APMA 1210 Midterm 2 Practice Exam

November 2022

1 Shadow Prices

(a) Note that although this is a minimization problem, we can compute shadow prices the same way; it's just that an increase in z is bad instead of good. Changing the first constraint to $E+B \leq 6$ will not change the optimal solution, so its shadow price is zero. Similarly, changing the second constraint to $12E + 8B \geq 25$ also will not change the optimal solution - it swaps the vertex $(0, 3)$ of the feasible region for a pair of vertices $(0.25, 2.75)$ and $(0, 3.125)$ but neither of these yields a smaller z-value. Therefore this constraint also has a shadow price of zero. What if we change the third constraint to $E + 2B \geq 5$? Then the feasible region changes dramatically, with vertices at $(1, 2)$ and $(5, 0)$ - overlapping an existing vertex - instead of $(2, 1)$ and $(4, 0)$. The optimal value is now $z = 0.8$ at the vertex $(1, 2)$, meaning 0.1 is the shadow price for this constraint. For the fourth constraint, changing it to $12E + 12B \geq 37$ means that it now intersects the constraint $E + 2B \geq 4$ at $(2\frac{1}{6}, \frac{11}{12})$, and this vertex remains optimal with $z = 0.7083$ as the new optimal value. Therefore the shadow price is 0.0083 for this constraint.

(b) For ease of finding the dual, first rewrite the first constraint to be $-E - B \geq -5$. Using variable names that correspond to the meanings of each constraint, the dual problem for this LP is as follows:

$$\begin{aligned}
\text{Maximize: } & z = -5T + 24F + 4P + 36C \\
\text{Subject to: } & -T + 12F + P + 12C \leq 0.2 \\
& -T + 8F + 2P + 12C \leq 0.3 \\
& T, F, P, C \geq 0
\end{aligned}$$

Solving this LP yields an optimal value of $z = 0.7$ (matching the original problem) at the vertex $(0, 0, 0.1, 0.0083)$ which corresponds precisely to our previous findings.

2 Game Theory

(a) The gains matrix is as follows, with a 1 on the row/column indicating the choice to hold up 1 finger, and a 2 indicating the choice to hold up 2 fingers. Note that this is the gains matrix for Player 1 in particular:

	1	2
1	2	-3
2	-3	4

(b) First let x_1, x_2 represent the probabilities that Player 1 raise 1 finger or 2 fingers. Player 2's best strategy to defend against Player 1 is established by $z = \min\{2x_1 - 3x_2, -3x_1 + 4x_2\}$, i.e. $z \leq 2x_1 - 3x_2$ and $z \leq -3x_1 + 4x_2$ as inequality structures. This means that Player 1, seeking to modify these probabilites to maximize their own gains, should solve the LP:

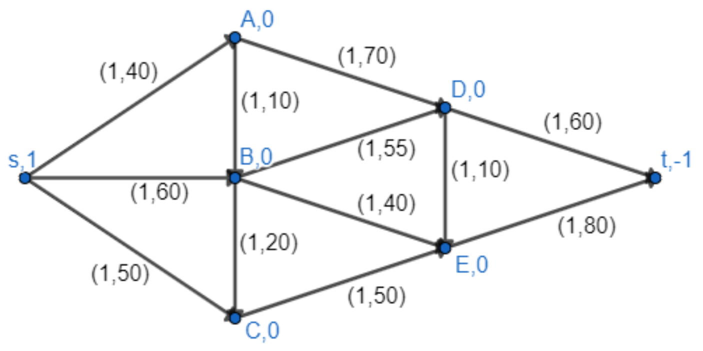
$$\begin{aligned} \text{Maximize: } & z \\ \text{Subject to: } & -2x_1 + 3x_2 + z \leq 0 \\ & 3x_1 - 4x_2 + z \leq 0 \\ & x_1 + x_2 = 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Conversely, let y_1, y_2 represent the probabilities that Player 2 raise 1 finger or 2 fingers. Player 1's best strategy to make advances against Player 2 is established by $z = \max\{2y_1 - 3y_2, -3y_1 + 4y_2\}$, or in the form of inequalities, $z \geq 2y_1 - 3y_2$ and $z \geq -3y_1 + 4y_2$. Therefore to minimize the losses for Player 2, the following LP should be solved:

$$\begin{aligned} \text{Minimize: } & z \\ \text{Subject to: } & -2y_1 + 3y_2 + z \geq 0 \\ & 3y_1 - 4y_2 + z \geq 0 \\ & y_1 + y_2 = 1 \\ & y_1, y_2 \geq 0 \end{aligned}$$

3 Setting Up a Network LP

(a) We will construct this network by drawing an edge from one town to another town if a road exists between them. The edge will be directed according to the start and end information in the table. Each edge has a cost in the form of a distance according to the table, but no explicit capacity; however, since we are planning a journey for a single car, we should enforce a capacity of 1 on each edge. For the nodes, the origin is supplying 1 car and the campsite has a demand of 1 car, but all other nodes have demands of 0. The resulting network is shown below, using s to represent the starting point and t to represent the campsite.



(b) Since all edges have the same capacity, we can combine their capacity constraints with their non-negativity constraints for a constraint of the form $0 \leq x_{ij} \leq 1$ where edge (i, j) exists in the network, $i, j \in \{s, A, B, C, D, E, t\}$. Our objective function will have the form $z = \sum c_{ij} x_{ij}$ using the costs on each edge, and every intermediate node will have a flow balance constraint of the form $\sum\{\text{edges out}\} - \sum\{\text{edges in}\} = 0$. The origin s and campsite t will have similar constraints, but with demands of 1 and -1, respectively. The resulting LP is:

$$\begin{aligned}
\text{Minimize: } & z = 40x_{sA} + 60x_{sB} + 50x_{sC} + 10x_{AB}70x_{AD} + 20x_{BC} + 55x_{BD} \\
& \quad + 40x_{BE} + 50x_{CE} + 10x_{DE} + 60x_{Dt} + 80x_{Et} \\
\text{Subject to: } & x_{sA} + x_{sB} + x_{sC} = 1 \\
& x_{AB} + x_{AD} - x_{sA} = 0 \\
& x_{BC} + x_{BD} + x_{BE} - x_{sB} - x_{AB} = 0 \\
& x_{CE} - x_{sC} - x_{BC} = 0 \\
& x_{DE} + x_{Dt} - x_{AD} - xBD = 0 \\
& x_{Et} - x_{BE} - x_{CE} - x_{DE} = 0 \\
& - x_{Dt} - x_{Et} = -1 \\
& 0 \leq x_{ij} \leq 1, (i, j) \text{ is in the network}, i, j \in \{s, A, B, C, D, E, t\}
\end{aligned}$$

4 Network Simplex Algorithm

If we start from the leaf e, the edge (g, e) will need to have a weight of 2 to satisfy the demand at that leaf. This accounts for 2 of the 5 units that need to be sent out from g, which has only one other incident edge, so we immediately find that the edge (g, b) should have a weight of 3 to balance the demand at g. Then b has 3 units in and a demand of 0, so it must send 3 units out along its only other incident edge (b, a), meaning this edge has a weight of 3. Note that aside from (b, a), a has two other edges incident on it, so we cannot analyze it yet. We turn instead to the leaf d, and weight the edge (a, d) with 6 units to satisfy its demand. Now there is only 1 edge left on a, which has 3 units in from

b and 6 units out to d, so to satisfy its demand of 0 the edge (f, a) must have weight 3. Finally, edge (f, c) must then have weight 6 to simultaneously satisfy the remaining node weights on f and c.

Having established this, we must now check for optimality and subsequently choose a step to take, which requires computing the reduced cost for all edges not in the tree - that's the set $\{(a, c), (a, e), (b, c), (b, e), (d, b), (d, e), (f, b), (f, g)\}$. Notice that when computing reduced costs, the cost of the edge you are adding to create the cycle is included. These reduced costs are outlined in the table below:

Edge	Same direction	Opposite direction	Reduced cost
(a, c)	$(a, c), (f, a)$	(f, c)	48+56-108=-4
(a, e)	$(a, e), (g, b), (b, a)$	(g, e)	10+33+7-19=31
(b, c)	$(b, c), (f, a)$	$(f, c), (b, a)$	65+56-108-7=6
(b, e)	$(b, e), (g, b)$	(g, e)	7+33-19=21
(d, b)	$(d, b), (b, a), (a, d)$	N/A	38+7+28=73
(d, e)	$(d, e), (g, b), (b, a), (a, d)$	(g, b)	15+33+7+28-19=64
(f, b)	$(f, b), (b, a)$	(f, a)	48+7-56=-1
(f, g)	$(f, g), (g, b), (b, a)$	(f, a)	24+33+7-56=8

Only two edges, (a, c) and (f, b), have a negative reduced cost. Of these, (a, c) is the most negative, so this is the edge we will add to the spanning tree. There are only 3 edges in the resulting cycle - one is (a, c) itself, and there is only one edge in the opposite direction in this cycle, so we will increase the flow on (a, c) until that edge has been cancelled out, and adjust the flow accordingly on the remaining edge (f, a) accordingly. Thus the new spanning tree will no longer include (f, c), and will have a flow of 6 on (a, c) and a flow of 9 on (f, a) but all other edges will look the same.

We can now check for optimality and select a new step if needed. Note that the only edges that require recalculation are the ones for which (f, c) was previously involved in the cycle; all others will have the same reduced cost. For completeness, the full table is included again below:

Edge	Same direction	Opposite direction	Reduced cost
(a, e)	$(a, e), (g, b), (b, a)$	(g, e)	10+33+7-19=31
(b, c)	(b, c)	$(a, c), (b, a)$	65-48-7=10
(b, e)	$(b, e), (g, b)$	(g, e)	7+33-19=21
(d, b)	$(d, b), (b, a), (a, d)$	N/A	38+7+28=73
(d, e)	$(d, e), (g, b), (b, a), (a, d)$	(g, b)	15+33+7+28-19=64
(f, b)	$(f, b), (b, a)$	(f, a)	48+7-56=-1
(f, c)	(f, c)	$(a, c), (f, a)$	108-48-56=4
(f, g)	$(f, g), (g, b), (b, a)$	(f, a)	24+33+7-56=8

Now there is only one edge - (f, b) - with a negative reduced cost, so we will add it to our spanning tree. Again, the resulting cycle contains only 3 edges, and of those only 1 is in the opposite direction to (f, b). So we increase the flow on the cycle, raising the resulting flow on (f, b) and correspondingly (b, a), until the flow on the opposing edge (f, a) is cancelled out. The result is that (f, b) will have a flow of 9, (b, a) will have a flow of 12, (f, a) is removed from the tree, and the rest of the tree stays the same.

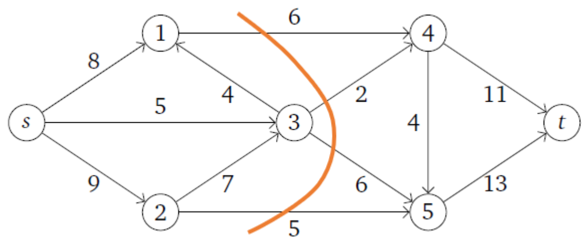
Once again, we check for optimality by evaluating the reduced costs on all edges not currently in the spanning tree, paying particular attention to those which previously would have created cycles involving (f, a). All other edges will have the same reduced cost, included here for completeness:

Edge	Same direction	Opposite direction	Reduced cost
(a, e)	$(a, e), (g, b), (b, a)$	(g, e)	10+33+7-19=31
(b, c)	(b, c)	$(a, c), (b, a)$	65-48-7=10
(b, e)	$(b, e), (g, b)$	(g, e)	7+33-19=21
(d, b)	$(d, b), (b, a), (a, d)$	N/A	38+7+28=73
(d, e)	$(d, e), (g, b), (b, a), (a, d)$	(g, b)	15+33+7+28-19=64
(f, a)	(f, a)	$(b, a), (f, b)$	56-7-48=1
(f, c)	(f, c)	$(a, c), (b, a), (f, b)$	108-48-7-48=5
(f, g)	$(f, g), (g, b)$	(f, b)	24+33-48=9

Now all reduced costs are positive, indicating that this spanning tree is optimal. Therefore the optimal spanning tree to respect the demands of each node consists of the edges $\{(a, c), (a, d), (b, a), (f, b), (g, b), (g, e)\}$, with flows of 6, 6, 12, 9, 3, and 2 respectively.

5 Max Flow and Min Cut

In the solution for this problem, we will present the first $s - t$ path and its residual, and then for every subsequent path we will present the path, the augmented graph, and the residual, until the residual reveals that there are no more $s - t$ paths. Note that on the residual graphs, reversed edges are drawn in green and edges pointing in their original direction are drawn in red, but both green and red edges can be utilized to construct a viable $s - t$ path so long as they are utilized in the direction they point. This process will ultimately indicate that the max flow is 19, with the following corresponding min cut:



We begin with a single $s-t$ path and its residual. Note that the first path is its own augmented flow graph, so there is no need to redraw it.

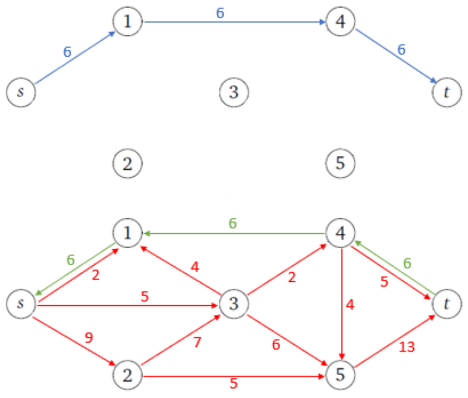


Figure 1: Selection of first $s-t$ path (blue for flow) and resulting residual graph. Path: $s \to 1 \to 4 \to t$

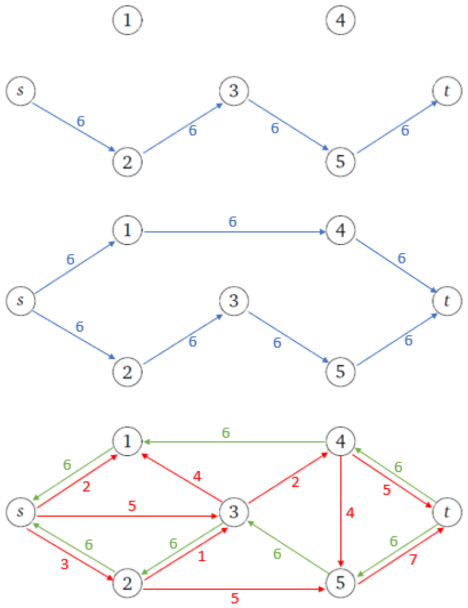


Figure 2: Selection of second $s-t$ path, resulting augmented flow graph, and residual of augmented flow graph. Path: $s \to 2 \to 3 \to 5 \to t$

For the third step, we use both red and green edges to construct an $s-t$ path - this is the benefit of using the residual approach.

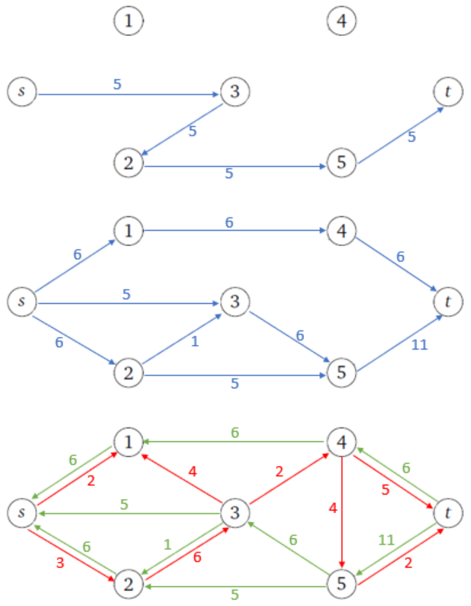


Figure 3: Selection of third $s - t$ path, resulting augmented flow graph, and residual of augmented flow graph. Path: $s \to 3 \to 2 \to 5 \to t$

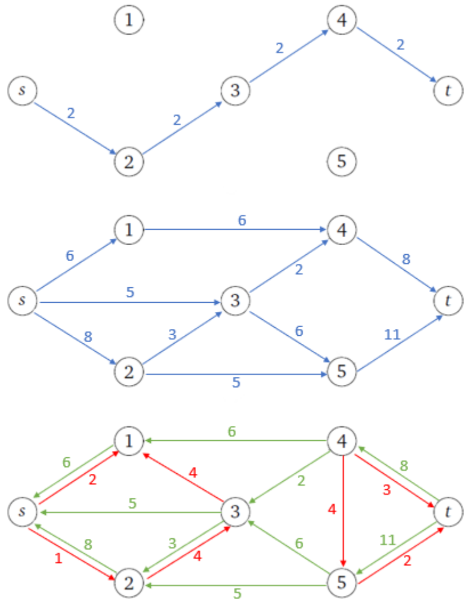


Figure 4: Selection of fourth $s - t$ path, resulting augmented flow graph, and residual of augmented flow graph. Path: $s \to 2 \to 3 \to 4 \to t$

After this fourth stage, there is no longer any way to follow the directed arrows to get an $s - t$ path. The maximum flow is a total of 19 units along the paths indicated in the augmented flow graph, which is included separately here for ease of viewing:

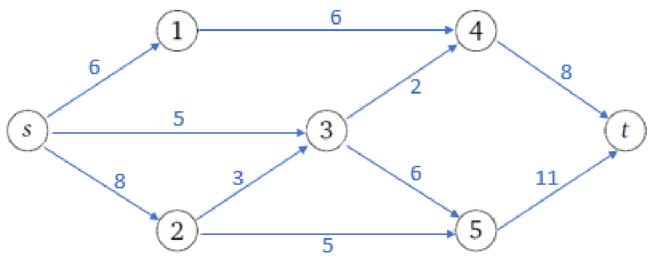


Figure 5: Augmented flow graph from fourth stage, which gives the max flow through the network

Note that in the final residual graph, five of the edges have maximal forward flow on them (indicated by the absence of a red edge for more available forward units). Four of these edges are the edges that are cut in the minimum-weight cut indicated at the beginning of this problem. This is always true - the edges removed to create a min cut are always edges on which there is maximal flow in the max flow solution. In this case, these edges are $\{(1,4),(2,5),(3,4),(3,5)\}$ for a cut such that the sets of vertices are $\{s,1,2,3\}$ and $\{4,5,t\}$.

6 Dynamic Programming

This problem will be a 3-stage problem, where at each stage we consider how many funding blocks to assign to each project. (A slight apologetic disclaimer: the hint was supposed to indicate that the values at each stage needed to represent the remaining available funds, but was poorly written.) By the end we can reasonably assume we should have assigned all four blocks of $5,000 in funds, so as we go along we want to keep track of how many blocks have been assigned so far. We can think of this as a network, where each stage has vertices representing 0-4 blocks of funding remaining for assignment, and the edges coming into vertices in a stage indicate how many blocks of funding are being assigned to that project. Note that the final stage will have 5 vertices but edges should only come into the vertex representing 0 blocks remaining, since we should assign all 4 blocks by the time the program is complete. We should also have a zero-stage (starting vertex) indicating that 4 blocks are available. The network looks like this, where all edges should be followed to the right:

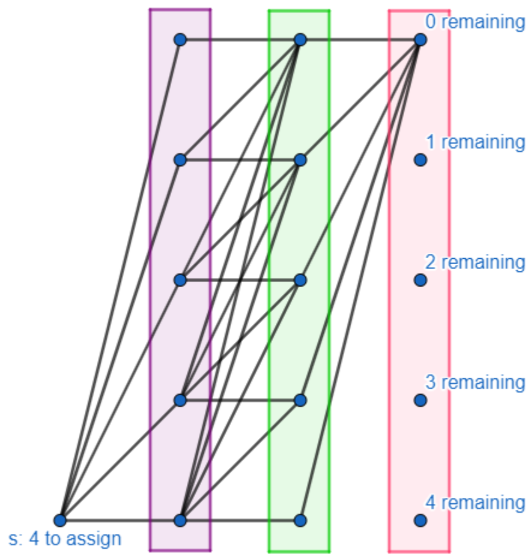


Figure 6: Dynamic programming network. Project 1 is the purple box, Project 2 is the green box, Project 3 is the pink box.

Each edge indicates the number of funding blocks being assigned to the next project from the available pool. For convenience in constructing tables, we will label these vertices (other than s) as $P1_i, P2_i, P3_i$ where $i = 0, 1, 2, 3, 4$. So for example the ending vertex is $P3_0$. The value carried through the network is the total number of hours saved through the funding allocation, so our goal is the following:

$$f(P3_0) = \max(f(P2_0), f(P2_1) + 3, f(P2_2) + 6, f(P2_3) + 8, f(P2_4) + 9)$$

Note that the quantities being added to each recursive function call are the amounts of time that would be saved by allocating 0, 1, 2, 3, or 4 blocks of funding to Project 3, respectively, corresponding to the number of available blocks remaining once Project 1 and Project 2 have been allocated. Following this paradigm, we begin to construct our tables. A dash will indicate that no route exists between a particular starting vertex and route option.

Stage 3:

Vertex	$t = P3_0$	Max savings	Best route
$P2_0$	0	0	$P2_0 \to t$
$P2_1$	3	3	$P2_1 \to t$
$P2_2$	6	6	$P2_2 \to t$
$P2_3$	8	8	$P2_3 \to t$
$P2_4$	9	9	$P2_4 \to t$

Stage 2:

Vertex	$P2_0 \to t$	$P2_1 \to t$	$P2_2 \to t$	$P2_3 \to t$	$P2_4 \to t$	Max savings	Best route
$P1_0$	0+0	-	-	-	-	0	$P1_0 \to P2_0$
$P1_1$	4+0	0+3	-	-	-	4	$P1_1 \to P2_0$
$P1_2$	7+0	4+3	0+6	-	-	7	$P1_2 \to P2_0, P2_1$
$P1_3$	9+0	7+3	4+6	0+8	-	10	$P1_3 \to P2_1, P2_2$
$P1_4$	11+0	9+3	7+6	4+8	0+9	13	$P1_4 \to P2_2$

Stage 1:

Vertex	$P1_0 \to t$	$P1_1 \to t$	$P1_2 \to t$	$P1_3 \to t$	$P1_4 \to t$	Max savings	Best route
s	8+0	7+4	5+7	3+10	0+13	13	$s \to P1_3, P1_4$

According to these, the maximum savings is 13 hours and can be achieved in 3 ways. First, one could assign no funding to Project 1, and 2 blocks of funding each to Project 2 and Project 3. Second, one could assign 1 block of funding to Project 1, 2 blocks to Project 2, and 1 block to Project 3. Or third, one could assign 1 block of funding to Project 1, 1 block to Project 2, and 2 blocks to Project 3. Any of these paths will give the optimal result.