Network Dynamics and Learning Homework I

Shannon Mc Mahon
Student id: s289958

***Abstract*—The following document contains the discussion of the solutions of the assigned exercises.**
Solution discussed with Andrea Rubeis (s290216).
Python implementation is available on *Github*.

I.

Consider unitary o-d network flows (i.e., integer flows whose integer units cannot be split) on the graph G = (V, E) in Figure I, and assume that each link has integer capacity.

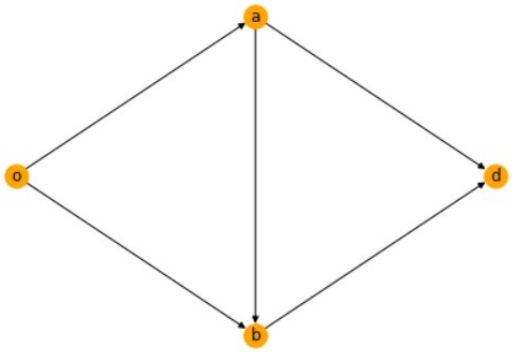

Fig. 1.

(a) The infimum of the total capacity that needs to be removed for no feasible unitary flows from o to d to exist is, according to Menger's Theoreom, the capacity of the min-cut, so in this case 2.

(b) Assume that the link capacities are

$$C_{(o,a)} = C_{(a,d)} = 3, \quad C_{(o,b)} = C_{(a,b)} = C_{(b,d)} = 2$$

Then to decide where 1 unit of additional capacity should be allocated to maximize the feasible throughput from o to d we compute the the capacity C_U of all possible o-d cuts U in G. For commodity we also report E_U which contains the edges from U to U_c (used to calculate the capacity of the cut).
From table I we can see that the min-cut capacity is 5, and there are three cuts with such capacity. When deciding where to add additional capacity to maximise the flow from 0 to d the best choice consists in allocating capacity to edges that are common to multiple cuts (for example edge (o,a) is common to the first and third partition).
However, since we are requested to add only one unit of capacity this is not sufficient to increment the maximal throughput, which remains 5 (for example if we added one unit to (o,a), edge that is common to the 1st and 3rd partition, the capacities of the cuts would become, ordered as in the table, (6,7,6,5). Then, from the theory we know the maximum throughput is equal to the min-cut capacity, which is 5 and as such has not improved. In other words there will always be one of the min-cuts whose capacity will remain 5).

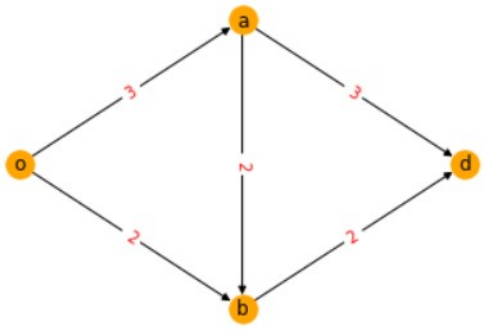

Fig. 2. Graph G with labels corresponding to the capacities of the edges.

U	U^c	E_U	C_U
o	a,b,d	(o,a),(o,b)	5
o,a	b,d	(o,b),(a,b),(a,d)	7
o,b	a,d	(o,a),(b,d)	5
o,a,b	d	(a,d),(b,d)	5

TABLE I
ALL O-D CUTS U OF GRAPH G.

(c) If instead we are allowed to allocate two units of capacity it is possible to increase the maximum throughput. Once again following the idea of adding capacity to the common edges we find that the following combinations lead to a throughput of 6:

- +1 unit on (o,a), +1 unit on (a,d)
- +1 unit on (o,a), +1 unit on (b,d)
- +1 unit on (o,b), +1 unit on (b,d)

To better understand what it means to target common edges consider for example what would happen if we were to add 1 unit to (o,b) and 1 to (a,d): (o.b) is shared by the 1st and 2nd cut, while (a,d) is shared by by the 2nd and 4th. The 3rd cut is not "covered" by the addition of these two units, and as such its capacity remains equal to

5, and in turn the maximum throughput does not increase.

(d) Following the above reasoning we can find were to allocate 4 additional units of capacity (always starting from the initial state of G):

- +2 units on (o,a), +2 units on (a,d)
- +2 units on (o,a), +2 units on (b,d)
- +2 units on (o,b), +2 units on (b,d)
- +2 units on (o,a), +1 unit on (a,d), +1 unit (b,d)
- +2 units on (b,d), +1 unit on (o,a), +1 unit on (o,b)
- +1 unit on (o,a), +1 unit on (o,b), +1 unit on (a,d), +1 unit on (b,d)

In all these cases the maximum throughput is equal to 7, with sum of cut capacities equal to 30.

II.

Consider a set of people (p1, p2, p3, p4) and a set of books (b1, b2, b3, b4). Each person is interested in a subset of books, specifically:

$$p1 \to (b1, b2),\ p2 \to (b2, b3)$$

$$p3 \to (b1, b4),\ p4 \to (b1, b2, b4)$$

(a) We can represent the interest pattern by using a simple bipartite graph G as seen in the figure below, where the two disjoint and independent sets are $P = \{p1, p2, p3, p4\}$ and $B = \{b1, b2, b3, b4\}$

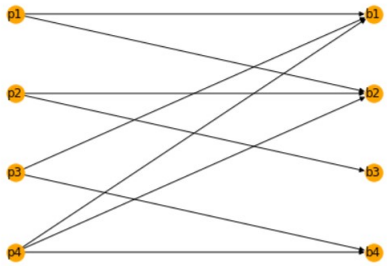

Fig. 3. Bipartite representation of the problem: on the left the people set P, on the right the book set B.

(b) We can establish whether there exists a perfect matching that assigns a book of interest to every person by making use of the max-flow algorithm. We firstly modify our graph as follows:

- Introduce an origin node *o* and a destination node *d*
- $\forall p \in P$ add an edge *(o,p)*, with capacity 1
- $\forall b \in B$ add an edge *(b,d)*, with capacity 1
- Fix capacity equal to 1 for edges from *P* to *B*

Figure 4 shows the resulting graph G1.

From the theory we know that a P-perfect matching on G1 exists if and only if it there exists a flow with throughput equal to $|P|$ on the network. To check if such condition is satisfied we can use the max-flow algorithm offered by NetworkX. This allows us to determine that the maximum flow that can be sent on G is in fact $|P| = 4$. Indeed a perfect matching exists, as Figure 5 illustrates.

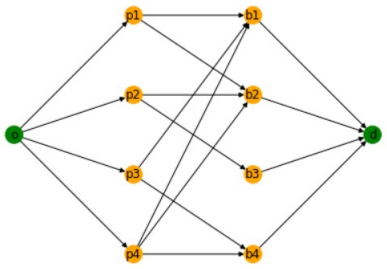

Fig. 4. Modified graph G1.

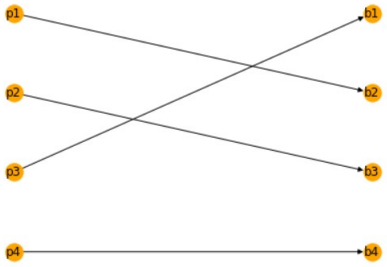

Fig. 5. Perfect matching.

(c) We now assume that there are multiple copies of books, specifically the distribution of the number of copies is (2, 3, 2, 2), and there is no constraint on the number of books that each person can take. However, each person cannot take multiple copies of the same book. Our objective is to determine how many books of interest can be assigned in total.

Once again we use the analogy with max-flow problems to answer this question assigned in total.

Starting from graph G1, we make a few modifications:

- $\forall p \in P$ set the capacity of edge *(o,p)* equal to the number of books p is interested in. For example, edge *(o,p1)* will have capacity 2 since p1 is interested in books $b1$ and $b2$. The capacity here in other words represents the maximum number of books that each person can aspire to get (i.e. he gets all the books he is interested in).
- $\forall b \in B$ set the capacity of edge *(b,d)* equal to the number of copies available of such book. For example, edge *(b1,d)* will have capacity equal to 2.
- Leave the capacities equal to 1 for edges from *P* to *B*. This ensures that each person cannot take more than one copy of a given book.

By then applying the max-flow algorithm we can determine the maximum flow, or equivalently how many books of interest can be assigned in total. As can be seen in graph G2 in figure 6, a total of 8 books are assigned (1 to p1, 2 to p2, 2 to p3, 3 to p4).

(d) Finally we suppose that the library can sell a copy of a book and buy a copy of another book. We wish to

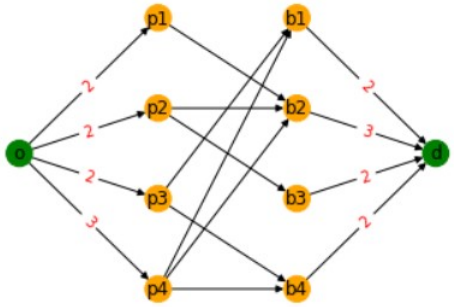

Fig. 6. Graph G2 (only the edges through which flow passes have been represented. The numbers in red represent the capacity of the arcs).

determine which books should be sold and bought to maximize the number of assigned books.
Intuitively to maximize the number of assigned books one needs to satisfy the interests of the readers. In other words we would ideally want there to be a number of copies of each book (capacities of arcs from B set to node d) equal to the demand that there is for each book (the in-degree of nodes in B). The in-degree of nodes $b1, b2, b3, b4$ are respectively $3, 3, 1, 2$. If we compare this to the number of available copies of each of these books $2, 3, 2, 2$ we can see that it is sufficient to sell 1 copy of b3 and buy one of b1. This leads to in increase in the number of assignments from 8 to 9.
Figure 7 shows the resulting Graph G3 of assignments.

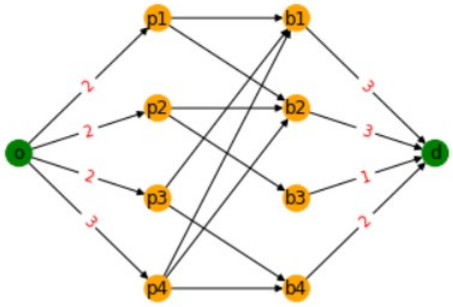

Fig. 7. Graph G3 (only the edges through which flow passes have been represented. The numbers in red represent the capacity of the arcs).

III.

Before solving this exercise we recall some of the theory concepts necessary to understand the resolution.
. Recall that SO-TAP (System Optimum Traffic Assignment Problem) deals with modelling congested transportation networks. Given a network we associate to each link e a cost $\psi_e(f_e) = f_e d_e(f_e)$, equal to the product of the flow f_e times the delay d_e, which can be interpreted as the total delay on link e.

An important family of delay functions, which corresponds to the one used in this exercise, is given by

$$d_e(f_e) = \begin{cases} \frac{l_e}{1-f_e/c_e}, & \text{if } 0 \le f_e < c_e \\ +\infty, & \text{if } f_e \ge c_e \end{cases} \tag{1}$$

where $l_e = d_e(0)$ and $c_e = sup\{x \ge 0 : d_e(x) < +\infty\}$ is the flow capacity on link e. Notice how the delay function d_e is convex, non decreasing and differentiable (this will be useful later on).
The System-Optimum Traffic Assignment Problem consists of the following network flow optimization problem,

$$\begin{aligned} & inf \sum_{e \in \varepsilon} \psi_e(f_e) \\ & f \in \mathbb{R}_+^{\varepsilon} \\ & Bf = v \end{aligned} \tag{2}$$

in which the linear equality constraints correspond to mass conservation at the nodes, while the linear inequality constraints correspond to non negativity of the flow variables.
The flow distribution emerging from this centralized decision scheme is the social optimum flow distribution. By contrast, we introduce the concept of Wardrop Equilibrium, which is the outcome of selfish behaviors of users. Such behavior is modeled by assuming that users choose their route so as to minimize the delay they experience along it.
The flow vector $f^{(0)}$ corresponding to a Wardrop equilibrium can be obtained as a solution of a network flow optimization given that the cost functions $\psi(f_e)$ are chosen as $\psi_e(f_e) = \int_0^{f_e} d_e(s)\, ds$. The flow distribution emerging from this uncoordinated selfish behaviour has every user chose a path with minimal delay.
Having now briefly recapped the main concepts we can move on to solve the exercise.

We are given the highway network in Los Angeles. The node-link incidence matrix B for this traffic network is given in the file traffic.mat, where the rows of B are associated with the nodes of the network and the columns of B with the links. Each link e_i has a maximum flow capacity c_{ei}. The capacities are given as a vector in the file capacities.mat. In a similar manner the minimum travelling times are given as a vector l_e in the file traveltime.mat.

(a) To find the shortest path between node 1 and 17 (or equivalently the fastest path) in an empty network we firstly construct the Graph of the highway network. The node-link incidence matrix allows us to define the edges of the graph, while we use the remaining files to endow each edge with the attributes flow, traveltime, capacity and length (found by multiplying the traveltime by the assumed speed limit 60 miles/hour). At this point we note that finding the shortest path coincides with solving problem (2) with $\psi(f_e) = l_e f_e$ and $v = (1, 0, 0, ..., 0, 0, -1)$. However the easiest way to compute the shortest path is by making use of the NetworkX shortest path function. This shows us that the shortest path is [1, 2, 3, 9, 13, 17]

with length 31.97 miles and total traveltime 0.53 hours. Figure 8 shows the highway network with labels corresponding to the traveltime.

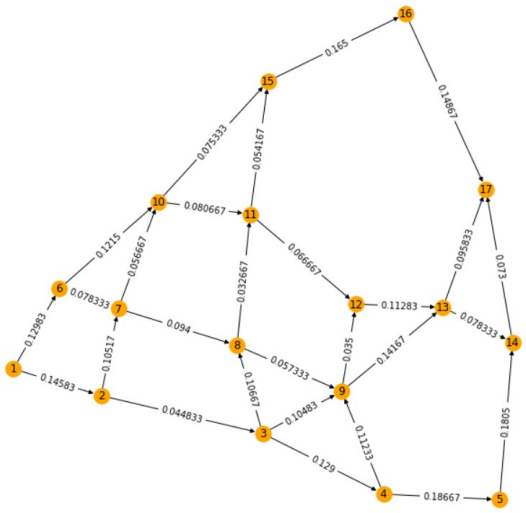

Fig. 8. Los Angeles highway network with labels corresponding to the traveltime for the link.

(b) To find the maximum flow between node 1 and 17 we can once more make use of the already implemented NetworkX function max-flow. We find that the maximum flow between node 1 (origin) and 17 (destination) is 22448.

(c) To compute the external inflow v satisfying the mass conservation at the nodes given the flow vector in flow.mat, it is sufficient to compute Bf. We find that v = [16806, 8570 19448, 4957, -746, 4768, 413, -2, -5671, 1169, -5, -7131, -380, -7412, -7810, -3430, -23544]. In the following points we assume that the exogenous inflow is zero in all the nodes except for node 1, for which v_1 has the same value computed in the point (c), and node 17, for which $v_{17} = -v_1$.

(d) Finding the social optimum $f*$ with respect to the delays on the different links $d_e(f_e)$ means solving problem (2) where the function to be minimised is the following:

$$\sum_{e\in\varepsilon}\psi_e(f_e) = \sum_{e\in\varepsilon} f_e d_e(f_e) = \sum_{e\in\varepsilon}\left(\frac{l_e c_e}{1-f_e/c_e} - l_e c_e\right)$$

We find that $f*$ = [6642.3, 6058.9, 3132.4, 3132.4, 10163.7, 4638.4, 3006.36, 2542.59, 3131.52, 583.4, 0, 2926.5, 0, 3132.4, 5525.3, 2854.3, 4886.44, 2215.44, 463.78, 2337.56, 3318.08, 5655.64, 2373.04, 0, 6414.12, 5505.44, 4886.44, 4886.44], while the associated cost is 25943.62. The results have been rounded to two decimals, just as the ones in the following points will be.

(e) We now proceed to Find the Wardrop equilibrium. To do so we need to minimise the following function:

$$\sum_{e\in\varepsilon}\psi_e(f_e) = \sum_{e\in\varepsilon}\int_0^{f_e} d_e(s)\,ds = -\sum_{e\in\varepsilon} l_e c_e \log\left(1-\frac{f_e}{c_e}\right)$$

We obtain $f^{(0)}$ = [6715.65, 6715.65, 2367.41, 2367.41, 10090.35, 4645.39, 2803.84, 2283.56, 3418.48, 0, 176.83, 4171.41, 0, 2367.41, 5444.96, 2353.17, 4933.34, 1841.55, 697.11, 3036.49, 3050.28, 6086.77, 2586.51, 0, 6918.74, 4953.92, 4933.34, 4933.34] with relative cost 26292.96. As we would expect the cost at Wardrop equilibrium exceeds that at social optimum, reflecting the "selfish" behaviour of users.
We proceed and introduce tolls w_e on each link, specifically $w_e = f_e^* d_e^{'}(f_e^*)$, where f_e^* is the flow at the system optimum found at point d). This particular choice of weights has a very interesting consequence discussed in point f). The new Wardrop equilibrium is then obtained by minimising

$$\sum_{e\in\varepsilon}\psi_e(f_e) = \sum_{e\in\varepsilon}\int_0^{f_e}(d_e(s)+w_e(s))\,ds = \sum_{e\in\varepsilon}(f_e^*\frac{l_e c_e}{(c_e-f_e^*)^2} f_e - l_e c_e \log\left(1-\frac{f_e}{c_e}\right)$$

From this we find $f^{(w)}$ = []6642.3, 6059.07, 3132.3, 3132.3, 10163.7, 4638.01, 3006.25, 2542.45, 3131.54, 583.23, 0, 2926.77, 0, 3132.3, 5525.68, 2854.25, 4886.43, 2215, 463.8, 2337.68, 3318.06, 5655.73, 2373.17, 0, 6414.11, 5505.46, 4886.43, 4886.43].
The associated perceived cost by users can be computes as follows:

$$\sum_{e\in\varepsilon}\psi_e(f_e) = \sum_{e\in\varepsilon} f_e^{(w)}(d_e(s)+w_e(s))\,ds = \sum_{e\in\varepsilon}\frac{l_e c_e}{1-f_e^{(w)}/c_e} - l_e c_e + f^{(w)} f_e^* \frac{l_e c_e}{(c_e-f_e^*)^2}$$

and is equal to 72000.21. As one would expect the cost rises due to the addition of tolls. We can also observe the effect of the introduction of such tolls: they influences the behavior of users, bringing the flow from user optimum towards social optimum. This is evident if we compare $f^{(w)}$ with f^* found in point d).

(f) We now redefine our cost function as:

$$\psi_e(f_e) = f_e(d_e - l_e) \tag{3}$$

Once again we can compute the system optimum for the costs above and find f^* = [6653.26, 5774.66, 3419.75, 3419.74, 10152.74, 4642.7, 3105.85, 2662.18,

3009.06, 878.6, 0.01, 2354.9, 0.01, 3419.74, 5510.04, 3043.69, 4881.8, 2415.46, 443.68, 2008.03, 3487.37, 5495.4, 2203.78, 0.0, 6300.68, 5623.52, 4881.8, 4881.8] with System optimum cost: 15095.51.

We are then requested to construct tolls $w_e \geq 0 \forall e \in \varepsilon$ such that the new Wardrop equilibrium with the such tolls will coincide with f^*.

For this purpose we recall that given a graph G such that:

(i) Each link is equipped with a nondecreasing differentiable delay function d_e, such that every cycle in G contains a link e with $d_e(0) > 0$

(ii) The delay function $d_e(f_e)$ is convex

(iii) Link tolls are chosen as $w_e = f_e^* d_e^{'}(f_e^*)$

Then, the Wardrop equilibrium flow coincides with the system optimum flow f^*. Since points (i) and (ii) are satisfied we need only construct the tolls as in (iii) to get the desired result.

Our tailored toll vector will then be w = []1.95, 0.15, 0.06 0.12, 1.43, 0.47, 0.12, 0.06, 0.25, 0.01, 0, 0.05, 0, 0.15, 0.48, 0.1, 0.07, 0.02, 0, 0.01, 0.07, 0.24, 0.06, 0, 0.38, 0.31, 0.19, 0.53].

One can verify that the flow of the Wardrop equilibrium constructed with such costs coincides with the system optimum (see attached Python notebook).