POLITECNICO
DI TORINO

HOMEWORK I

Network Dynamics and Learning

Politecnico di Torino, DATA SCIENCE AND ENGINEERING

Pierluigi Compagnone s288301
S288301@studenti.polito.it

May 14, 2022

Collaboration with Armando La Rocca and Giuseppe Desiderio

1 Exercise 1

Considering the graph $G = (V, E)$ in Figure 1, and assuming that each link l has integer capacity C_l

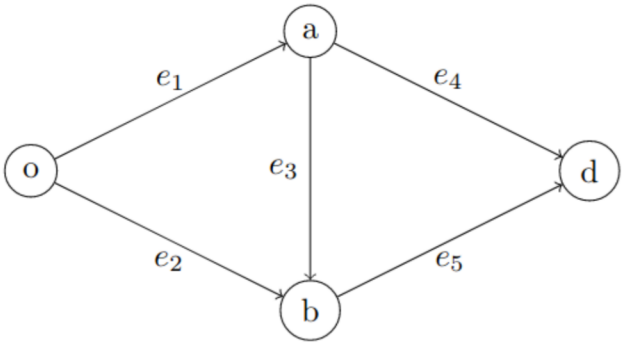

Figure 1

To find the minimum total capacity that needs to be removed for no feasible unitary $o - d$ flows to exist it is possible to exploit the *Max-flow min-cut theorem.* Indeed a consequence of this theorem is that the infimum capacity to be removed to make d not reachable from o corresponds to the minimum cut capacity $C^*_{o,d}$ among all possible *o-d cuts.*
An *o-d* cut is a partition of the node set V in two subsets U, $U^c = V \backslash U$, such that $o \in U$ and $d \in U^c$, and its capacity is given by the sum of the capacities of the edges crossed by the cut. So the first step is to find all the possible *o-d* cuts (Figure 2) and calculate their capacities.

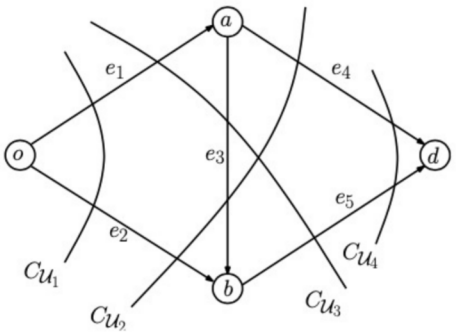

Figure 2

- $U_1 = \{o\}, U_1^c = \{a, b, d\} \longrightarrow C_{U_1} = C_1 + C_2$
- $U_2 = \{o, a\}, U_2^c = \{b, d\} \longrightarrow C_{U_2} = C_2 + C_3 + C_4$
- $U_3 = \{o, b\}, U_3^c = \{a, d\} \longrightarrow C_{U_3} = C_1 + C_5$
- $U_4 = \{o, a, b\}, U_4^c = \{d\} \longrightarrow C_{U_4} = C_4 + C_5$

Then the *min cut capacity* is

$$C^*_{o,d} = \min_{U_i} C_{U_i}, \quad 0 \leq i \leq 4$$

To disconnect o and d this capacity have to be removed on the edges crossed by the cuts of minimum capacity.

The throughput $\tau_{o,d} \geq 0$ is the total flow through the network from o to d, associated with a flow vector.
As stated in the *Max flow min cut theorem*, the maximum throughput $\tau^*_{o,d}$ coincides with the minimum cut capacity $C^*_{o,d}$ among all o-d cuts.

$$\tau^*_{o,d} = C^*_{o,d}$$

Assuming that the capacities on the links are:

$$C_1 = C_4 = 3, \quad C_2 = C_3 = C_5 = 2$$

The cut capacities calculated above become:

- $C_{U_1} = C_1 + C_2 = 5$
- $C_{U_2} = C_2 + C_3 + C_4 = 7$
- $C_{U_3} = C_1 + C_5 = 5$
- $C_{U_4} = C_4 + C_5 = 5$

In this case $\tau^*_{o,d} = C^*_{o,d} = 5$ and there are three cuts of minimum capacity C_1, C_3, C_4.

To increase the maximal throughput adding 1 unit of additional capacity, it should be possible to raise the capacities of all the minimum cuts, allocating the additional capacity on a single edge. However in this case it is not possible because there is no edge crossed by the all the three cuts, as shown in Figure 2, so with only 1 unit of capacity the maximal throughput would not grow.

Considering 2 units of additional capacity to be allocated it is possible to increase the minimum cut capacity and so the maximum throughput, however the additional capacity has to be distributed on two different edges, always because there is no link shared among the three minimum capacity cuts. Thus the maximum reachable throughput in this case is $\tau^*_{o,d} = 6$ and the combinations of capacity allocation that allow to increase the capacity of all the three minimum cuts are:

- $C_1, C_5 + 1 \longrightarrow C_{U_1} = 6,\ C_{U_2} = 7,\ C_{U_3} = 7,\ C_{U_4} = 6 \longrightarrow \tau^*_{o,d} = C^*_{o,d} = 6$
- $C_1, C_4 + 1 \longrightarrow C_{U_1} = 6,\ C_{U_2} = 8,\ C_{U_3} = 6,\ C_{U_4} = 6 \longrightarrow \tau^*_{o,d} = C^*_{o,d} = 6$
- $C_2, C_5 + 1 \longrightarrow C_{U_1} = 6,\ C_{U_2} = 8,\ C_{U_3} = 6,\ C_{U_4} = 6 \longrightarrow \tau^*_{o,d} = C^*_{o,d} = 6$

The same reasoning can also be repeated in case of 4 additional capacity units. Therefore, allocating all the units on a single edge can not increase the minimum cut capacity because there would still be a cut of capacity 5. Instead, allocating 3 capacity units on one edge and 1 on another edge is possible to increase the capacity of all the cuts, however the minimum cut capacity reachable would be 6. But in this case there are better capacity allocations that enable to reach a greater maximum throughput because they would raise the capacity of all cuts to a greater value.

- $C_1, C_5 + 2 \longrightarrow C_{U_1} = 7,\ C_{U_2} = 7,\ C_{U_3} = 9,\ C_{U_4} = 7 \longrightarrow \tau^*_{o,d} = C^*_{o,d} = 7$
- $C_1, C_4 + 2 \longrightarrow C_{U_1} = 7,\ C_{U_2} = 9,\ C_{U_3} = 7,\ C_{U_4} = 7 \longrightarrow \tau^*_{o,d} = C^*_{o,d} = 7$
- $C_2, C_5 + 2 \longrightarrow C_{U_1} = 7,\ C_{U_2} = 9,\ C_{U_3} = 7,\ C_{U_4} = 7 \longrightarrow \tau^*_{o,d} = C^*_{o,d} = 7$
- $C_1{+}2, C_4, C_5{+}1 \longrightarrow C_{U_1} = 7, C_{U_2} = 8, C_{U_3} = 8, C_{U_4} = 7 \longrightarrow \tau^*_{o,d} = C^*_{o,d} = 7$
- $C_5{+}2, C_1, C_2{+}1 \longrightarrow C_{U_1} = 7, C_{U_2} = 8, C_{U_3} = 8, C_{U_4} = 7 \longrightarrow \tau^*_{o,d} = C^*_{o,d} = 7$
- $C_1, C_2, C_4, C_5{+}1 \longrightarrow C_{U_1} = 7, C_{U_2} = 9, C_{U_3} = 7, C_{U_4} = 7 \longrightarrow \tau^*_{o,d} = C^*_{o,d} = 7$

So, adding 4 units of capacity the maximum feasible throughput is $\tau^*_{o,d} = 7$, and this result is obtained choosing each of the allocations listed above.
Moreover these allocations are equivalent also in terms of the sum of cut capacities, indeed in all the cases the cut capacities sum to 30.

2 Exercise 2

Given a set of people $P = \{p_1, p_2, p_3, p_4\}$ and a set of books $B = \{b_1, b_2, b_3, b_4\}$. Each person is interested in a subset of books, in particular:

$$p_1 \longrightarrow (b_1, b_2), \quad p_1 \longrightarrow (b_2, b_3), \quad p_3 \longrightarrow (b_1, b_4), \quad p_4 \longrightarrow (b_1, b_2, b_4)$$

.

This interest pattern is represented with the simple bipartite graph $G = (V, E)$ in Figure 3.

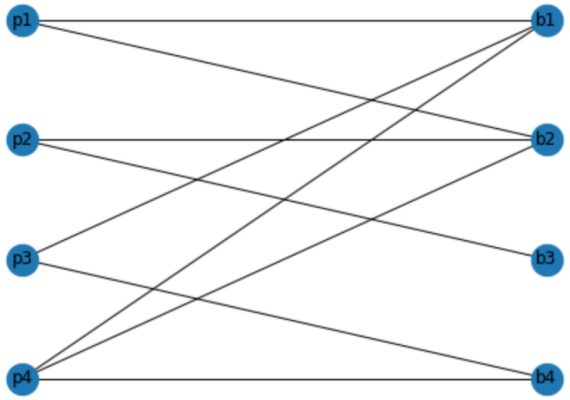

Figure 3

A matching in a simple graph $G'(V', E')$ is a subset of links $M \subseteq E'$ that have no node in common, and it is called perfect, or complete, if all nodes are matched. In the special case of a simple bipartite graph, like $G(V, E)$, with respect to the partition $V = P \cup B$, a matching M is called *P-perfect* if every node $p \in P$ is matched in M.
To establish whether there exists a perfect matching that assigns to every person a book of interest, means to find a *P-perfect* matching. To reach this goal is

possible to build a directed capacitated auxiliary network $G1$ starting from the original graph $G = (V, E)$ and then calculate a maximal flow on $G1$.
The node set of $G1$ is $V \cup o \cup d$, while its edge set is built as follow:

- For every node $p \in P$, an edge (o, p), with capacity 1 is added;
- For every node $b \in B$, an edge (b, d), with capacity 1 is added;
- For every undirected edge (p, b) in G, a directed edge (p, b) in $G1$ with capacity 1 is added;

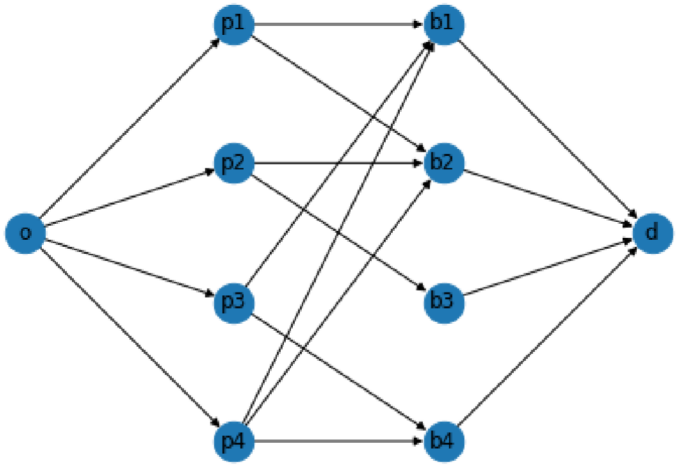

Figure 4

Solving the max flow problem on the graph in Figure 4 comes out that the maximum throughput is $\tau^*_{o,d} = 4$, thus a *P-perfect* matching exists because $\tau^*_{o,d} = |P|$. This is true because the maximum flow has integer components, being the capacities all integers, and because of the structure of $G1$. Indeed if there is a maximum flow with $\tau^*_{o,d} = |P|$, then every node in P has an input and an output flow 1, since all the edges have capacity 1, so from every node in P a unit of flow goes to a node in B. Moreover no node in B can receive flow from more than one node in P because it can have an outflow not greater than 1 toward d.
Furthermore in this case a *P-perfect* matching is a general perfect matching because $|P| = |B|$. In Figure 5 is showed a possible perfect matching found solving the max flow problem on $G1$.

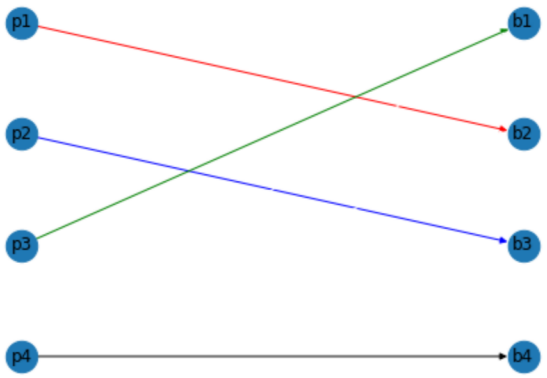

Figure 5

Assume now that there are multiple copies of the book, specifically the books' distribution is $(2, 3, 2, 2)$, and a person can take more than a book of interest, but not more copies of the same book. To solve the book assignment problem in order to maximize the number of books of interest assigned in total, the same process of the previous step is exploited.
However some adjustments are required, in particular the number of copies of a book b is reflected on the capacity of the edge (b, d) in the auxiliary network, whose structure is the same represented in Figure 4. Moreover also the capacities on the edges between o and the nodes of the set P have to be increased, otherwise it would be a bottleneck for the maximum flow through the network. In this case this capacities are increased to 4 that is the number of different books, considering that the people are not interested in taking more than one copy of the same book, but it could also be raised further without affecting the result. Then the capacity of edges among the nodes of P and those of B remain the same, because as said before people do not want multiple copies of the same book.
Modifying the network capacities of the graph in Figure 4 as just described, and solving the max flow problem on the resulting graph, the maximum throughput becomes $\tau^*_{o,d} = 8$ and stands for the total number of books assigned. The relative book assignment is represented in Figure 6.

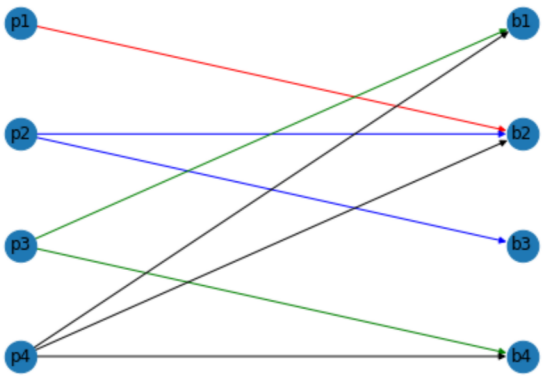

Figure 6: assigned book considering books' distribution (2,3,2,2)

Finally, supposing that there is the possibility to sell a copy of a book and to buy a copy of another book, it is intuitive to understand which book to sell and which to buy. It is enough to notice that there are 2 copies of b_3 but only one person is interested in taking it, while there are are 3 people interested in the book b_1, but only two copies are available. Thus, it is convenient to sell a copy of b_3 and buy a copy of b_1 to maximize the assigned book.
Indeed, changing the distribution of the books to $(3, 3, 1, 2)$, and consequently the capacity in the auxiliary graph on the edges from the nodes of set B to d, the maximum throughput becomes $\tau^*_{o,d} = 9$, meaning that all the available copies are assigned to the interested people.
The new assignment pattern is represented in Figure 7.

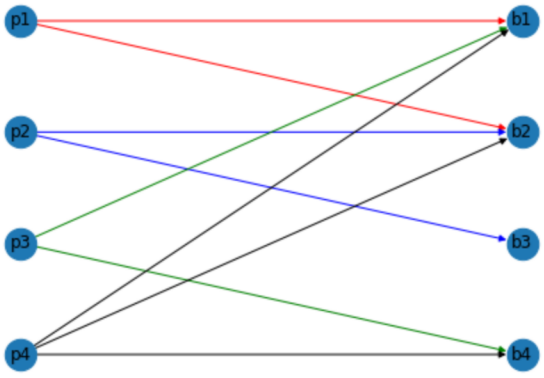

Figure 7: assigned book considering books' distribution (3,3,1,2)

3 Exercise 3

In this exercise some analysis are performed on a simplified version of the real highway network of Los Angeles $G = (V, E)$, represented in Figure 8.

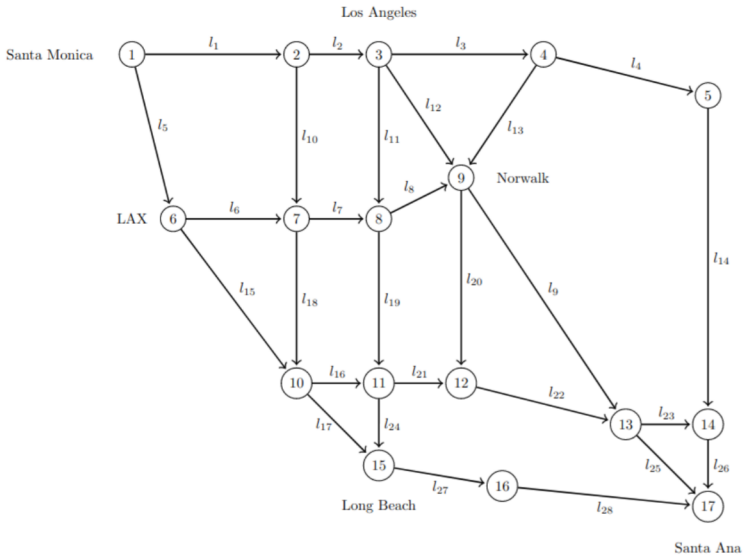

Figure 8

The data available about this network are the following:

- The node-link incidence matrix B;
- The vector of capacities C, containing for each edge $e \in E$ its maximum flow capacity C_e;
- The vector of the minimum travelling times l, composed by the travelling time l_e for each edge $e \in E$, when the corresponding road is empty.

The most of the problem faced in this section are network flow optimization problems of the type described below.

$$\min_f \sum_{e \in E} \psi_e(f_e)$$

$$f \geq 0$$
$$Bf = v$$

Where B is the node-link incidence matrix, $v \in \mathbb{R}^{|V|}$ is a zero sum vector of exogenous net flows and ψ_e is a convex non decreasing link cost function, that depends on the flow $f_e \geq 0$ through the link e. The goal of these optimization problems is to find a flow vector f that is feasible and minimizes the cost function. These tasks have been solved exploiting CVXPY, a python library for convex optimization problems. The proposal of the report for this exercise is mainly showing the process and briefly describing the theory behind it. The results that are not explicitly written here are all available in the notebook *hw-1-2021-es3.ipynb*.

The first task is to find the shortest path between node 1 and 17, this is equivalent to find the path with shortest travel time in an empty network.
To shape this situation as a network flow problem the exogenous flow vector becomes $v = (\delta^{(1)} - \delta^{(17)})$, so there is a unit of flow entering the source node and a unitary outflow from the destination node, in order to impose a unitary flow from 1 to 17. Then the link cost function is $\psi_e(f_e) = l_e f_e$, which is linear because congestion effects are not considered in this problem, indeed the cost for a unit of flow on link e is constantly l_e.
Thus the shortest path problem is formulated as follows:

$$\min_f \sum_{e \in E} l_e f_e$$

$$f \geq 0$$
$$Bf = v, \quad v = \delta^{(1)} - \delta^{(17)}$$

and the resulting shortest path is **s.p.**$= (e_1, e_2, e_9, e_{12}, e_{25})$

To find the maximum flow from node 1 to node 17 the optimization problem is slightly different, however also this is convex and so CVXPY can be used in this case too.
A new variable τ is defined in this case, which is the throughput to maximize, and the optimization problem is:

$$\max \tau$$
$$\tau \geq 0$$
$$0 \leq f \leq C$$
$$Bf = \tau(\delta^{(1)} - \delta^{(17)})$$

The maximum throughput obtained is $\tau^*_{1,17} = 22448$, and the correspondent flow vector is reported in the notebook.

Given the flow vector f in the file *flow.mat*, the exogenous net flow vector v is calculated as:

$$Bf = v$$

This vector defines the flow exiting or entering the network nodes, and it complies with the mass conservation principle.

$$\mathbf{1}'v = 0 \quad \textit{mass conservation}$$

A positive component v_i of v means that there is an inflow in the node i of quantity $|v_i|$, while a negative component stands for an outflow from the correspondent node.

From now on, the exogenous flow vector is assumed to be 0 for all the nodes, except for the node 1 where there is an inflow v_1 as computed before, and node 17 where there is a negative inflow equal to $-v_1$. It means that there is a single origin in node 1 and a single destination in node 17.

Given the following delay function:

$$d_e(f_e) = \frac{l_e}{1 - f_e/C_e} \tag{1}$$

that is convex and non decreasing, moreover it considers the congestion effects because the capacities are finite. In fact as f_e increases, the denominator decreases and so the delay is greater.
The system optimum traffic assignment problem is a network flow optimization problem whose cost function is defined as follows:

$$\min \sum_{e \in E} f_e d_e(f_e)$$

and replacing the delay defined in (1) it becomes:

$$\min \sum_{e \in E} \frac{f_e l_e}{1 - f_e/C_e} = \min \sum_{e \in E} \left(\frac{l_e C_e}{1 - f_e/C_e} - l_e C_e\right)$$

The constraints are the standard defined for network flow optimization. This optimization can be interpreted as solving the traffic assignment problem from the point of view of a central system, indeed the objective is to minimize the total delay, defined as the sum of the delays on all the links.
The social optimum flow vector, obtained solving this problem, and the correspondent cost are reported in the notebook.

Then the traffic assignment problem is shaped in a different way, no longer as it was managed by a central system, but rather as the outcome of a selfish behavior of users. Such behaviour is modeled by assuming that each user chooses its path with the will to minimize its own delay. This situation is formalized by the notion of *Wardrop equilibrium*, which is the vector obtained by optimizing:

$$\min \sum_{e \in E} \int_0^{f_e} d_e(s) ds$$

$$f \geq 0$$
$$Bf = v$$

Solving the integral with the delay function defined in (1) the cost function becomes:

$$\min \sum_{e \in E} l_e C_e [\log(C_e) - log(C_e - f_e)]$$

Also in this case the *Wardrop equilibrium* flow vector $f^{(0)}$ and its cost are calculated and printed in the code.

To assess how much worse the Wardrop equilibrium is than the social optimum, the *price of anarchy* is introduced. It is simply defined as the ratio between the total delay at the Wardrop equilibrium and the minimum possible total delay *(total delay at system optimum)*.

Now consider a delay function $d_e(f_e) + \omega_e$ for the Wardrop equilibrium, where ω_e is a toll on link e calculated with the formula:

$$\omega_e = f_e^* d_e'(f_e^*) \tag{2}$$

where f_e^* is the flow at the system optimum.
Computing the derivative the tolls can be calculated as:

$$\omega_e = f_e^* \frac{l_e C_e}{(C_e - f_e^*)^2}$$

Now the new objective function for the Wardrop equilibrium becomes:

$$\min \sum_{e \in E} \int_0^{f_e} (d_e(s) + \omega_e) ds = \min \sum_{e \in E} l_e C_e [\log(C_e) - log(C_e - f_e)] + \omega_e f_e$$

The obtained Wardrop flow vector is equivalent to the system optimum vector, indeed the correspondent *price of anarchy* becomes **PoA** = 1. The reason is that adding tolls as defined in (2) to the delay, is like to make selfish users pay for the cost that the other users pay because of their choice.

From now on, instead of the total delay, it is considered the total additional delay compared to the total delay in free flow. The cost function of the system optimum traffic assignment problem becomes:

$$\min \sum_{e \in E} f_e (d_e(f_e) - l_e) \tag{3}$$

while the constraints are the same. Solving this problem as usual with CVXPY the system optimum flow f'^* respect to the new cost function is calculated and it is reported in the notebook together with the respective cost.
In order to construct the tolls ω_e^* such that the new Wardrop equilibrium have a *price of anarchy* equal to 1, they are calculated as in (2), but of course this time considering the new system optimum f'^* and the new delay function. Although the delay function is different from the previous step, the only change is given by the subtraction of a constant l_e, so the derivative is the same. Thus the resulting tolls are:

$$\omega_e^* = f_e'^* \frac{l_e C_e}{(C_e - f_e'^*)^2}$$

and so the resulting objective function for the Wardrop equilibrium is:

$$\min \sum_{e \in E} \int_0^{f_e} (d_e(s) - l_e + \omega_e^*) ds = \min \sum_{e \in E} l_e C_e [\log(C_e) - log(C_e - f_e)] - l_e f_e + \omega_e^* f_e$$

The results are shown in the notebook as always and to check that the tolls are well designed the *price of anarchy* is calculated, obtaining $\mathbf{PoA} = 1$. Therefore the Wardrop equilibrium with tolls ω_e^* is equivalent to the system optimum, so the tolls are correct.