{yw} = 1, \\ \sum_{u \in V(G)} x_{uv} - \sum_{w \in V(G)} x_{vw} = 0,\ \forall v \neq s, y, \\ x_{uv} \geq 0, \forall u, v \in V(G) \end{aligned}$$