NETWORK OPTIMIZATION

Megan Bryant
mrbryant@email.wm.edu

Department of Mathematics, The College of William and Mary, P.O. Box 8795, Williamsburg, VA 23187

Executive Summary

Homework 4.
Due date: February 19, at the start of class
Collaborators:

4 Beam Assortment Problem

4.1 *Suppose you have a directed graph $G = (V, A)$ with non-negative arc lengths c_{ij} . You want to determine the shortest path length that can start either at nodes s_1 or s_2 and can end either at nodes t_1 or t_2. (These four nodes are all nodes in the graph G.) Explain how you could use your code for Dijkstras algorithm to solve this problem. You are not allowed to change your code, and you can run Dijkstras algorithm only once!*

4.2 *In this question, we consider a generalization of the Beam Assortment Problem. In the discussion of that application, we assumed that if we must cut a beam of length 5 units to a length of 2 units, we obtain a single beam of length 2 units; the remaining 3 units have no value. However, in practice, from a beam of length 5, we could cut two beams of length 2, and the remaining length of 1 unit will have some scrap value.*

a.) *Modify the formulation as a shortest path problem that we saw in class to incorporate these changes. You do not have to describe the formulation in general; instead, you can just show the construction for the following example:*

type	K_j	L_j	D_j	C_j
1	1	1	5	1.5
2	1	2	10	2
3	1	3	10	2.5

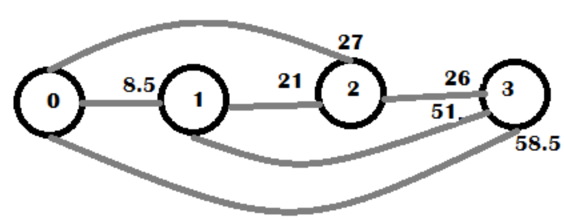
D_j : Demand of steel beam of length L_j
K_j: Cost for setup up the inventory facility to handle beams of length L_j.
a_{ij}: Number of beams i cut from beam of length j. $= \frac{L_j}{L_i}$.
C_j: Cost of a beam of length L_i.
c_{ij}: Cost of arc (i, j). $= K_j + C_j \sum_{k=i+1}^{j} \left(D_k * \lceil \frac{1}{a_{kj}} \rceil \right)$

min $c_{ij} x_{ij}$

$$\text{s.t. } \sum x_{ij} - \sum x_{ji} = \begin{cases} n-1 & \text{for } i = s \\ -1 & \text{for all } i \in N - \{s\} \end{cases}$$

$\sum a_{ij} x_{ij} \geq D_i$
$x_{ij} \geq 0$ for all $(i, j) \in A$

The scrap value is 0. Draw the graph, find the shortest path from node 0 to node 3, and explain what the corresponding solution to the Beam Assortment Problem is.



$$C_{01} = K_1 + C_1 \sum_{k=1}^{j} \left(D_k * \lceil \frac{1}{a_{kj}} \rceil \right) = 8.5$$

$$C_{02} = K_2 + C_2 \sum_{k=1}^{j} \left(D_k * \lceil \frac{1}{a_{kj}} \rceil \right) = 27$$

$$C_{03} = K_3 + C_3 \sum_{k=1}^{j} \left(D_k * \lceil \frac{1}{a_{kj}} \rceil \right) = 58.5$$

$$C_{12} = K_2 + C_2 \sum_{k=2}^{j} \left(D_k * \lceil \frac{1}{a_{kj}} \rceil \right) = 21$$

$$C_{13} = K_3 + C_3 \sum_{k=2}^{j} \left(D_k * \lceil \frac{1}{a_{kj}} \rceil \right) = 51$$

$$C_{23} = K_3 + C_3 \sum_{k=3}^{j} \left(D_k * \lceil \frac{1}{a_{kj}} \rceil \right) = 26$$

We know based on the setup of the problem that we are looking for the shortest path from node 0 to node 5. Therefore, we would use a label correcting algorithm as opposed to a label setting algorithm.

We see from the above graph and the arc cost calculations, that the shortest path from 0 to 3 is arcs $(0, 1), (1, 2)$, and $(1, 3)$ with a cost of 58.5. Meaning, all beams should be cut from beams of length 3.

b.) *There exists a solution to this Beam Assortment Problem that is cheaper than the one you found in part (a). Give such a solution.*

Table 1: Cutting Stock, $L_1 = 1$

Pattern #	1
a_{1j1}	1
a_{2j1}	0
a_{3j1}	0
Total	1

Table 2: Cutting Stock, $L_2 = 2$

Pattern #	1	2
a_{1j2}	2	0
a_{2j2}	0	1
a_{3j2}	0	0
Total	2	1

Table 3: Cutting Stock, $L_3 = 3$

Pattern #	1	2	3
a_{1j3}	3	1	0
a_{2j3}	0	2	0
a_{3j3}	0	0	1
Total	1	2	3

We see from the above cutting stock pattern tables that the most optimal solution is cutting patterns a_{2j3}, a_{3j3}, and a_{2j2} in the amounts of $5, 10,$, and 5 respectively. This results in a total cost of 49.5, which is a significant reduction on the shortest path solution of 58.5.

c.) *Explain why the shortest path formulation not a correct formulation of the modified Beam Assortment Problem.*

The shortest path formulation isn't the correct choice for this modified beam assortment problem since we can cut multiples of a lower level type beam from a higher level type beam. This means that we can have multiple costs on each arc depending on how many we cut from each larger sized beam. We can also make different pattern cuts, meaning from beams of size 3, we can either cut three size 1s, one size 2, one size 3, or one size 1 and one size 2.This means that the shortest path solution set doesn't include the optimal beam assortment solution as it the shortest path problem doesn't account for all of the variations. It would be better formulated as a cutting stock problem.

4.3 *Beverly wants to rent her vacation home in Cape Cod for the period from July 1 to July 15. She has received the following bids, which each give the day the rental starts (check in 3 p.m.), the day it ends (check out noon), and the bid for the total stay:*

start	end	bid ($)
July 1	July 8	800
July 1	July 15	1600
July 7	July 13	900
July 12	July 15	300
July 12	July 13	200
July 13	July 15	300
July 3	July 5	500
July 2	July 11	1500
July 9	July 12	300

a.) *Formulate the problem of finding the selection of bids that maximizes her revenue as a shortest path problem. (Recall that maximizing $\sum_j c_j x_j$ is the same as minimizing $\sum_j (-c_j) x_j$, so you may want to first find a formulation that has negative arc lengths, and then figure out how to modify your formulation so that all arc lengths are non-negative.) Follow the three steps outlined in class to describe your construction, and to argue why your shortest path formulation correctly models Beverlys problem.*

Step 1: Explain how to take the input to problem A and turn it into input for problem B. Given the input from Beverly's Profit problem, we will consider the following:

Nodes:

- Special Source Node 0 and sink node 16. Source node connects to every node with an arc coming out of it. Sink node connects to every node with an ending arc.
- nodes $1, \ldots, n$ representing days $1, \ldots, n$

Arcs:

- Directed arc (i, j) if $j > i$. To be intepreted as: if (i, j) is on chosen, we accept the minimal cost bid (highest profit) starting on day i and ending on day j.
- Arc Length: $c_{ij} = -p_{ij}$. Note: If two bids occupy the same dates, the smaller bid is disregarded as it can never be optimal.

Step 2: Explain how the solution fo problem B gives a solution to problem A.

We can formulate this example by using the principles discussed in section 4.4, application 4.3, the knapsack problem. As in the knapsack problem application, we can find the solution to Beverly's problem by finding the longest path in the graph. We know that the longest path and the shortest path are 'closely related' and we can begin by considering cost, $c_{ij} = -p_{ij}$. Since all of the profits in the longest path formulation were positive, all of the costs in the shortest path formulation will be negative and will not contain any positive length directed cycle. Therefore, all paths in the shortest path problem have nonnegative lengths and we can solve this problem efficiently using the shortest path methodology.

Step 3: Explain why the solution for problem A from Step 2 must be the optimal solution for problem A.

The solution from step 2 must be the optimal solution for problem A because it corresponds to the longest path solution of the negative utility (positive cost). The longest path is the path with the most profit and therefore, by definition, must correspond to the optimal solution.

b.) *Solve the problem from (a) using Dijkstras algorithm. It is your choice whether you want to do so by hand, or using your Dijkstra implementation.*

Below, we have a node and arc list in which the nodes that can be reached from a certain node are labeled with the appropriate cost and the nodes that cannot be reached are labeled with $-\infty$. Nodes which are unconnected are not included for simplicities sake. Note that the subscript denotes the predecessor.

Table 4: Node and Arc List

S	d(1)	d(2)	d(3)	d(5)	d(7)	d(8)	d(9)	d(11)	d(12)	d(13)	d(15)
{0}	0	0	0	0	0	0	0	0	0	0	0
{1}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-800_1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-1500_1
{2}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-1500_2	$-\infty$	$-\infty$	$-\infty$
{3}	$-\infty$	$-\infty$	-500_3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
{7}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-900_7	$-\infty$
{9}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-300_9	$-\infty$	$-\infty$
{12}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-200_{12}	$-\infty$
{13}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-300_{13}
{16}	0	0	0	0	0	0	0	0	0	0	0

We will now perform Dijkstra's algorithm. Note that the underlined node is the node chosen and the subscript is the predecessor. If the end node selected is not a predecessor for another node, then we will select the next node which is a predecessor and continue.

Table 5: Dijkstra's Algorithm

S	d(1)	d(2)	d(3)	d(5)	d(7)	d(8)	d(9)	d(11)	d(12)	d(13)	d(15)
$\{0\}$	0	$\underline{0}$	0	0	0	0	0	0	0	0	0
$\{0,2\}$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$\underline{-1500_2}$	$-\infty$	$-\infty$	$-\infty$
$\{0,2,12\}$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$\underline{-200_{12}}$	$-\infty$
$\{0,2,12,13\}$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$\underline{-300_{13}}$
$\{0,2,12,15\}$	0	0	0	0	0	0	0	0	0	0	$\underline{0}$

Therefore, we see that the shortest past of the utility (corresponding to the longest path of the profit) is the set containing nodes $\{0, 2, 12, 13, 15\}$ Corresponding to bids starting on day 2, 11, and 13 of $1500, 200$, and 300 respectively for a total profit of $\$2,000$ for Beverly.