Network Dynamics and Learning - Homework #1

Hafez Ghaemi - hafez.ghaemi@studenti.polito.it

November 10, 2021

1 Exercise 1

In this exercise, we will analyse the effect of link capacities on the maximum unitary flow throughput from a source node o to a destination node d in a simple directed graph. The graph is drawn in Figure 1.

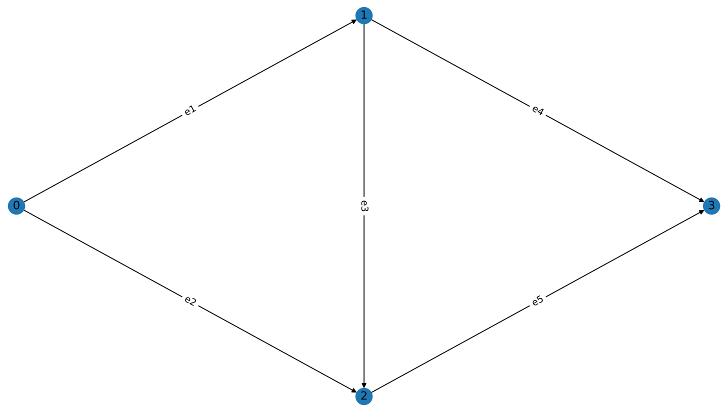

Figure 1: The graph to be analyzed

1.1 Part a

Assuming that link e_i has capacity c_i, we can state the network resilience interpretation of the max-flow min-cut theorem: The minimum (here also infimum) total capacity to be removed from the network to make d not reachable from o coincides with the min-cut capacity $C^*_{(o,d)}$. Therefore, we need to find the min-cut capacity, and remove the capacities from the path that has this capacity to achieve this goal. From theory we know that:

$$C^*_{(o,d)} = \min_{u \subseteq \nu, o \in u, d \notin u} C_u \tag{1}$$

where C_u is defined as:

$$C_u := \sum_{i \in u} \sum_{j \in \nu \setminus u} C_{ij} = 1 \tag{2}$$

The graph in this exercise has four possible cuts, and therefore, we need to remove the minimum among the capacities of these four cuts that is also the maximal throughput from o to d:

$$C_{del} = C^*_{(o,d)} = \min\{c_1 + c_2, c_1 + c_5, c_4 + c_5, c_2 + c_3 + c_4\} \tag{3}$$

1.2 Part b

Using the max-flow min-cut theorem, in this case equation 3, we can calculate the current maximum throughput based on $C = [3, 2, 2, 3, 2]$, that is $c_1 + c_2 = 5$. Now, we have one additional unit of capacity that we can add to any other link. By considering all possibilities (adding this unit to any link), we conclude that adding this additional unit to any other link, does not change the minimum value obtained in equation 3. Therefore, the maximum throughput remains the same, i.e., 5 whichever link we choose for adding the additional unit of capacity.

1.3 Part c

Similar to part b, we will try all combinations of adding two additional units of capacity to the links (one unit to two different nodes, or two units to one node). This time we observe that the maximum flow will increase to 6. The new optimal capacity vector is $C = [4, 2, 2, 4, 2]$. So, the additional units should be added to links e_1 and e_4. All cuts will have the same capacity of 6 after this modification.

1.4 Part d

Similar to the previous parts, we will try all combinations of adding four additional units of capacity to the links (These combinations are $(1, 1, 1, 1)$, (4), $(2, 2)$, and $(1, 2, 1)$ with different permutations to be added to link capacities). We observe that the maximum flow will increase to 7. The new optimal capacity vector is $C = [4, 3, 2, 4, 3]$ which means that the best combination is $(1, 1, 1, 1)$ and one additional unit should be added to links e_1, e_2, e_4, and e_5. The cuts with minimum capacities after this modification will be $c_1 + c_2$ and $c_4 + c_5$.

2 Exercise 2

In this exercise we will consider the matching problem and the application of Hall's theorem and Ford-Fulkerson algorithm in bipartite graphs.

2.1 Part a

The interest pattern can be visualized using a bipartite graph in Figure2. The left part represents persons, and the right part represents books.

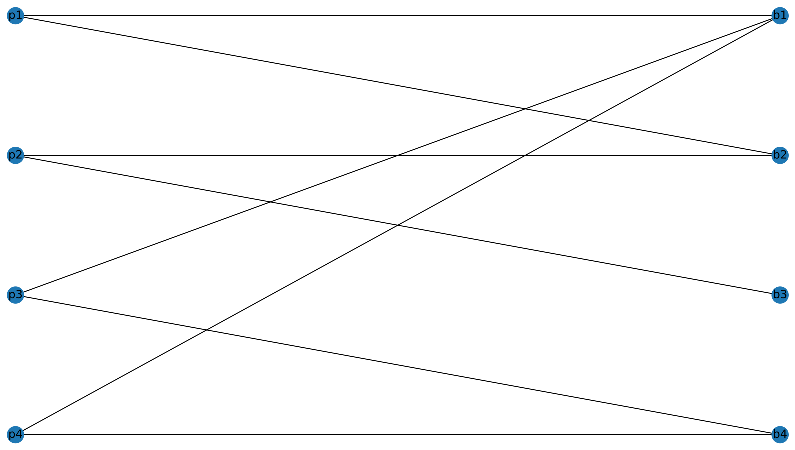

Figure 2: The bipartite graph visualizing the interest pattern

2.2 Part b

In order to find out if a perfect matching exists or not, we will create the auxiliary graph given in Figure 3 and solve the max-flow problem for the flow from o to d. Since one copy of each book exists, in this part, all links from o to persons, from persons to books, and from books to d have capacity 1. From theory, we know that if the Hall theorem's conditions were satisfied and a perfect matching existed, the Ford-Fulkerson

(FF) algorithm would yield a maximum flow equal to n (number of persons), and the path found would be the desired matching between persons and books.

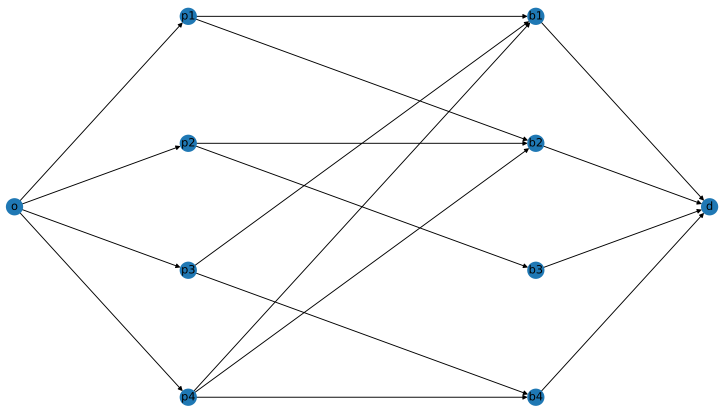

Figure 3: The auxiliary graph used for finding a perfect matching

By running FF on this graph we observe that indeed, the maximum flow is four and the Hall conditions are satisfied. The perfect matching is:

$\{p1 : b2, p2 : b3, p3 : b1, p4 : b4\}$

2.3 Part c

To find the maximum number of books assigned and the corresponding assignment, we, again, use the auxiliary graph in Figure 3. This time, we assign capacity equal to n to links from o to persons, and capacity 1 to to the middle nodes because each person can only take a maximum of one copy from each book. For links from books to d we use the number of copies for each book from the given vector $[2, 3, 2, 2]$, and this part is the bottleneck of our network. Now, using the FF algorithm we can find the maximum flow from o to d which corresponds to the maximum number of books that can be assigned. The path found by FF presents this optimal assignment of the books. Using the given vector, the total number of 8 books could be allocated in the following manner:

$\{p1 : [b1 : 0, b2 : 1], p2 : [b2 : 1, b3 : 1], p3 : [b1 : 1, b4 : 1], p4 : [b1 : 1, b2 : 1, b4 : 1]\}$

2.4 Part d

In order to find the optimal sell/buy decision for maximizing the number of assigned books, we iterate over every possibility of selling one copy of a book and buying a copy of another, and then run the FF algorithm on the corresponding auxiliary graph that is formed in a fashion similar to Part c to find the maximum flow and the path for each possibility, and then choose the possibility with the highest maximum flow. The optimal decision is to sell a copy of b_3 and buy a copy of b_1. By taking this decision, the total number of 9 books (all books) could be allocated in the following manner:

$\{p1 : [b1 : 1, b2 : 1], p2 : [b2 : 1, b3 : 1], p3 : [b1 : 1, b4 : 1], p4 : [b1 : 1, b2 : 1, b4 : 1]\}$

3 Exercise 3

After loading the given $.mat$ files using the commands given, we can create the graph by utilizing the concept of link-incidence matrix B. For each link (edge) we are able to determine the head and tail by finding the indices that are equal to 1 or -1. Furthermore, we set the capacity and travel time for each link as its attributes. Now the complete graph of the highway map is constructed. The map schematic can be seen in Figure 4.

3.1 Part a

We can find the shortest path between node 1 and 17 using the travel time of each link as weights. The computed path is:

$[1, 2, 3, 9, 13, 17]$

and you by computing the sum of weights, we see that travelling on this path takes 0.533 hour.

3.2 Part b

By considering the capacities, and running the Ford-Fulkerson algorithm, we can calculate the maximum flow from between nodes 1 and 17 along the path. The maximum flow is 22448, and this flow should be transported in the following manner (the flows going out from each node's out-links are given and you can follow this path on the highway map in Figure 4).

$\{1 : \{2 : 8741, 6 : 13707\}, 2 : \{3 : 8741, 7 : 0\}, 3 : \{4 : 0, 8 : 0, 9 : 8741\}, 4 : \{5 : 0, 9 : 0\}, 5 : \{14 : 0\}, 6 : \{7 : 4624, 10 : 9083\}, 7 : \{8 : 4624, 10 : 0\}, 8 : \{9 : 4624, 11 : 0\}, 9 : \{13 : 6297, 12 : 7068\}, 13 : \{14 : 3835, 17 : 10355\}, 14 : \{17 : 3835\}, 10 : \{11 : 825, 15 : 8258\}, 11 : \{12 : 825, 15 : 0\}, 15 : \{16 : 8258\}, 12 : \{13 : 7893\}, 17 : \{\}, 16 : \{17 : 8258\}\}$

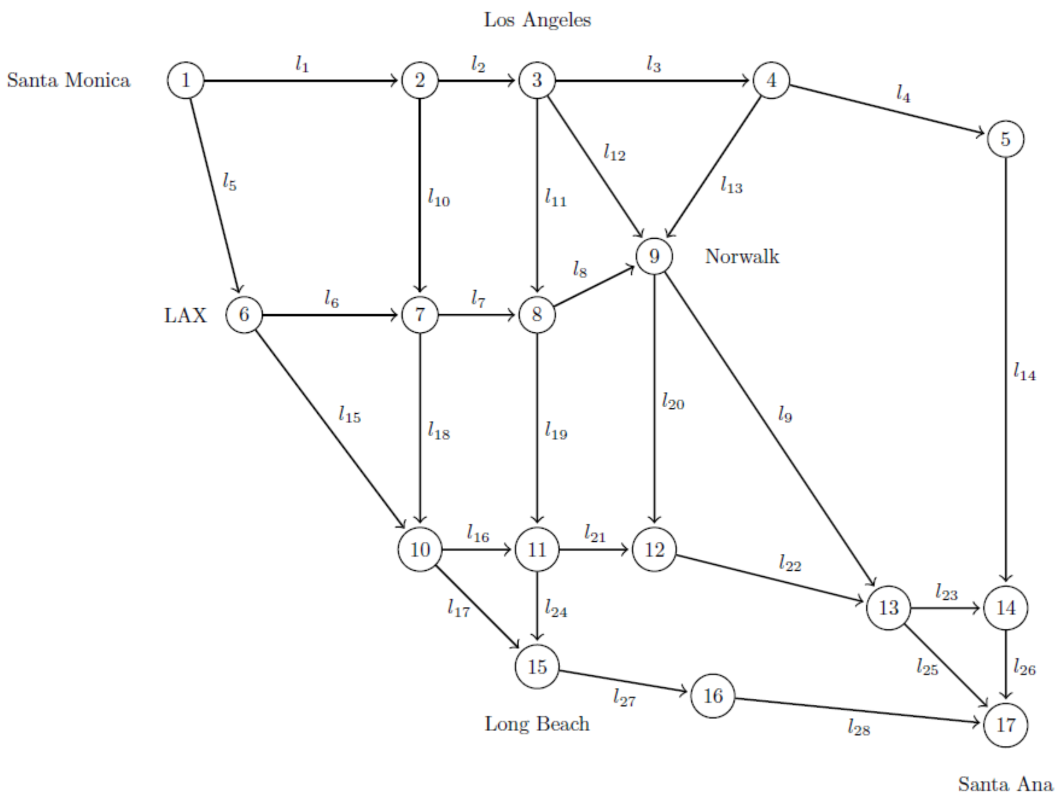

Figure 4: The schematic of the highway network of Los Angles

3.3 Part c

Using a simple matrix multiplication, the exogenous flow for each node using the given flow vector from equation4.

$$Bf = v \tag{4}$$

Doing so would yield the following vector for exogenous flows for nodes 1 to 17, respectively (negative values indicated in-flow and positive ones indicate out-flow):

$[16806, 8570, 19448, 4957, -746, 4768, 413, -2, -5671,$

$1169, -5, -7131, -380, -7412, -7810, -3430, -23544]$

From now on, we assume that all nodes have zero in-flow except nodes 1 and 17 and the value for v_1 is the one computed above, and $v_{17} = -v_1$.

3.4 Part d

From theory we know that since we have:

$$\Sigma_{i\in u} v_i < \Sigma_{e\in E:\theta(e)\in u,\kappa(e)\in\nu\setminus u} c_e \tag{5}$$

for every $u \subsetneq \nu$ that $\Sigma_{i\in u}\nu_i > 0$, the social optimum problem has a feasible solution that can be found by solving the optimization problem in equation 6 using Lagrangian multipliers.

$$M(\nu) := \min_{f\geq 0, Bf=\nu} \Sigma_{e\in E}\Psi_e(f_e) \tag{6}$$

In this case, the cost function is defined in equation 7.

$$\Sigma_{e\in E}\Psi_e(f_e) = \Sigma_{e\in E} f_e d_e(f_e) = \Sigma_{e\in E}\frac{f_e l_e}{1 - f_e/C_e} = \Sigma_{e\in E}(\frac{l_e c_e}{1 - f_e/C_e} - l_e C_e) \tag{7}$$

Solving this optimization problem (subject to the flow constraints) yields the total cost (delay) of 25943.6 for the social optimum and the following flow vector:

$[6642.2, 6058.9, 3132.3, 3132.3, 10163.8, 4638.3, 3006.3, 2542.6, 3131.5, 583.3, 0., 2926.6, 0., 3132.3,$

$5525.5, 2854.3, 4886.4, 2215.2, 463.7, 2337.7, 3318., 5655.7, 2373.1, 0., 6414.1, 5505.4, 4886.5, 4886.5]$

3.5 Part e.1

Similarly we can dins the Wardrop equilibrium (when drivers choose their route analogous to an anarchist community) by minimizing the cost function in equation 8 subject to the flow constraints.

$$\Sigma_{e\in E} e(f_e) = \Sigma_{e\in E}\int_0^{f_e} d_e(s)ds \tag{8}$$

Solving would yield the total delay of 26293 (note that we use $\Sigma_{e\in E} f_e d_e(f_e)$ for calculating the total delay, so the cost can be comparable for the social optimum and the Wardrop equilibrium). The resulting flow vector for Wardrop equilibrium is:

$[6715.6, 6715.6, 2367.4, 2367.4, 10090.4, 4645.4, 2803.8, 2283.6, 3418.5, 0, 176.8, 4171.4, 0, 2367.4, 5445, 4171.4,$

$0, 2367.4, 5445, 2353.2, 4933.3, 1841.6, 697.1, 3036.5, 3050.3, 6086.8, 2586.5, 0, 6918.7, 4953.9, 4933.3, 4933.3]$

The price of anarchy can be calculated by equation 9.

$$PoA = \frac{\text{total delay at Wardrop}}{\text{Minimum possible total delay (social optimum)}} \tag{9}$$

In our case PoA will be 1.013 which means that letting drivers decide their route (an anrchist community) does not impose too much additional cost on the society.

3.6 Part e.2

From theory, we know that to design tolls that would make price of anarchy equal to one, or in other words the total delay at Wardrop equal to minimum possible delay, we have to define the toll as equation 10.

$$\omega_e = f_e^* d_e'(f_e^*) \tag{10}$$

where d_e' is the derivative of the delay function which results in $d_e'(f_e) = \dfrac{C_e l_e}{(f_e - C_e)^2}$. Now, by setting the new delay function $d_e(f_e) + \omega_e$ inside the integral in equation 8, we can calculate the new Wardrop equilibrium with tolls. This new equilibrium results in the total delay of 25943.6 and the flow vector below:

$[6643., 6059.1, 3132.5, 3132.5, 10163., 4638.3, 3006.3, 2542.3, 3131.5, 583.9, 0., 2926.6, 0., 3132.5,$

$5524.8, 2854.2, 4886.4, 2215.8, 464., 2337.5, 3318.2, 5655.7, 2373., 0., 6414.1, 5505.5, 4886.4, 4886.4]$

We can observe that the theory is validated. The costs and the flows (except for small computational rounding errors) are equal for the social optimum and the Wardrop equilibrium with tolls. Also, because of this fact, the price of anarchy is equal to 1 for this case.

3.7 Part f

The new cost function to be replaced into equation 5 is going to be equation 11.

$$c_e(f_e) = f_e(d_e(f_e) - l_e) \tag{11}$$

Consequently, the new optimal cost function with tolls to make Wardrop equilibrium equal to the social optimum is given in equation 12. The toll function is the same as before, but the function under the integral should follow the new delay function in equation 11.

$$\sum_{e \in E} \int_0^{f_e} (d_e(s) - l_e) ds + f_e \omega_e^* \tag{12}$$

By minimizing this cost function, we can calculate the Wardrop equilibrium. Comparing these flows with the ones from social optimum calculated minimizing the cost function formed by equation 11, again we can validate the theory and reach the same total cost (using the new cost function) of 15095.51 and the flow vector below:

$$[6653.1, 5775.4, 3419.5, 3419.5, 10152.9, 4642.4, 3105.5, 2661.7, 3009.2, 877.7, 0., 2355.9, 0, 3419.5, 5510.4,$$

$$3043.4, 4881.7, 2414.6, 3043.4, 4881.7, 2414.6443.8, 2008.5, 3487.1, 5495.6, 2204.1, 0,$$

$$6300.7, 5623.5, 4881.7, 4881.7]$$

4 Notes

For most of the graph operations in this assignment, the Python package NetworkX (Hagberg, Swart, & S Chult, 2008) was used. Furthermore, for optimizing cost functions in exercise 3, the CVXPY package (Agrawal, Verschueren, Diamond, & Boyd, 2018; Diamond & Boyd, 2016) was used.

I have exchanged some theoretical ideas in parts $2.c$ and $2.d.$ and some code-related ideas with student Hajali Bayramov.

References

Agrawal, A., Verschueren, R., Diamond, S., & Boyd, S. (2018). A rewriting system for convex optimization problems. *Journal of Control and Decision*, *5*(1), 42–60.

Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, *17*(83), 1–5.

Hagberg, A., Swart, P., & S Chult, D. (2008). *Exploring network structure, dynamics, and function using networkx* (Tech. Rep.). Los Alamos National Lab.(LANL), Los Alamos, NM (United States).