Network Dynamics Homework I

Claudio Fantasia(s319911), Minh Triet Ngo(s309062)

19 November, 2023

1 Exercise 1

Consider the network in Figure 1 with link capacities $c_1 = c_3 = c_5 = 3,\ c_6 = 1,\ c_2 = c_4 = 2$

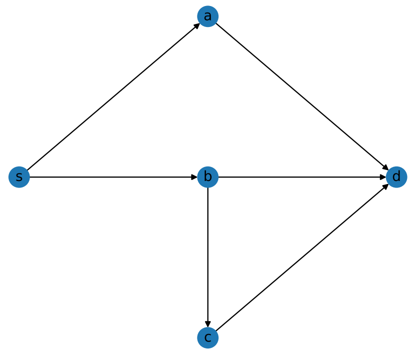

Figure 1: Graph Topology of Exercise 1

1.1 a)

What is the minimum aggregate capacity that needs to be removed for no feasible flow from o to d to exist? First of all it is important to notice that there exist three paths from o to d:
$\gamma_1 = (e_1, e_2)$
$\gamma_2 = (e_3, e_4)$
$\gamma_3 = (e_3, e_5, e_6)$

It's honorable to cite the **Menger's Theorem** which tell us how many nodes or links must be removed from a graph to disconnect two nodes in the case where we do not take into account the capacity of links. This theorem gives us some insights about the problem.
In our case we have two *link-independent path o-d*, because γ_2 and γ_3 have e_3 in common.
We have chosen to remove e_2 because it's the link with less capacity capable to disconnect γ_1. And then we could decide to disconnect e_3 in order to disconnect both γ_2 and γ_3 or to disconnect first e_4 for γ_2 and e_5 for γ_3. Since $c_3 = c_4 + c_5$, we decided just to remove e_3 and e_2 from the graph.
The minimum aggegate capacity that needs to be removed is $c_2 + c_3 = 5$

1.2 b)

What is the maximum aggregate capacity that can be removed from the links without affecting the maximum throughput from o to d?
In order to solve this problem, we have to analyze the several cuts present in the graph and then we

can remove capacity from the cuts that have higher capacity than the minimum cuts.
The cuts are:
$\{o\} \to \{a,b,c,d\} = c_1 + c_3 = 6$
$\{o,a\} \to \{b,c,d\} = c_2 + c_3 = 5$
$\{o,b\} \to \{a,c,d\} = c_1 + c_4 + c_5 = 8$
$\{o,c\} \to \{a,b,d\} = c_1 + c_3 + c_6 = 7$
$\{o,a,b\} \to \{c,d\} = c_2 + c_4 + c_5 = 7$
$\{o,b,c\} \to \{a,d\} = c_1 + c_4 + c_6 = 6$
$\{o,a,c\} \to \{b,d\} = c_2 + c_3 + c_6 = 6$
$\{o,a,b,c\} \to \{d\} = c_2 + c_4 + c_6 = 5$
Notice that we can't remove capacity from e_2, e_3, e_4, e_6 because these edges are present in the minimum cuts, so we can only remove capacity from e_1, e_5. So focusing on the minimum cut containing e_1: $\{o\} \to \{a,b,c,d\} = e_1 + e_3 = 6$ we removed 1 capacity from e_1. Then focusing on the minimum cut containing e_5 : $\{o,a,b\} \to \{c,d\} = e_2 + e_4 + e_5 = 7$ we removed 2 capacity from e_5.
Overall we can remove at most 3 capacities.

1.3 c)

You are given $x > 0$ extra units of capacity ($x \in Z$). How should you distribute the in order to maximize the throughput that can be sent from o to d? Plot the maximum throughput from o to d as a function of $x \geq 0$.
The first mandatory move is to increase the minimal cuts : $\{o,a\} \to \{b,c,d\} = e_2 + e_3 = 5$ and $\{o,a,b,c\} \to \{d\} = e_2 + e_4 + e_6 = 5$ Notice that they have in common the edge e_2 so we increase it by one, increasing the *throughput* τ by 1.
We are in this new situation:
$\{o\} \to \{a,b,c,d\} = c_1 + c_3 = 6$
$\{o,a\} \to \{b,c,d\} = c_2 + c_3 = 6$
$\{o,b\} \to \{a,c,d\} = c_1 + c_4 + c_5 = 8$
$\{o,c\} \to \{a,b,d\} = c_1 + c_3 + c_6 = 7$
$\{o,a,b\} \to \{c,d\} = c_2 + c_4 + c_5 = 8$
$\{o,b,c\} \to \{a,d\} = c_1 + c_4 + c_6 = 6$
$\{o,a,c\} \to \{b,d\} = c_2 + c_3 + c_6 = 7$
$\{o,a,b,c\} \to \{d\} = c_2 + c_4 + c_6 = 6$

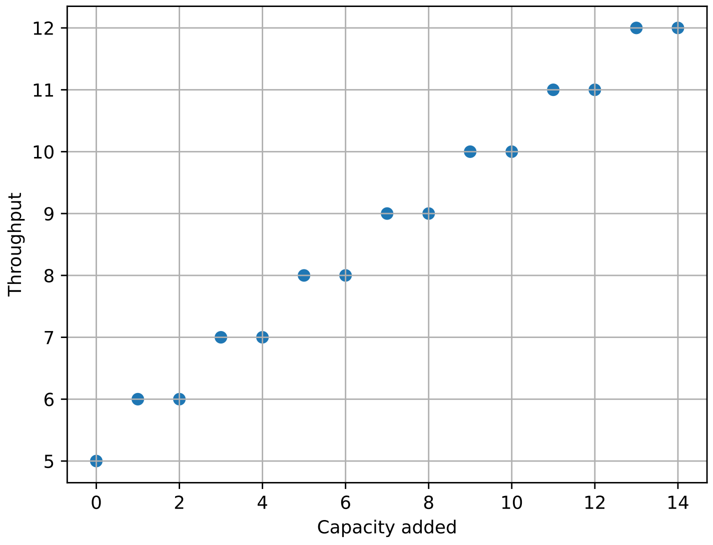

Figure 2: Throughput's graphic

We could implement a software solution that adds one capacity to one edge at time and see the combination of capacity's increments that maximize the throughput, but it would not be an elegant solution. Instead we notice that the edge e_1(or e_3) is present in a major part of the cuts, while the edge e_2(or e_4) is present in the remaining part of the cuts. Given that it is possible to infer after adding capacity to the couple (e_1, e_2) or (e_3, e_4) we are gonna increase the *throughput* by 1.
Notice that this solution could be found just looking the graph in Figure 1 and adding capacity to the edges in the shortest path. This is true after the first mandatory move that we made.
The resulting graph can be seen in Figure 2

2 Excercise 2

We formulate the problem into a bipartite graph $G = (P \cup B, \mathcal{E})$ *in which* $P = \{p1,p2,p3,p4\}$ *and* $B = \{b1,\ b2,\ b3,\ b4\}$. $\overline{G} = (P \cup B \cup \{s,d\}, \mathcal{E}^{'})$ *is the graph topology (Figure3). with* $\mathcal{E}^{'} = \mathcal{E} \cup \{(s,p) : p \in P\} \cup \{(b,d) : b \in B\}$.
G is a bipartite graph, so it means that links between nodes belonging to the same set (e.g. P) are not admissible.

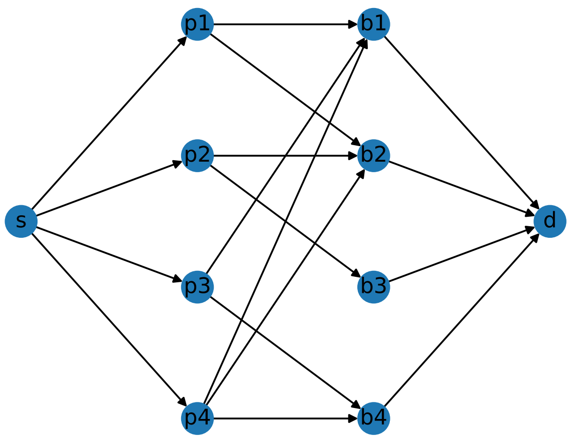

Figure 3: Extended Graph $\overline{G}$

2.1 a)

Find perfect matching
To understand the connection between max-flow problem and perfect matching problem, it is important to recall a few results.
Theorem 1 (Max-flow Min-Cut): In any flow network, the maximum value of an admissible flow equals the minimum cost of a cut.
Theorem 2 (Integral flow): If each edge of a flow network has integral capacity, then there exists an integral admissible flow of maximum value.
Corollary: Let G be the bipartite graph and G' be the flow network derived by G such that $c_e = 1$ for all $e \in \mathcal{E}^{'}$. Then the value of the maximum flow equals the size of a maximum matching in $\mathcal{E}$.
Proof
Let f^* be the maximum flow within the flow network $G^{'}$. $f^*(e) = \{0, 1\}$ $\forall e \in \mathcal{E}'$. If $f^*(b,p) = 1$ for a certain b, p then due to the conservation of mass and integral flow theorem, we have $f(e) = 0$ $\forall e \in \mathcal{N}_b \cup \mathcal{N}_p^- \setminus (b,p)$. For this reason, the number of match $|M| = \nu^*$ and $|M^*| \geq \nu^*$.

On the other hand, let f is a flow associated with the perfect matching. $f(e) = 1$ if e is a match otherwise $f(e) = 0$. Therefore, $\nu = |M^*|$ and $\nu^* \geq |M^*|$.
Hence, the corollary is true.

As a consequence of the corollary, if there exists a perfect matching within a bipartite graph then a maximum flow calculated by Ford-Fulkerson algorithm on the derived network G' can indicate such matching.

In our particular problem, the maximum flow founded is equal to the cardinality of B and P. Therefore, there exists a perfect matching as indicated by Figure 4

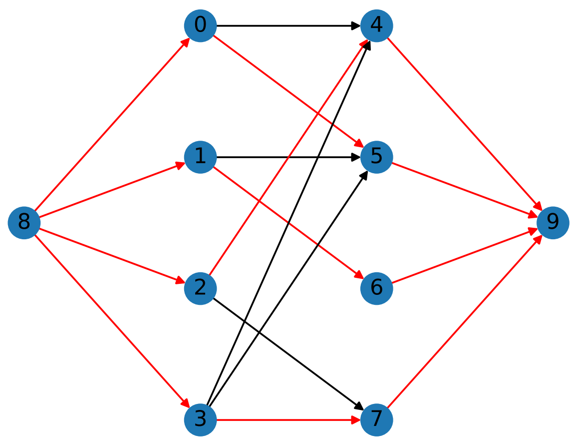

Figure 4: Perfect matching $\overline{G}$

2.2 b)

Assume now that there are multiple copies books, and the distribution of the number of copies is (2, 3, 2, 2). Each person can take an arbitrary number of different books. Exploit the analogy with max-flow problems to establish how many books of interest can be assigned in total.
Note that one person can take at most one book of the same type, that means that the capacity of links between people and books has been set up to 1. While the capacity of links between the books and the destination has been set up to the number of available copies.
This problem is similar to the previous one in the sense that we want to maximize the number of pair (p,b) the flow through which is different from zero. To do that, we set $c(s, p)$ = number of books of interest correspondent to that person, $c(s, b) = 1$ and $c(b, d)$ = the number of copies $\forall p \in P, b \in B$. Although it is safe to assume that $c(s, p) = |B|$. We can see the flow connected to the maximum flow in Figure 5

2.3 c)

Suppose that the library can sell a copy of a book and buy a copy of another book. Which books should be sold and bought to maximize the number of assigned books?

We can see directly from the topology of the graph 3 that the in-links of the node b_1 are 3 and the copies of b_1 are 2 (so the capacity of link between node b_1 and d). While the in-links of the node b_3 are equal to 1 and the copies are 2.
Overall we have to sell a copy of b_3 and buy a copy of b_1 since there is a requirement for $b1$ that is not

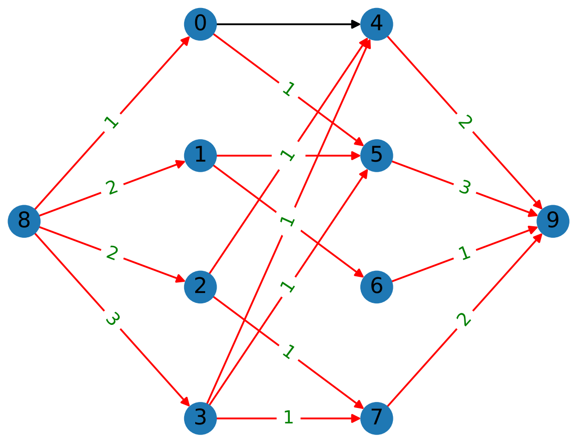

Figure 5: Books assignment

satisfied and we have an extra copy of $b3$ that is not given to any person.
Buying and selling books, so changing capacities of links (b_1,d) and (b_3,d), we obtain the flow in Figure 6

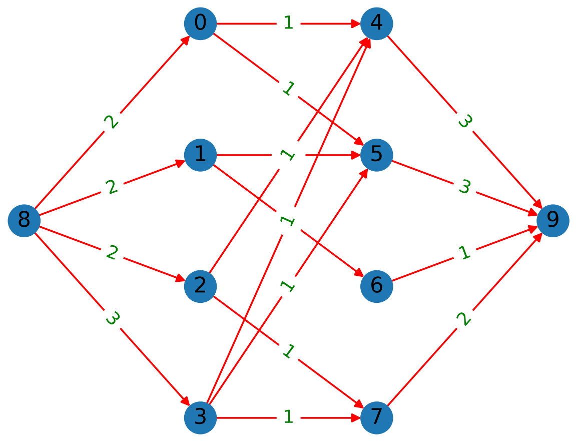

Figure 6: Books assignment with modification in capacity

3 Exercise 3

We are given the highway network in Los Angeles. To simplify the problem, an approximate highway map is given in Figure 7, covering part of the real highway network. We are given the node-link incidence matrix $\boldsymbol{B}$*. Each node represents an intersection between highways (and some of the area around).*

Each link $e_i \in (e_1, \ldots, e_{28})$, has a maximum flow capacity c_{ei}. Furthermore, each link has a minimum travelling time l_{ei}, which the drivers experience when the road is empty. In the same manner as for the capacities, the minimum travelling times are given as a vector l_e. These values are simply retrieved by dividing the length of the highway segment with the assumed speed limit 60 miles/hour.
For each link, we introduce the delay function:

$$\tau_e(f_e) = \frac{l_e}{1 - f_e/c_e}, \; 0 \leq f_e \leq c_e \tag{1}$$

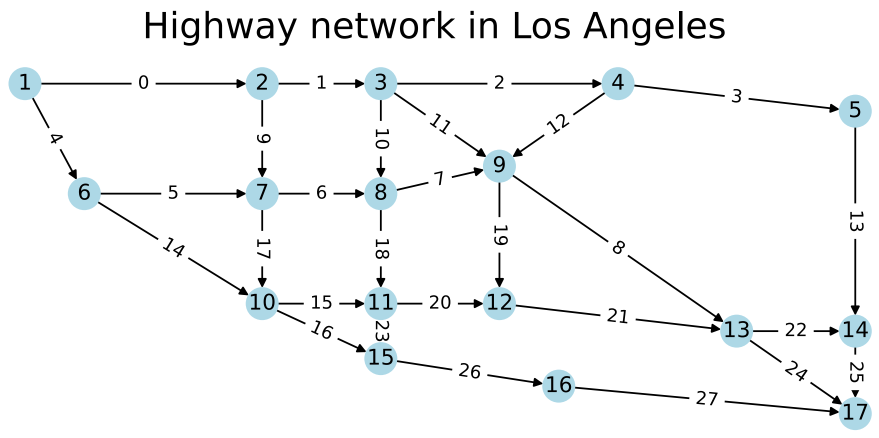

Figure 7: Highway Network in Los Angeles

3.1 a)

Find the shortest path between node 1 and 17. This is equivalent to the fastest path (path with shortest traveling time) in an empty network.

In order to find the shortest path in this network we can use the networkX function: *shortest_path* that use dijkstra algorithm to find the shortest path between two nodes.
Otherwise we can solve this optimization problem:

$$M(\nu) := \min_{Bf=\nu, 0 \leq f_e < c_e} \psi_e(f_e)$$
$$\psi_e(f_e) = l_e \cdot f_e$$
$$\nu = \delta^{(o)} - \delta^{(d)}$$

From now on, in order to solve optimization problems we implemented our solutions present in Network_HW1_Ex3.ipynb using **Gurobi**(gurobipy library)
After solving the optimization problem we compared our results with the one given by *nx.shortest_path* and they are the same. The shortest path is $(v_1, v_2, v_3, v_9, v_{13}, v_{17})$ as we can see in 8

3.2 b)

Find the maximum flow between node 1 and 17. For solving b) we used the *max-flow min-cut theorem* which states that the maximum flow through any network from a given source to a given sink is exactly equal to the value of the minimum cut.
We have used the networkX function: *nx.maximum_flow*, but another way to compute it was to implement the **Ford and Fulkerson algorithm**, which guarantee that the flow found is integer (only if all the capacities are integers) and correspond to the maximum throughput.
We can see the maximum flow found in the Figure 9

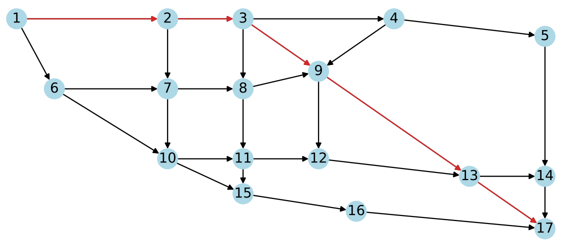

Figure 8: Flow of the shortest path

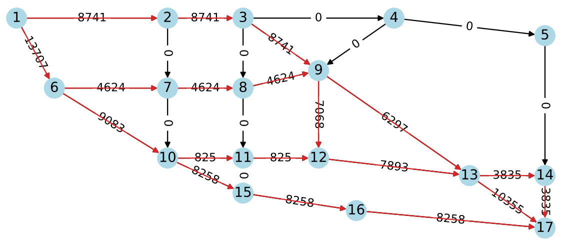

Figure 9: Maximun flow between 1 and 17

3.3 c)

Given the flow vector in flow.mat, compute the external inflow ν satisfying $Bf = \nu$
We just took the flow vector from the file *flow.mat* and computed the python command B @ $f = v$. This is easily possible because B it's not large, but knowing that $B \in Z^{|nodes|,|edges|}$ if the number of nodes and edges became too high and so B too large, we may use *GMRES algorithm* to find a faster solution.
Then we have set $v_1 = -v_{17}$ and all the other components of exogenous flow equal to 0 for the next task.

3.4 d)

Find the social optimum f^* with respect to the delays on the different links $\tau_e(f_e)$.
To find the flow that satisfy the **Social/System Optimum Traffic Assignment(SO-TAP)**, we have to change our optimization problem into:

$$M(\nu) := \min_{Bf=\nu, 0 \le f_e < c_e} \psi_e(f_e)$$

$$\psi_e(f_e) = \sum_{e \in \mathcal{E}} f_e \cdot \tau_e(f_e) = \sum_{e \in \mathcal{E}} \left(\frac{l_e c_e}{1 - f_e/c_e} - l_e c_e \right)$$

Notice that Gurobi does not support dividing by variables, but there is another way to compute it. Instead of maximizing(minimizing) $1/f$, the user can maximize(minimize) a dummy variable z and place as a constraint that $z \cdot f = 1$ in addition to the existing constraints $Bf = \nu$, $f \ge 0$ and $f_e < c_e$
The system optimal flow is:

$$f = [6454., 5919., 2995., 2995., 9828., 4497., 2897., 2436., 3047., 535., 0., 2924., 0., 2995., 5331., 2731., 4734., 2134., 461., 2313., 3192., 5505., 2311., 0., 6242., 5306., 4734., 4734.]$$

3.5 e)

Find the Wardrop equilibrium $f^{(0)}$
First of all solving the Wardrop equilibrium means finding a flow that solves the **User Optimum Traffic Assignment Problem(UO-TAP)**.
Wardrop equilibrium is the solution of UO-TAP which can be interpreted as a network flow $f^{(0)}$ that is associated with an o-d path distribution z such that $z_\gamma > 0$ for some o-d path γ

$$\min_{0 \le f_e < c_e, Bf = \nu(\delta^{(o)} - \delta^{(d)})} \sum_{e \in \mathcal{E}} \int_0^{f_e} \tau_e(s) ds$$

Instead of giving the integral as input to Gurobi, we integrated the function, obtaining:

$$\min_{0 \le f_e < c_e, Bf = \nu(\delta^{(o)} - \delta^{(d)})} \sum_{e \in \mathcal{E}} -l_e c_e \log(1 - f_e/c_e)$$

Solving this optimization problem we found that the flow is equal to

$$\begin{aligned} f = [&6542., 6542., 2231., 2231., 9740., 4505., 2705., 2213., 3337., 0., 198., 4113., 0.,\\ &2231., 5235., 2261., 4775., 1801., 691., 2988., 2951., 5939., 2486., 0., 6790., 4717.,\\ &4775., 4775.] \end{aligned}$$

With this solution we can compute the **Price of Anarchy(PoA)** that is a number always greater or equal than 1 and it can be represented as the ratio between the total delay at the Wardrop equilibrium with respect to the total delay at system optimum. It is defined as follow:

$$PoA(0) = \frac{\sum_{e \in \mathcal{E}} f_e^{(0)} \tau_e(f_e^{(0)})}{\min_{0 \le f_e < c_e, Bf = \nu(\delta^{(o)} - \delta^{(d)})} \sum_{e \in \mathcal{E}} f_e \tau_e(f_e)}$$

We found out that in our case the PoA computed is 1.0135

3.6 f)

Introduce tolls, such that toll on link e is $\omega_e = \psi'(f_e^*) - \tau_e(f_e^*)$. For the considered $\psi_e(f_e), \omega_e = f_e^* \tau'(f_e^*)$ where f_e^* is the flow at the system optimum. Now the delay on link e is given by $\tau_e(f_e) + \omega_e$. Compute the new Wardrop equilibrium $f^{(\omega)}$.
Total travel time found computed using user optimum flow is always greater or equal than the one computed using system optimum flow and this can be proven, because the **Price of Anarchy(PoA)** is a number always greater or equal than 1
For solving these disparities between the system optimal and the user optimal, we can introduce tolls in order to make an edge less desirable for a single user/agent.
In order to align System optimum and User optimum we first have to compute the tolls as $\omega = f_e^* \tau'(f_e^*)$ using the System optimal flow. Then we have to compute our new optimization problem as:

$$\min_{0 \le f_e < c_e, Bf = \nu(\delta^{(o)} - \delta^{(d)})} \sum_{e \in \mathcal{E}} \int_0^{f_e} \tau_e(s) ds + f_e \omega_e$$

After this addition of tolls we can observe the alignment of optimal flow for System Optimum and User Optimum plus tolls in the Figure 10

3.7 g)

Find Social optimal flow and Wardrop equilibrium with tolls with respect to the new cost function

$$\psi_e(f_e) = f_e(\tau_e(f_e) - l_e)$$

After that we have found the optimal flow for the minimization problem:

$$\psi_e(f_e) = \sum_{e \in \mathcal{E}} f_e \cdot \tau_e(f_e) = \sum_{e \in \mathcal{E}} \left(\frac{l_e c_e}{1 - f_e/c_e} - l_e c_e - l_e f_e \right)$$

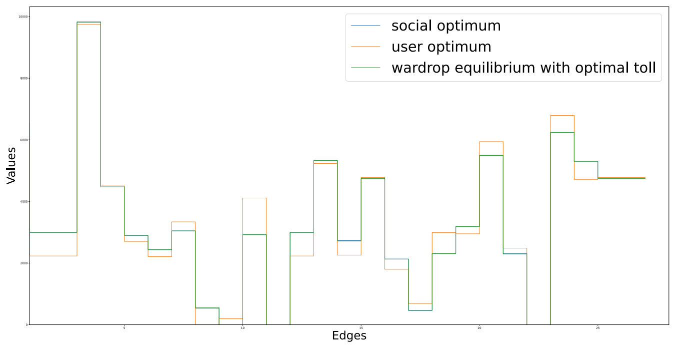

Figure 10: Comparison between SO, UO and UO with toll

We computed tolls as $\omega_e = f_e^* \tau^{'}(f_e^*)$ where f_e^* is the social optimal flow with the changed traffic function. Then as did before we just inserted the tolls into definition of Wardrop Equilibrium:

$$\min_{0 \le f_e < c_e, Bf = \nu(\delta^{(o)} - \delta^{(d)})} \sum_{e \in \mathcal{E}} \int_0^{f_e} \tau_e(s) ds + f_e \omega_e$$

The resulting optimal flow given Wardrop equilibrium with tolls is very close to the optimal social flow. Indeed the price of anarchy computed after the toll design is 1.0000225758836336.

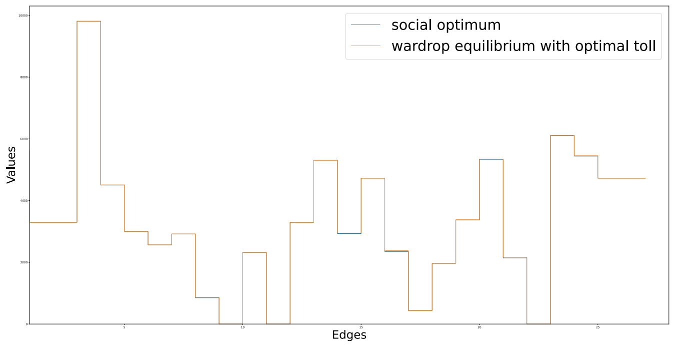

Figure 11: Comparison between SO, UO with toll