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 8.
Due date: April 10th, at the start of class
Collaborators:

1

(Library Secondary Storage) A library facing insufficient primary storage space for its collection is considering the possibility of using secondary facilities, such as closed stacks or remote locations, to store portions of its collection. These options are preferred to an expensive expansion of primary storage. Each secondary storage facility has limited capacity and a particular access costs for retrieving information. Through appropriate data collection, we can determine the usage rates for the information needs of the users. Let b_j denote the capacity of storage facility j, and v_j the access cost per unit item from this facility. In addition, let a_i denote the number of items of a particular class i requiring storage, and let u_i denote the expected rate (per unit time) that we will need to retrieve books from this class. Our goal is to store the books in a way that will minimize the expected retrieval cost.

a.) *A transportation problem is a special case of the minimum cost flow problem, in which the graph $G = (N, A)$ is bipartite and the nodes "on the right" have a negative supply. More formally, the node set can be divided into N_L, N_R such that every arc (i, j) has $i \in N_L, j \in N_R$, and $b_i \geq 0$ for all $i \in N_L$, and $b_i \leq 0$ for all $i \in N_R$.*

i. *Assume that $\sum_i a_i = \sum_j b_j$, i.e. the amount of space available in secondary storage is exactly equal to the amount of space required. Show how to formulate this problem as a transportation problem.*

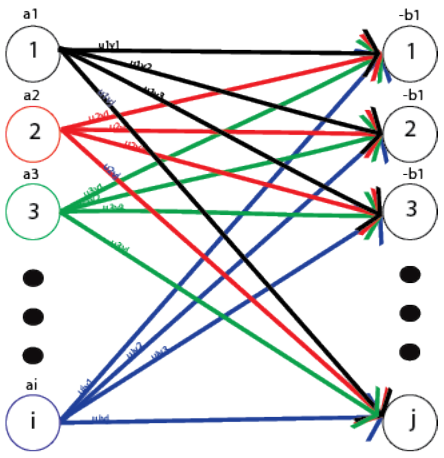
The Library Second Storage Problem is an instance of a Factored Transportation problem, a special case of the TP. This means that it is a transportation problem having cost coefficients of the form $c_{ij} = u_i v_j$. Therefore, we can formulate it as a traditional TP with adjusments. The modified model is below.

a_i:	The number of items in class i requiring storage.
b_j:	Capacity of storage facility j.
x_{ij}:	The number of items of class i stored in facility j.
u_i:	Expected number of retrievals of any class i item per unit time.
v_j:	Access cost per unit for storage facility j.

$$\min \sum_{i=1}^{m} \sum_{j=1}^{m} u_i v_j x_{ij}$$

$$\begin{aligned} \text{s.t. } & \sum_j x_{ij} = a_i, \forall i \\ & \sum_i x_{ij} = b_j, \forall j \\ & x_{ij} \geq 0 \end{aligned}$$

Now, we know that because we are formulating our problem as a minimum cost flow in which the graph is bipartite that all of the b_i values, that is the capacity of facility j values, must be negative. To accomplish this, we will simply multiply the given positive values by -1. Let m represent the number of item classes and n represent the number of storage facilities. Then, we can begin constructing our graph for the min cost flow algorithm by creating m item class nodes and n facility nodes. Next, we must make the graph complete by adding arcs from every item class node i to every facility node j. The cost on these arcs (i, j) is $u_i v_j$ and the capacities infinite. The visualization of this graphical formulation is shown below.



We know that a feasible solution to the Min Cost Flow problem exists if and only if the max flow problem has a maximum flow value of $\sum_{i \in N_L} a_i > 0$. To find our initial feasible flow, we must add a source node s, arcs (s, i) from the source to item class nodes, set the capacity of those arcs to a_i, add arcs (j, t) for each facility node j, and set the capacity of those arcs to the negative of the b_i. We then find the max flow to use as initial feasible solution. We know that a max flow exists since we have stipulated that $\sum_i a_i = \sum_j b_j$.

The flow along the an arc (i, j) tells us how to allocate the items of each class. Meaning, the flow x_{ij} along any arc in the min cost graph tells us the number of items of class i stored in facility j. The cost of the expected number of retrievals per unit time, u_i can similarly be found by multiplying the access cost per unit for storage facility j, v_j, by the number of items of class i retrieved from facility j, that is (x_{ij}, u_i). Flow balance equations at each of the nodes ensures that every item in every item class is stored.

ii. *Extend your formulation to deal with the case when* $\sum_i a_i < \sum_j b_j$.

If $\sum_i a_i < \sum_j b_j$, then we can invent a new class of items requiring storage, called $i + 1$ with the number of items in that class $a_{i+1} = \sum_j b_j - \sum_{i=1}^{i} a_i$ (the excess in storage). We can then let $u_{i+1} = 0$. We must also include additional variables $x_{i+1,j}$ in the model. Since there are no real items in class $i + 1$, we thus know that any flow along the arcs originating from $i + 1$ represents place holders with no associated costs. Without loss in generality, we can then relabel these item classes from 1 to i. Thus, we see that we once again have $\sum_i a_i = \sum_j b_j$.

This new model is therefore equivalent to the model from part **i** with the items in class i acting as empty space fillers. Storing these blank items in any facility costs nothing because the costs of accessing one of these blank items, $c_{i+1,j} = u_{i+1} v_j = 0$, is nothing as the expected number of retrievals of these items is nothing.

Therefore, we can solve this variation the same as before, utilizing max flow to find an initial feasible solution and then min cost algorithm to find our optimal minimum cost.

b.) *This problem has a special structure, which allows for a very simple algorithm for finding the optimal solution. Give a simple rule for finding the optimal assignment of items to storage facilities and prove that your rule finds the optimal solution.*

We know that the cost to store items from a class i in a facility j is $c_{ij} = u_i v_j$. We know by our objective that we want to minimize cost as much as possible. Therefore, without loss of generality, let's begin by reordering the item classes such that for all $i = 1, \dots, m$, $a_{i+1} \geq a_i$. Thus we have the item classes ordered by least items to most items. Then, we can begin at item class 1 and push flow along the path $s - 1 - k$ where j is the facility with the lowest cost $c_{1k} \leq c_{1j}, \forall j$. If the items in class 1 are exhausted, we can move onto the next class. Otherwise, we can continue by pushing as much flow as possible on the path to the next cheapest storage facility. We continue on in this fashion until all items in every class are stored.

Succinctly, this is as follows:

Optimal Factored Transportation

```
begin
        $a_i \geq a_{i+1}, \forall i = 1, \dots, m$
        $i = 1$
        while $a_i > 0$ do
        begin
            $j = \min\{c_{ij}\}$
            $x_{ij} = \max\{a_i - b_j, a_i\}$
            $a_i \leftarrow a_i - b_j$
        end;
        $i \leftarrow i + 1$
end;
```

This algorithm provides an optimal solution because for each item class i, the retrieval costs are the most minimum possible of the available choices. Since each retrieval cost is the smallest that it can be, we know that the objective must be optimal.

2

(Airplane Ticketing) In this problem, you will extend the Hopping Airplane problem to solve the following problem from Rovbert Vanderbei's Linear Programming book:

A small airline, Ivy Air, flies between three cities: Ithaca, Newark, and Boston. They offer several flights, but, for this problem, let us focus on the Friday afternoon flight that departs from Ithaca, stops in Newark, and continues to Boston. There are three types of passengers :

1. Those traveling from Ithaca to Newark.
2. Those traveling from Newark to Boston.
3. Those traveling from Ithaca to Boston

The aircraft is a small commuter plane that seats 30 passengers. The airline offers three fair classes :

1. Y class: full coach.
2. B class: nonrefundable.
3. M class: nonrefundable, 3-week advance purchase.

Ticket prices, which are largely determined by external influences (i.e. competitors) have been set and advertised as follows :

	Ithaca-Newark	Newark-Boston	Ithaca-Boston
Y	300	160	360
B	220	130	280
M	100	80	140

Based on past experience, demand forecasters at Ivy Air have determined the following expected number of potential customers in each of the 9 possible origin-destination fair class combinations :

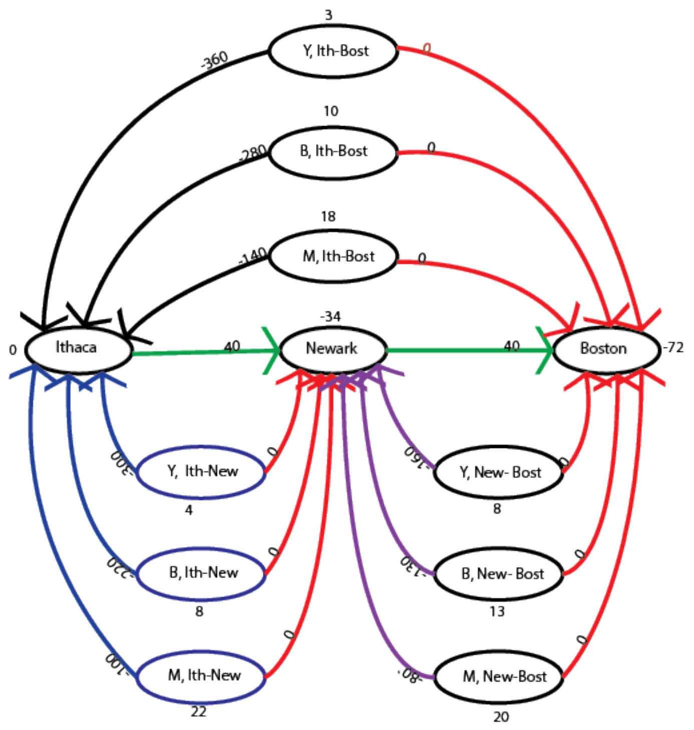
	Ithaca-Newark	Newark-Boston	Ithaca-Boston
Y	4	8	3
B	8	13	10
M	22	20	18

The goal is to decide how many tickets from each of the 9 origin/destination/fare combinations to sell. The constraints are that the plane cannot be overbooked on either of the two legs of the flight and that the number of tickets made available cannot exceed the forecasted maximum demand. The objective is to maximize the revenue. Formulate this problem as a min cost flow problem.

In order to get a proper understanding of our problem it is best to begin by defining each of the parameters/variables.

I:	The set of cities: Ithaca, Newark, Boston.
T:	The set of fare types: Y. B. M.
b_{ijt}:	The demand for travel from city i to city j of type t.
f_{ijt}:	The fare for travel from city i to city j of type t.
u_{ij}:	The capacity of the plain from city i to city j.
x_{ijt}:	The flow along the graph.

We will begin by imagining our graph structure. Let there be a node for each leg of the flight designated $(i-j,t)$ such that i is the originating city and j is the destination city and t is the fare type. Let there also be nodes for each city with designation $i \in I$. Now, we will add arcs originating at nodes $(i-j,t)$ to both i and j. Let the cost on arc $((i-j,t),i)$ be $-f_{ijt}$ and the capacities ∞. Thus, the flow on these arcs will represent the passengers of this fare type travelling from city i to j. Let us also add arcs $((i-j,t),j)$ with cost 0 and capacity ∞. These arcs will represent unfilled demand, that is they will represent passengers of fare type t who did not receive a seat on flight $i-j$. Now, let's add arcs from $i-j$ for $i = 1,\ldots,(|I|-1)$ and $j = 2,\ldots,|I|$ with capacity u_{ij}. These arcs represent the actual capacity of the flight. For the city nodes, the b_i values should be set to the sum of the negatives of the b_{ijt} values for nodes with arcs with cost 0 ending at city i. Thus we have the following graphical formulation.



This min cost flow problem solves the airplane ticketing problem if and only if the sum of the demand equals the sum of the people travelling and a feasible integer flow from a source node to a sink node (both not yet generated) exists. We should first create a source node s and arcs $(s, (i-j, t))$ connecting the source node to all possible flight/fare combinations. The capacities on these arcs should be the demands b_{ijt} Then, we need to add a sink node t and an arc (i, t) connecting each city to the sink. The capacity on these arcs will be the negative of the demand b_{ijt}. We know that there exists a max flow since the sum of the demand equals the sum of the people travelling. Thus, a feasible integer flow in the graph at a cost of C (which is actually profit since its negative) corresponds to a feasible way selling tickets for each flight/fare combination with a revenue of $-C = \sum_{(i,j)\in A} f_{ijt} * x_{ijt}$.

The flow on an arc from a destination i to a destination $i+1$ represents the passengers on the plane between destination i and $i+1$. The capacity of these arcs ensures that the plane doesn't exceed capacity. Flow balance equations at each node ensures that every passenger is accounted for. Thus, we can use the minimum cost flow problem to solve our airplane ticketing problem.

3

(The Napkin Laundering Problem) A caterer has to provide fresh napkins over a period of n *days; the number* d_j *of napkins required on day* j *is known in advance. To satisfy these requirements, the caterer can either*

- Buy new napkins (at a cents apiece)
- Have the used napkins laundered
  - using fast service (napkins returned q days later for b cents a piece).
  - using slow service (napkins returned p days later for c cents a piece).

For concreteness, you may work with the following example: $n = 6, p = 4, q = 2. a = 200, b = 75, c = 25.$

j	1	2	3	4	5	6
$d(j)$	50	60	80	70	50	60

Formulate the caterer's problem as a minimum cost network flow problem. As usual, make sure to explain why the minimum cost flow problem you set up is equivalent to finding the minimum cost solution for the caterer. Warning: My guess is that this problem is going to take a long time to figure out. It may be helpful to brainstorm on this problem in a group, and I can also give you hints if you are completely stuck.

First, let's begin by considering the information that we are given. The following parameters are specified

n:	The number of days needing napkins.
d_j:	The demand for napkins on day i.
a:	The cost of a new napkin.
b:	The cost for the fast laundry service.
c:	The cost for the slow laundry service.
q:	The time for the fast laundry service.
p:	The time for the slow laundry service.

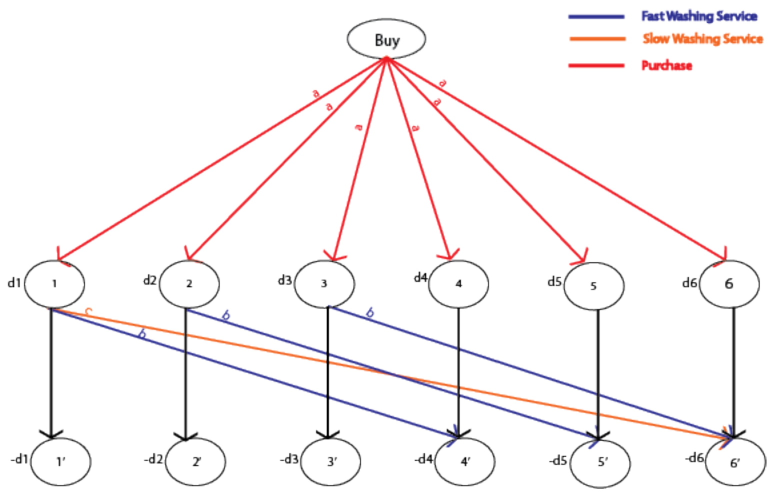
Now, we must immediately take note of a few things. We are not given a number for starting inventory, therefore we must assume that we begin with 0 clean napkins in stock. We are also not told what time of day the napkins are needed and what time of day the napkins are returned from the laundry service. Therefore, we must make the assumption that the napkins cannot be used on the same day which they are returned from the laundry service. Therefore, they are available for use 1 day after ($p + 1$ for the slow and $q + 1$ for the fast service). We have also not been informed of any restrictions regarding the number of napkins that each washing service can process, therefore we shall assume that the capacities of these services is in fact infinite.

We are asked to formulate this problem as a minimum cost network flow problem. Based on the information at hand, this seems to be a variation of the project selection problem from our last homework (at least the basic premises seem similar). Since we are using a minimum cost flow problem, we know that we need both supply and demand nodes. Therefore, it makes sense to have supply nodes $i = 1, \ldots, n$ with simple numerical designation and an equal number of demand nodes designated that numerical primed (i.e. supply node 1 and demand node 1').

Now, we must add the appropriate arcs. The supply nodes should all have an arc originating from a buy node designated 'buy' (for simplicity). The costs on these arcs should be a, the cost of purchasing a napkin, and the capacities infinite. Then, we need to add arcs (i, j) such that i is a supply node and j is the corresponding primed demand node. The cost on these arcs should be 0 and the capacities infinite.

Now, we must model the laundry services. For the fast service, we can accomplish this by creating arcs that originate at a supply node i (beginning at node 1) and terminate at a demand node $j = i+q+1 : j \leq n$. The cost of these arcs should be b and the capacities infinite. For the slow service, we can accomplish this by creating arcs that originate at a supply node i (beginning at node 1) and terminate at a demand node $j = i + p + 1 : j \leq n$. The cost of these arcs should be c and the capacities infinite.

The following is the graphical formulation. If the cost of an arc is not denoted, then it is 0. The laundry times for the concrete example are shown to illustrate.



This min cost flow problem solves the napkin problem if and only if the sum of the supply equals the sum of the demand and a feasible integer flow from a source node to a sink node (not yet generated) exists. The 'buy' node services as the source node for the purposes of finding this feasible flow. A sink node must be added with arcs (j, t) such that j is a demand node and t is the sink node. The capacity of these arcs is infinite and the cost is zero. A feasible integer flow in the graph at a cost of C corresponds to a feasible way of supplying fresh napkins each day satisfying demand. The flow on arcs terminating at the demand nodes represent the method of satisfying demand. If the flow x_{ij} for demand node j is coming from supply node $i = j$, then we know that we have purchased the napkins at a cost of $a * x_{ij}$. If the flow is coming from node $i = j - p - 1$ or $i = j - q - 1$, then we know that we have utilized either the slow or fast laundry service at a cost of cx_{ij} or bx_{ij} respectively. Flow balance constraints at each of the nodes ensures that the demand of each day is met. Thus, we can use the minimum cost flow problem to solve our napkin problem.