Homework 1

Vincenzo Marciano' (282004)

November 14, 2021

Abstract

In the first Homework assignment for Network Dynamics and Learning course three different exercises will be exploited, in order to evaluate the first sections of the program..

The exercises are made in collaboration with Andrea Ferretti (289677).

Exercise 1

1.a

Since the problem states we have to deal with a set of capacities in which each one is equal to C_l, we can re-write for a moment our multigraph in Figure 1 as:

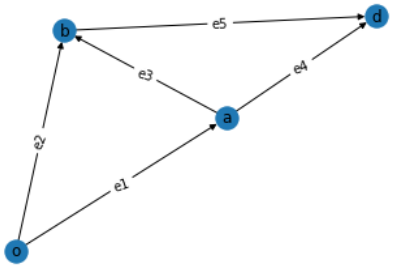

Figure 1: Initial multigraph $\mathcal{G}$

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, C_l)$$

Also, each of the four nodes represented in the figure can be intended, without any concern, different from each other. Taking in mind these premises, the answer to our question becomes a specific constraint of the *maximum flow problem*: we ensure each *o-d flow*, namely f, must be at least equal to zero and at most equal to C_l. In a more compact way we'll have:

$$0 \leq f \leq C_l$$

Now, we must rely on the concept of *o-d cut*, i.e. a partition between two subsets of nodes $\mathcal{U}_i$ and $\mathcal{U}_i^c$, in which origin node o belongs to the first set and the destination node d belongs to the second one. Therefore, our cuts with the respective capacities would be:

$$\mathcal{U}_1 = o$$

$$\mathcal{U}_2 = o, a$$

$$\mathcal{U}_3 = o, b$$

$$\mathcal{U}_4 = o, a, b$$

With:

$$c_{\mathcal{U}_1} = c_{e_1} + c_{e_2} = 2 \cdot C_l$$

$$c_{\mathcal{U}_2} = c_{e_4} + c_{e_3} + c_{e_2} = 3 \cdot C_l$$
$$c_{\mathcal{U}_3} = c_{e_1} + c_{e_5} = 2 \cdot C_l$$
$$c_{\mathcal{U}_4} = c_{e_2} + c_{e_5} = 2 \cdot C_l$$

From here we recall the *min-cut capacity*: a minimum capacity among the whole set mentioned above will be our infimum to be removed in order to not obtain any feasible unitary flows from o to d. In a more theoretical way:

$$c^*_{o,d} = \min c_{\mathcal{U}}$$
$$\mathcal{U} \subseteq \mathcal{V}$$
$$o \in \mathcal{U}, d \notin \mathcal{U}$$

It's now obvious that $c^*_{o,d} = 2 \cdot C_l$ is the best fit for our purposes

.

1.b

With the new values of capacities illustrated in Figure 2, the resulting subset capacities are:

$$c_{\mathcal{U}_1} = c_{e_1} + c_{e_2} = 3 + 2 = 5$$
$$c_{\mathcal{U}_2} = c_{e_4} + c_{e_3} + c_{e_2} = 3 + 2 + 2 = 7$$
$$c_{\mathcal{U}_3} = c_{e_1} + c_{e_5} = 3 + 2 = 5$$
$$c_{\mathcal{U}_4} = c_{e_4} + c_{e_5} = 3 + 2 = 5$$

In this way the new known *min-cut* is $c^*_{o,d} = 5$. However, as the assignment suggests, if we add a *1-unit additional capacity* to the newtwork the result **does not change**. The key idea behind this assumption is the following: working with the minimum-capacities subsets, i.e. the ones with $C_i = 5$, one can add the unitary capacity to one link among e_1, e_2, e_4 and e_5. However, there is any link in common between *all* the aforementioned subsets $\mathcal{U}_\backslash$, and for this reason the resulting minimum capacity will be always $c^*_{o,d} = 5$.

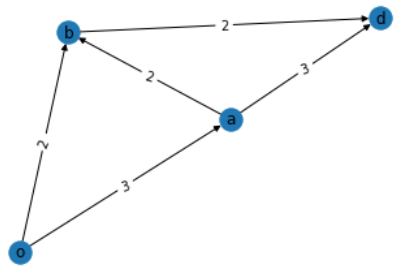

Figure 2: Multigraph with updated capacities.

1.c

Still working with the same capacities of the previous point, the new task is to add *2-unit additional capacity* along the graph: the situation here is quite different, as well as the final result.

Let's start by looking at all the links included in the three subsets of minimal capacity:

$$c_{\mathcal{U}_1} = c_{e_1} + c_{e_2}$$
$$c_{\mathcal{U}_3} = c_{e_1} + c_{e_5}$$
$$c_{\mathcal{U}_4} = c_{e_4} + c_{e_5}$$

A smart intuition is to distribute the 2 units both to e_1 and e_5 as it's shown in Figure 3, since these two links appear in at least two set of capacities. In this way the updated link capacities $c_{e_i}^{(1)}$ are:

$$c_{\mathcal{U}_1} = c_{e_1}^{(1)} + c_{e_2} = 4 + 2 = 6$$

$$c_{\mathcal{U}_3} = c_{e_1}^{(1)} + c_{e_5}^{(1)} = 4 + 3 = 7$$

$$c_{\mathcal{U}_4} = c_{e_4} + c_{e_5}^{(1)} = 3 + 3 = 6$$

Eventually, the resulting minimum capacity will be $c_{o,d}^* = 6$

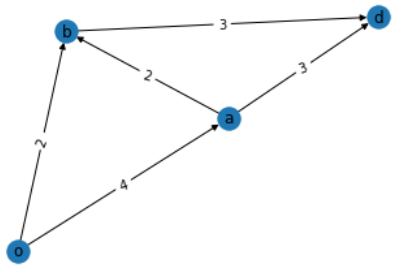

Figure 3: Multigraph with 2-units distributed capacities.

1.d

By having a total available capacity of 4 to be added, we can see, as in the previous point, how c_{e_1} and c_{e_5} are in common between all the three minimal cuts with capacity 5. The best idea so would be to add a total of 2 capacity to each one of those two edges as Figure 4 shows, therefore having as a result a new *min-cut* $c_{o,d}^* = 7$:

$$c_{\mathcal{U}_1} = c_{e_1} + c_{e_2} = 5 + 2 = 7$$

$$c_{\mathcal{U}_2} = c_{e_4} + c_{e_3} + c_{e_2} = 3 + 2 + 2 = 7$$

$$c_{\mathcal{U}_3} = c_{e_1} + c_{e_5} = 5 + 4 = 7$$

$$c_{\mathcal{U}_4} = c_{e_4} + c_{e_5} = 3 + 4 = 7$$

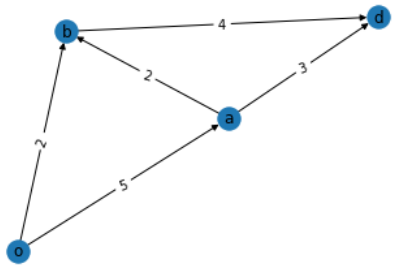

Figure 4: Multigraph with 4-units distributed capacities.

Exercise 2

2.a

In the Figure 5 the interest pattern is represented by using a bipartite graph.

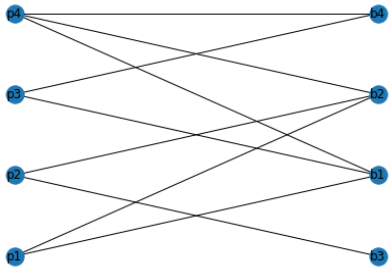

Figure 5: Resulting bipartite graph of the following exercise.

2.b

From Hall's marriage Theorem we know that for a simple bipartite graph $G = (V, E)$ and $V_0 \subseteq V$, there exists a V_0-perfect matching in G if and only if

$$|\mathcal{U}| \leq |\mathcal{N}_{\mathcal{U}}| \qquad \forall \mathcal{U} \subseteq V_0,$$

where

$$\mathcal{N}_{\mathcal{U}} = \sum_{i \in \mathcal{U}} \mathcal{N}_i$$

is the neighborhood of $\mathcal{U}$ in $\mathcal{G}$.
In other words, for every subset $\mathcal{U}$ of V_0 , we require that $\mathcal{U}$ possesses a number of neighboring nodes which is at least the number of nodes in $\mathcal{U}$.
Given a simple bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we consider the directed capacitated graph $\mathcal{G}_1$ with node set $\mathcal{V} \cup s \cup t$, and edge set constructed as follows:

- for every node $n \in V_0$, add an edge (s,n) with capacity 1;
- for every node $n \in V_1$, add an edge (n,t) with capacity 1;
- for every undirected edge (i, j) in $\mathcal{G}$, add a directed edge (i, j) in $\mathcal{G}_1$ with capacity 1.

As the left side of Figure 6 suggests, one can introduce two specific nodes (in this case 8 and 9 respectively) that can be considered the "head" and the "tail" of our graph: in other words, we want to know where the books come from and go to. There is an analogy between perfect matchings and

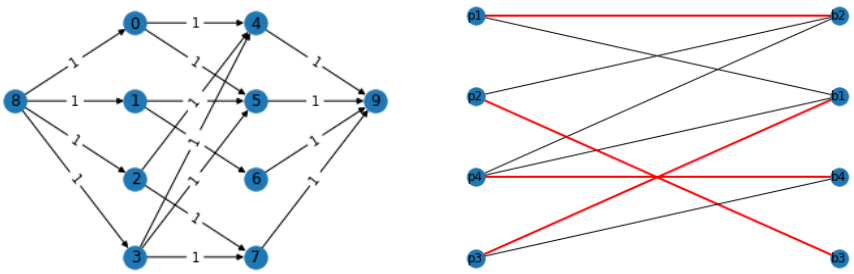

Figure 6: Left: new directed and capacitated graph. Right: resulting perfect matching graph, in which perfect matchings are displayed with red lines.

maximal flow on this auxiliary network $\mathcal{G}1$ illustrated in Figure 6 Left. In particular, a V0-perfect matching on $\mathcal{G}$ exists if and only if it there exists a flow with throughput $|V0|$ on the network $\mathcal{G}1$. We can now exploit Ford-Fulkerson algorithm to find the maximum flow that can be sent in $\mathcal{G}1$, obtaining the bipartite graph in the Right side of Figure 6.

2.c

As the exercise illustrates, the link between one person p_i and a book b_j must have a unitary capacity, since he/she can take each book once.

Furthermore, a key idea is working with *out-degree* information: each person's incoming link must have capacity equal to its out-degree in order to give to each person the maximum number of linked books.

For the books, the capacity of the edges linking their nodes to the node 9 manages the total number of books available, since each edge between people and books has a flow equal to 1 if the book is picked and the flow out going from a book node as the total number of books picked up. In this case the number of available books is given by assignment: (2,3,2,2). The max flow in the new directed graph is 8, which represents the number of books of interest that can be assigned in total.

The previously described situation is shown in the Figure 7.

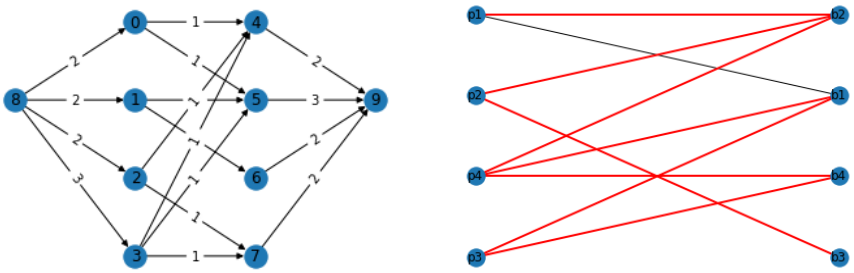

Figure 7: New resulting graph. Left: capacitated graph. Right: red lines exploiting perfect matchings.

2.d

In the last point of this exercise one key element is the *in-degree* of book nodes: taking in consideration one certain book, the optimization in this case relies on the number of copies of that book in order to match the number of links of people that could take the imputed book.

A better explanation could be retrieved by the graph displayed in Figure 8.

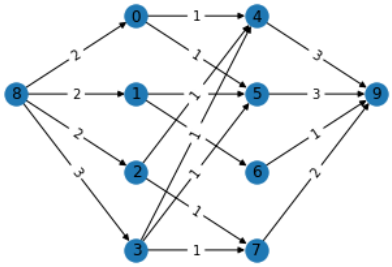

Figure 8: New resulting graph with new hypothesis.

From here, the quantitative answer that satisfies the task request is to **buy one book labeled "1"** and to **sell one book labeled "3"**.

Exercise 3

Given the node-link incidence matrix B and the details about the *capacities, minimum traveling times and flow* for every link, we can immediately construct the graph with all the required attributes:

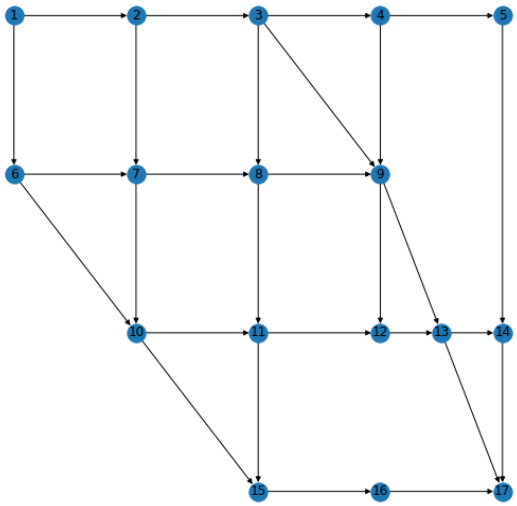

Figure 9: Graph representing the approximate highway map.

3.a

First we draw the graph with the edge labels representing the traveling times:

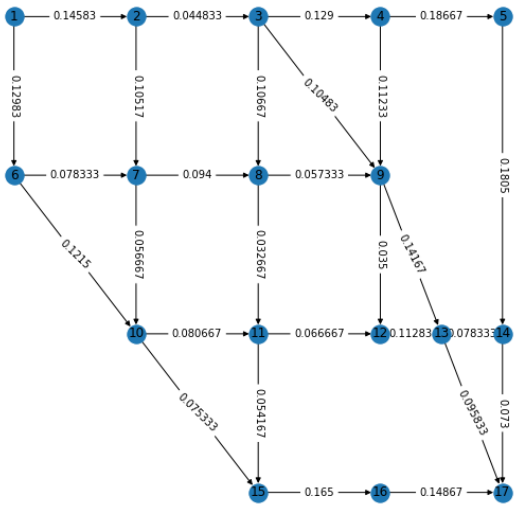

Figure 10: Graph with traveling time.

In order to find the shortest path between node 1 and 17 (or the path with shortest traveling time in an empty network) we can use a `Networkx` function called `shortest_path`, which calculates the shortest path from a node to another in the graph, considering as a weight the *travel time*. The

obtained shortest path is 0.532996, given by selecting the nodes [1, 2, 3, 9, 13, 17], as shown in Figure 11

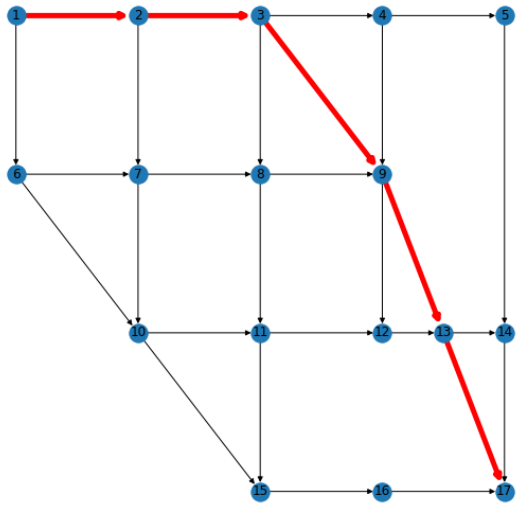

Figure 11: Shortest path on the graph.

3.b

In the following figure, the graph is shown with the edge capacities (for the max flow problem):

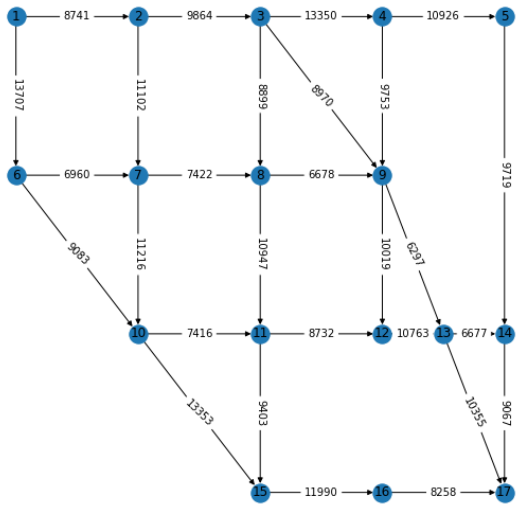

Figure 12: Graph with capacities.

The maximum flow can be easily computed using the `Networkx` method `maximum_flow`, which returns a dictionary containing the value of the maximum flow from a source node to a destination node, along with the flow passing through all the edges. The maximum flow obtained is 22448 and the value of flow for each edge is shown in Figure 13

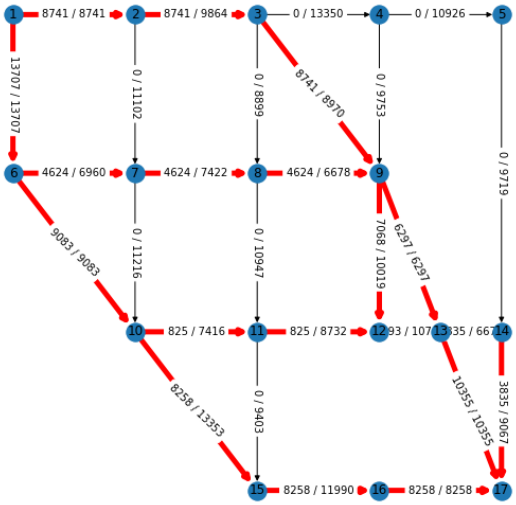

Figure 13: Maximum flow for every edge of the graph.

3.c

The graph along with the flow network in every edge is shown in the next figure:

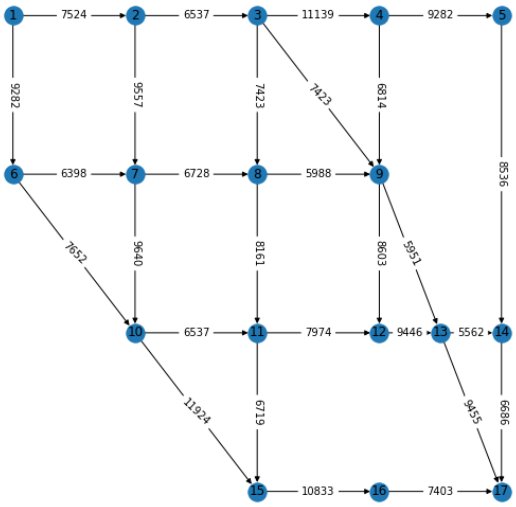

Figure 14: Graph with the flows.

In order to compute the external inflow ν satisfying $Bf = \nu$ we perform the matrix multiplication $B * f$, obtaining ν = [16806 8570 19448 4957 -746 4768 413 -2 -5671 1169 -5 -7131 -380 -7412 -7810 -3430 -23544]

3.d

For this new section of the third task, the previous ν of the first and last node are retrieved, i.e. **16806** and **-23544**.

Then, it's necessary to minimizing the total delay function given by the sum of each link $e \in \mathcal{E}$ cost function. Formally, the objective becomes:

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

Knowing that the delay function $d_e(f_e)$ is not constant, it can be ensured the non-congestion feature.

Exploiting *cvxpy* APIs, we can easily compute the social optimum flow f^* and the minimized cost needed for our task. From now on and for our convenience, the optimal flow is displayed in the submitted code, yet a screenshot will illustrate the final result, as it's shown in Figure 15.

```
Social optimal flow: [6.64219910e+03 6.05893789e+03 3.13232779e+03 3.13232589e+03
 1.01638009e+04 4.63831664e+03 3.00634073e+03 2.54263460e+03
 3.13154448e+03 5.83261212e+02 1.45164550e-02 2.92659559e+03
 1.89781986e-03 3.13232589e+03 5.52548426e+03 2.85427264e+03
 4.88644874e+03 2.21523712e+03 4.63720641e+02 2.33768761e+03
 3.31799129e+03 5.65567890e+03 2.37310712e+03 1.99567283e-03
 6.41411626e+03 5.50543301e+03 4.88645073e+03 4.88645073e+03]
```

Figure 15: Optimal flow for the proposed delay function

The **Social Optimal cost** is equal to **25943.60**.

3.e

Wardrop Equilibrium

Solving the proposed integral will result in obtaining the *Wardrop Equilibrium*. Wardrop Equilibrium is a path flow distribution such that if a path is used then the cost of the path would be miimal, obtining:

$$z_p > 0 \rightarrow c_p(z) \leq c_r(z), \forall r \in \mathcal{P}$$

Where:

- $\mathcal{P}$ is the path set;
- $c_p(z) = \sum_{e\in p} d_e(f_e)$ is the path cost is equal to the sum of the delays that compose the path.

After some algebraic calculations, the result is a new objective function to be solved as it follows:

$$\sum_{e\in\mathcal{E}} \frac{l_e C_e}{\ln(1 - f_e/C_e)}$$

From here the Wardrop Equilibrium as a solution is shown in Figure 16, still exploiting the aforementioned library. With these results, a further step is to study how *Price of Anarchy* may vary.

```
User optimal and Wadrop Equilibrium: [6.71564895e+03 6.71564803e+03 2.36740801e+03 2.36740792e+03
 1.00903510e+04 4.64539489e+03 2.80384316e+03 2.28356194e+03
 3.41848003e+03 9.22328268e-04 1.76829408e+02 4.17141061e+03
 8.92024178e-05 2.36740792e+03 5.44495611e+03 2.35317044e+03
 4.93333832e+03 1.84155266e+03 6.97110629e+02 3.03649261e+03
 3.05028094e+03 6.08677356e+03 2.58651143e+03 1.24029072e-04
 6.91874216e+03 4.95391934e+03 4.93333845e+03 4.93333845e+03]
```

Figure 16: Wardrop Equilibrium $f^{(0)}$

In doing so, the first ingredient is to evaluate the **User Minimum (Wardrop) Cost**, achieved by summing each entry of the function above: the final result will be equal to **26293**. Now, it's quite trivial observing a increment with respect to the Social Optimum discussed in the section 3.d, hence the **PoA**, given by the ratio of the two obtained costs, is greater than 1 and in particular equal to **1.013**.

Tolls

Exploiting *Tolls* is a smart way to reduce the Price of Anarchy around the value of 1, i.e. it lets the User Optimum almost equal to the Social one. In other words, we want to improve the performance of our network adjusting the "selfish" behaviour that Wardrop Equilibrium induces in our system.

To achieve the proposed result, the Tolls must be computed by the following key idea:

$$\omega_e = f_e^* d_e'(fe^*)$$

Where f_e^* is the System Optimum flow. Once Tolls are obtained, the new delay function will experience a new additive element in its equation, becoming: $d_\omega = d_e(f_e) + \omega_e$. Solving the Wardrop Equilibrium we observe a new User Optimal flow very similar to the one achieved in the Social Optimum task (Figure 17 for reference) as well as the new **User Minimum Cost**, equal to **25943.62**. As a final result, we observe an improvement in the netowrk give by the **PoA** reduced to **1**.

```
User optimal with tolls: [6.64297510e+03 6.05907793e+03 3.13247156e+03 3.13247145e+03
 1.01630248e+04 4.63825748e+03 3.00632468e+03 2.54233942e+03
 3.13149039e+03 5.83897168e+02 4.41635538e-04 2.92660593e+03
 1.13364091e-04 3.13247145e+03 5.52476733e+03 2.85422618e+03
 4.88637111e+03 2.21582997e+03 4.63985699e+02 2.33745508e+03
 3.31821174e+03 5.65566682e+03 2.37303573e+03 1.44842042e-04
 6.41412148e+03 5.50550718e+03 4.88637126e+03 4.88637126e+03]
```

Figure 17: User Optimum Flow obtained by exploiting tolls

3.f

A new cost function would be exploited in this very last task, as the exercise suggests. The function is:

$$c_e(f_e) = f_e(d_e(f_e) - l_e) = \left(\frac{l_e C_e}{1 - f_e/C_e} - l_e C_e - l_e f_e\right)$$

Social Optimum (Updated)

Obtaining the new Social Optimum is now quite straight-forward: we must update the delay function in the section 3.d with the new proposed function, obtaining a new Social Optimum flow as the one displayed in Figure 18.

```
Social optimal flow: [6.65329658e+03 5.77466230e+03 3.41971657e+03 3.41971062e+03
 1.01527034e+04 4.64278036e+03 3.10584008e+03 2.66218478e+03
 3.00907935e+03 8.78634280e+02 7.42401749e-03 2.35493830e+03
 5.94907576e-03 3.41971062e+03 5.50992306e+03 3.04369256e+03
 4.88180506e+03 2.41557456e+03 4.43662730e+02 2.00804968e+03
 3.48735309e+03 5.49540277e+03 2.20377848e+03 2.20338871e-03
 6.30070364e+03 5.62348910e+03 4.88180726e+03 4.88180726e+03]
```

Figure 18: New Social Optimal flow with the new updated delay function

Wardrop Equilibrium with Tolls (Updated)

Now, it's time to search those tolls which let us obtain an optimal flow in the Wardrop Equilibrium equal to the Social Optimal one in the Figure 18. In doing so, we must update our new tolls function, that becomes:

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

Therefore, a new integral function will be exploited to find new Wardrop Equilibrium and to guarantee the request. In particular, the new objective function to minimize is:

$$\sum_{e \in \mathcal{E}} \left(\int_0^{f_e} (d_e(s) - l_e)\, ds \right) + f_e \omega_e^* = \sum_{e \in \mathcal{E}} [l_e C_e (\ln(C_e) - \ln(C_e - f_e)) - l_e f_e + f_e \omega_e^*]$$

By solving this equation, the outcome gives a User Optimal flow **equal** to the Social Optimal one.