Network Dynamics and Learning

Homework I

Davide Gariglio

s292964@studenti.polito.it

1 Introduction

In this first *Homework* we are required to apply the theoretical knowledge acquired during the lectures and laboratory activities, such as: *Maximum flow and Minimum cut, Bipartite graphs, Matchings and Network Flow Optimization.* All the exercise have been solved in *python* language using the *NetworkX and cvxpy libraries*; the solutions of the exercises are provided with commented python notebook and explained in the following report.

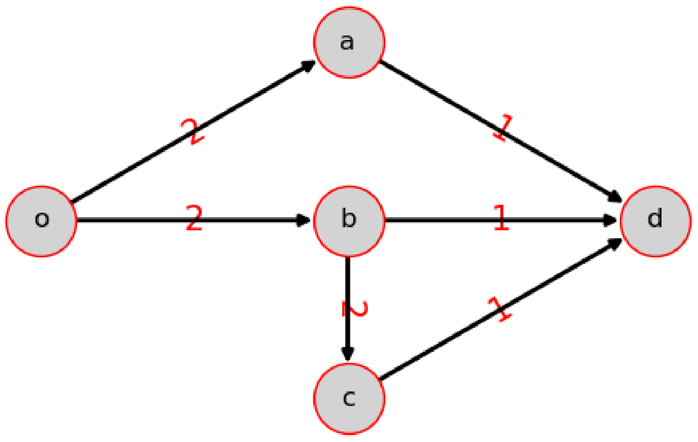

Figure 1: Network graph (with link capacities) over which the different tasks asked in Exercise 1 are performed

2 Problem 1

In this first exercise we are required to perform some *Flow analysis* on the graph reported in Fig. 1.

2.1 1.a

In this exercise we are required to calculate the *minimum aggregate capacity* that needs to be removed in order to have a no feasible flow from node o to d. Basically, as a first step, it is necessary to identify the *minimum capacity cut*.

In this way, by means of the *Max-flow Min-cut Theorem*, we can assert that the amount of flow flowing from o to d is the sum of the capacities of the arcs that joins the node partitions created by the *min-cut*.

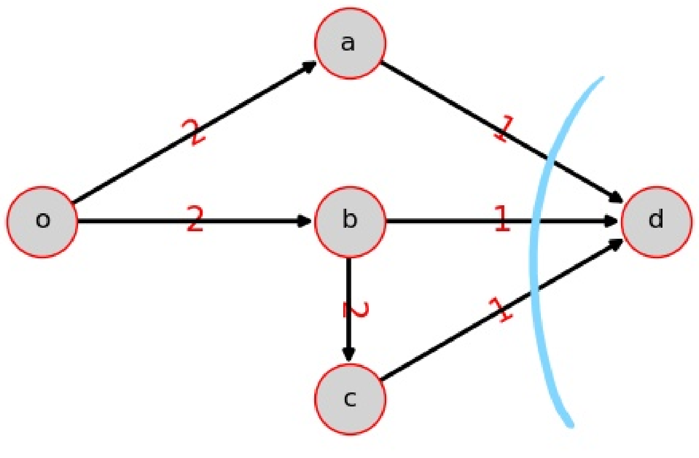

Figure 2: Original graph with the *min-cut* $c_{\mathcal{U}}$ highlighted in blue. It divides the network in two distinct partitions: $\mathcal{U}$ and $\mathcal{U}^c$

The graph in Fig. 2, it is possible to identify a *min-cut capacity* equal to three, which represents the bottleneck of the network, that divides the graph in two sets: $\mathcal{U} = \{o, a, b, c\}$ and $\mathcal{U}^c = \{d\}$. The capacity of the cut is given by the sum of the capacities of all the edges starting from node partition $\mathcal{U}$ and ending in the complementary one ($\mathcal{U}^c$).

This quantity is described by the Eq.1 .

$$c_{\mathcal{U}} := \sum_{i\in\mathcal{U}} \sum_{j\in\mathcal{U}^c} c_{ij} \tag{1}$$

According to the *Connectivity and Flow theory* (Section 3 of the theory lessons), the minimum aggregated capacity to be removed in order to have a non feasible flow from o to d coincides with the capacity associated with the *minimum cut* (shown in Fig. 2); therefore its value is equal to three.

2.2 1.b

In this exercise it is required to calculate the maximum aggregated capacity that can be removed from the links without affecting the *maximum throughput* from o to d.

In order to calculate this value, we have to study the *Maximum Flow Problem* associated with the graph first; then we can identify the links that are not saturated, so the ones which have a capacity greater than the actual flow passing through them.

We know from the theory that the *Maximum Flow* is equal to the *Minimum Cut Capacity* (which represents the bottleneck of the network), so it is equal to three (in the provided notebook it is visible the dictionary showing the flow across the graph). Knowing the amount of flow passing through the different edges allows to discover if there are any which are not saturated. In particular, the links (o, a) and (b, c) have both a residual capacity equal to 1; this means that it is possible to remove an *aggregated capacity* equal to 2 from the links in order to not affect the final throughput.

2.3 1.c

In the last part of Problem 1, we are given $x > 0$ extra units of capacity and we are required to distribute them in order to maximize the throughput of the network.

First, for every extra unit of capacity, we compute the *Minimum Cut*; we know by the theory that the cut that has been found is the bottleneck of the network, so it is necessary to distribute the extra capacity over the edges belonging to it.

In simple terms we follow an iterative approach in which, for every extra unit of capacity, we identify the bottleneck of the network and we try to increase it by adding a single unit of extra capacity to one of the edges passing through the cut. During this process, depending on the quantity

x, the *minimum cut* changes position and the associated *capacity*. Finally, we solve the *Maximum Flow Problem* associated to the new graph and by doing so we are able to study the throughput of the network as a function of the extra capacity x (this is show in Fig.3).

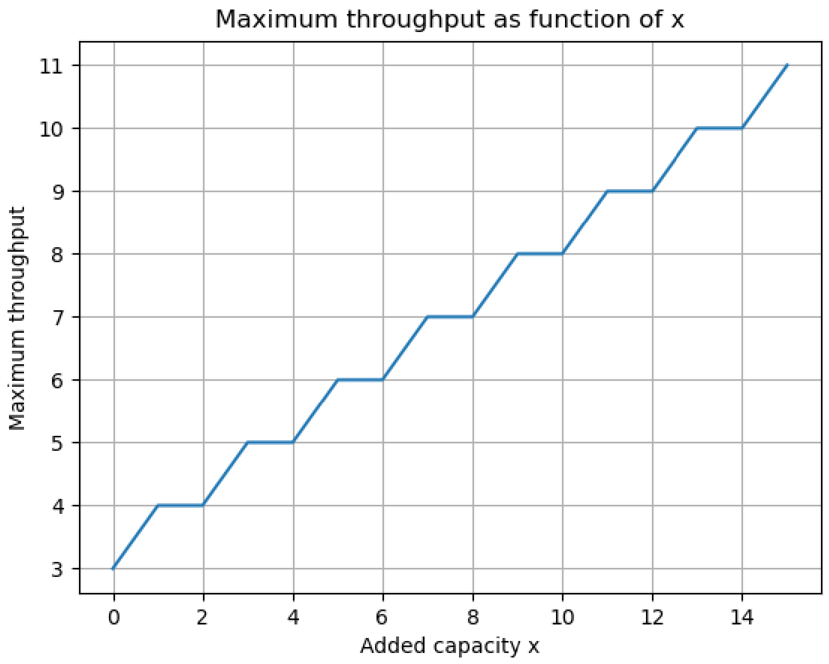

Figure 3: It is reported the *Maximum throughput* of the network as a function of the extra capacity x. We can notice that if no extra capacity is used, the *Maximum Flow* coincides with the *Minimum Cut Capacity* found in Section 2.1 (as expected thanks to the *Max-Flow Min-Cut Theorem*). We can see that the trend is of course increasing, but not linear. It is necessary to have a single extra unit of capacity in the original graph to increase the throughput up to 4, but to enhance it even further we need 2 more extra units of capacity each time.

3 Problem 2

In this problem we are giver a set of people $\{p1, p2, p3, p4\}$ and a set of books $\{b1, b2, b3, b4\}$. Each person p_i is interested in one or more book b_j, particular:

$$p1 \rightarrow \{b1; b2\};\ p2 \rightarrow \{b2; b3\};\ p3 \rightarrow \{b1; b4\};\ p4 \rightarrow \{b1; b2; b4\}$$

This relationship is represented by the directed graph in Fig. 4, on which the next tasks are performed. The problem may be formulated also with a bipartite graph (so with undirected edges); however, in the next steps it is necessary to convert it into a directed graph in order to address the next questions.

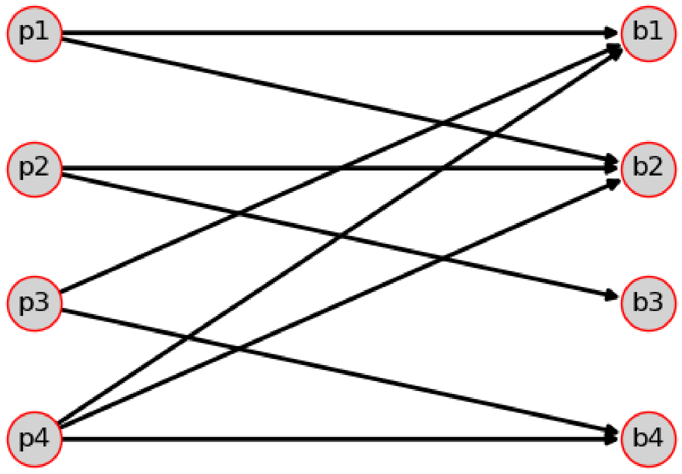

Figure 4: Directed graph representing the interests of people p_i on the different books b_j with: $i = \{1, 2, 3, 4\}$, $j = \{1, 2, 3, 4\}$.

3.1 2.a

The first task we have to address is to exploit the *Maximum Flow Problem* in order to find a perfect matching, if it exists.

First of all, in order to model a *Maximum Flow Problem*, it is necessary to add a source and a destination node (s and d respectively) as reported in Fig. 5; moreover, the capacity of each edge has been set equal to 1. The *source* node is connected to the subset of nodes related to the people, while the *destination* node (which will be the *sink* of the graph) is connected to the books .

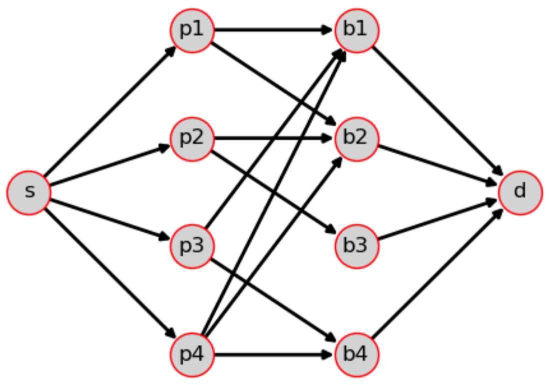

Figure 5: Original graph with the *source* and *destination* nodes in order to solve the associated *Maximum Flow Problem*. Each edge has a capacity equal to one.

As an interpretation of the *Maximum Flow Problem*, the result will give the flow passing through the edges (i, j). There are only two possible values of flows (due to the unitary capacity of the arcs):

$$f_{(i,j)} = \begin{cases} 1 & \text{if person } i \text{ will get book } j \\ 0 & \text{otherwise} \end{cases}$$

The *Minimum Cut Capacity* of the graph is equal to 4. Therefore, the bottleneck of the network will be equal to the *max-flow* (as we have already seen in Section 2). The solution of the *Maximum Flow Problem* is reported in Fig. 6, giving as result the following match:

$$p_1 \rightarrow b_2;\ p_2 \rightarrow b_3;\ p_3 \rightarrow b_1;\ p_4 \rightarrow b_4;$$

This match is also *perfect* because its cardinality is the maximal one; in other words, all the people p_i are satisfied by taking at least one book p_j in which they were interested in.

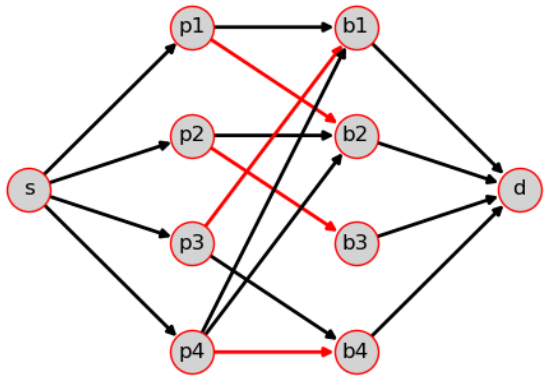

Figure 6: Network graph with the matching highlighted. The red links are referring to the ones with $f_{(i,j)} = 1$; for visual clarity we left black the edges connected to s and d (even though their flow is equal to one) in order to better emphasize the connections among the set of people and the set of books.

3.2 2.b

In this second part we know that there are different copies of the 4 books, in particular the vector representing the availability is: $(2; 3; 2; 2)$. The interests of the people are not changed, but this time they can take an arbitrary number of different books. We are required to establish how many books of interest can be assigned in total, exploiting the analogy with *Maximum Flow Problems*.

The problem remains the same, but this time we have some capacities which are not unitary; in particular:

- each (s, p_i) link capacity is given by the amount of different books in which a person p_i is interested in
- each (b_j, d) link capacity is given by the amount of copies of book b_j available
- each (p_i, b_j) link remains with unitary capacity since each person could take a single copy of every book is interested in

The same operations done in Section 3.1 are performed, giving as result of the *Maximum Flow Problem* the flow dictionary representing the number of different books that each person can take; this is summarized as:

$$\begin{gathered} p_1 \to \{b_2\}; \\ p_2 \to \{b2, b3\}; \\ p_3 \to \{b_1, b_4\}; \\ p_4 \to \{b_1, b_2, b_4\}; \end{gathered}$$

Then, the *Max-Flow Value* is 8 (equal to the *Minimum Cut Capacity*, as expected), which is the total amount of books that can be assigned. The only person non fully satisfied by is p_1, who wanted b_1 too, but the availability is not sufficient.

3.3 2.c

In this last part we are allowed to modify the availability of the books: we can sell a book and replace it with another copy of a different one. Practically, this means that we are free to add and remove a unit of capacity from one of the (b_j, d) edges. In other words it is required to try to balance demand and request of all the books, attempting to maximize the number of copies assigned.

in order to do so, we exploit the knowledge of balanced node and apply it in this particular scenario. A node is balanced if $w_i = w_i^-$, so if the *out-degree* equals the *in-degree*; this are respectively:

$$w_i = \sum_{j \in \mathcal{V}} W_{ij}, \quad \text{and} \quad w_i^- = \sum_{j \in \mathcal{V}} W_{ji}$$

In this specific problem all the nodes p_i are already balanced; so, balancing each remaining node b_j means satisfy completely all the people. For every node b_j it is calculated the total possible *in-flow* and *out-flow*; if these values are not equal, the *netflow* (equal to *in-flow* - *out-flow*)will be different from zero and the node can be balanced. We try to transfer this knowledge to the capacity measure, in particular: if the sum of the capacities of the in-going edges is equal to the sum of the capacities of the out-going edges, demand and offer are balanced. As result of this process we found that:

$$\begin{gathered} netflow_{b_1} = 1; \\ netflow_{b_3} = -1; \end{gathered}$$

This means that *book*$_1$ has an higher demand with respect to its offer, while there is an extra copy of *book*$_3$. As a conclusion, in order to maximize the number of assigned books it is necessary to sell a copy of *book*$_3$ and buy a copy of *book*$_1$. This allows to enhance the *Maximum Flow Value* up to 9; it is coherent with respect to what we discussed at the end of Section 3.2, where it has been highlighted that p_1 was not completely satisfied because one copy of *book*$_1$ was missing.

4 Problem 3

The last problem of this Homework involves a simplified representation of the highway network in Los Angeles, on which we apply different notions related to *Flows* and the associated *Optimization* (Chapter 3 and 4 of lecture notes). In this study we are asked to consider the node 1 as *source* and the node 17 as *target*. Given the graph in Fig.7 with its node-link incidence matrix B,

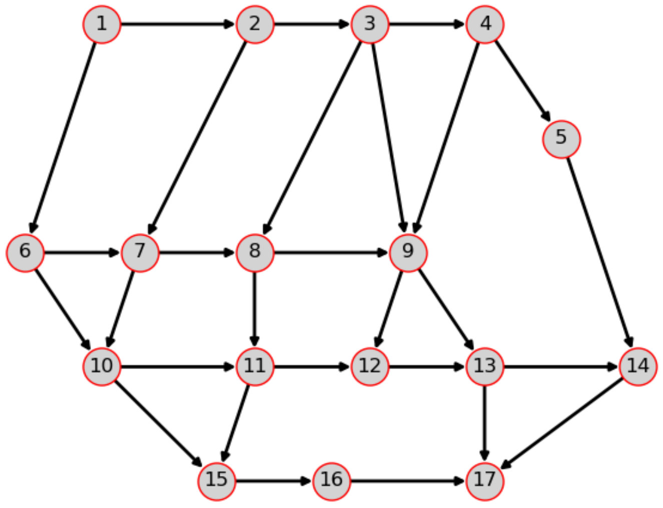

Figure 7: Graph representation of the Los Angeles highway.

for each edge we are given the: *flow* (f_e), *capacity* (c_e) and *traveltime* (l_e).

For each link it is also introduced a delay function as follows:

$$\tau_e\left(f_e\right) = \frac{l_e}{1 - f_e/c_e}, \quad 0 \leq f_e < c_e \tag{2}$$

For $fe \geq ce$, the value of $\tau_e\left(f_e\right)$ is considered as $+\infty$.

4.1 3.a

In this first task, it is required to find the shortest path between node 1 and 17; in order to do this, it is necessary looking for the fastest path in an empty network. Thanks to the *NetworkX* library, it is possible to find it with the *shortest_path()* method; the algorithm executed in order to find the result is the Dijkstra one.

Therefore, the node sequence that characterize the shortest path is the following:

$$\gamma^* = (1, 2, 3, 9, 13, 17)$$

and is also represented in Fig. 8; this is characterized by a total travel time equal to 0.533 h, which is approximately 32 m.

4.2 3.b

In this second exercise the goal is to calculate the *Maximum Flow* from node 1 to node 17. In order to do this, we have perform the same operations done in Sections 2, 3. After the execution of the algorithm, the result of the *Maximum Flow* is: $v^*_{1,17} = 22448$.

4.3 3.c

In this third part we are required to solve a linear algebra problem related to the flow in this network; in particular it is required to calculate the externa inflow ν that satisfies the *mass conservation* equation; 3

$$Bf = \nu \tag{3}$$

In the attached notebook it is possible to inspect the vector ν obtained as result.

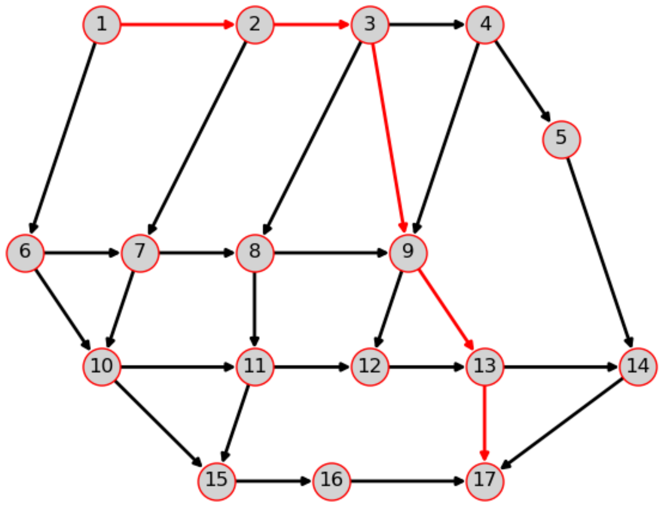

Figure 8: Los Angeles highway graph in which is highlighted the shortest path from node 1 to node 17.

4.4 3.d

From now on, we will use the *cvxpy* library in order to solve linear models. Moreover, in order to solve the next problems, we assume that the exogenous inflow in all the nodes is zero except for node 1, for which ν_1 has the same value computed in the Section 4.3 ($\nu_1 = 16806$), and node 17, for which $\nu_1 7 = \nu_1$.

In this exercise it is required to find the social optimum f^* considering the minimization of the following function:

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

Therefore the linear problem is:

$$\inf_{\substack{f\in\mathbb{R}_+^{\mathcal{E}} \\ Bf=\nu}} \sum_{e\in\mathcal{E}} \left(\frac{l_e c_e}{1-f_e/c_e} - l_e c_e\right).$$

This is the *System–Optimum Traffic Assignment Problem* (SO-TAP). The *Social Optimum Flow* obtained is reported in the attached notebook; it is characterized by an *Optimal Cost* equal to 25946.6.

4.5 3.e

After the *Social Optimum* f^*, we are required to calculate the *Wardrop Equilibium* $f^{(0)}$. In contrast with Section 4.4 in which there was a complete view of the system, in this exercise each driver will behave selfishly by choosing the path that considers the fastest one. This problem is called *User Optimal Transportation Problem* (UO-TAP) and would lead to a sub-optimal solution with respect to the one computed by solving the SO-TAP in Section 4.4. This sub-optimality can be measured by the Eq. 4 and is called *Price of Anarchy*.

$$\text{PoA}(0) = \frac{\begin{array}{c}\text{total cost/delay at}\\ \text{user optimum}\end{array}}{\begin{array}{c}\text{total cost/delay at}\\ \text{social optimum}\end{array}} = \frac{\sum_{e\in\mathcal{E}} f_e^{(0)}\tau_e\left(f_e^{(0)}\right)}{\sum_{e\in\mathcal{E}} f_e^*\tau_e\left(f_e^*\right)} \tag{4}$$

In order to find the *Wardrop Equilibrium Flow* $f^{(0)}$, the cost function changes and the linear problem becomes:

$$\min_{\substack{f\in\mathbb{R}_+^{\mathcal{E}} \\ Bf=\nu}} \sum_{e\in\mathcal{E}} \int_0^{f_e} \tau_e(s)ds.$$

The final *Wardrop Flow* is reported in the attached notebook and is characterized by a cost equal to 26292.96, which is higher that the *Social Optimum* one. As expected the selfish behaviour of each driver penalize the whole network performances; this can be seen by the *Price of Anarchy* which is greater than 1 and can be seen in Eq. 5.

$$\text{PoA}(0) = \frac{\begin{array}{c}\text{total cost/delay at}\\ \text{user optimum}\end{array}}{\begin{array}{c}\text{total cost/delay at}\\ \text{social optimum}\end{array}} = \frac{26292.96}{25943.6} = 1.0135 \tag{5}$$

4.6 3.f

In order to influence the behaviour of each driver, it is possible to introduce a toll ω_e on each edge (road); in this way, their perceived cost has one more component represented in addition to the travel time. This is represented by the money they have to spend in order to pass from link e too.

In this particular problem, we introduce a toll $\omega_e = f_e^* \tau_e'(f_e^*)$, where f_e^* is the flow at the system optimum calculated in Section 4.4. Therefore the total delay on a link e is $\tau_e(f_e) + \omega_e$; the goal is to calculate the *Wardrop Equilibrium* $f^{(\omega)}$.

In the attached notebook is reported the $f^{(\omega)}$; with the flow we obtained, the total delay of the system is equal to the one found by the SO-TAP solution. In fact, by computing the *Price of Anarchy* we obtain $POA = 1$. This is a concrete case in which adding tolls on the links can improve the network performance, even in case of selfish behaviour of the drivers.

4.7 3.g

In this last problem we change the cost of the system; in particular it is required to set it as the total additional delay with respect to the total delay in free flow. This is represented by Eq. 6.

$$\psi_e(f_e) = f_e(\tau_e(f_e) - l_e) \tag{6}$$

The new linear problem becomes:

$$\min_{f \in \mathbb{R}_+^{\mathcal{E}}} \sum_{e \in \mathcal{E}} f_e \left(\frac{l_e c_e}{1 - f_e / c_e} - l_e c_e \right) - f_e l_e.$$
$$Bf = \nu$$

The first task we are required to perform is to calculate the *System Optimum Flow* f^* for the new problem. In the attached notebook we found that the total cost is equal to 15095.5.

After this step, we have to construct a toll ω^* such that the *Wardroop Equilibrium Flow* $f^{(\omega^*)}$ coincides with f^*. In other words we solve the UO-TAP with the *System Optimum Flow* vector f^* obtained before and find the ω vector.

As result we obtained that the omega vector is different from zero along the path $\gamma = (1, 2, 7, 8, 11, 15, 16, 17)$ and is reported in Fig. 9.

The flow $f^{(\omega^*)}$ obtained at *Wardrop Equilibrium* with the constructed tolls is characterized by a $POA = 1$; this verifies that $f^* = f^{(\omega^*)}$, so the total delay values $\psi(f^*)$ and $\psi(f^{(\omega^*)})$ are equal as requested.

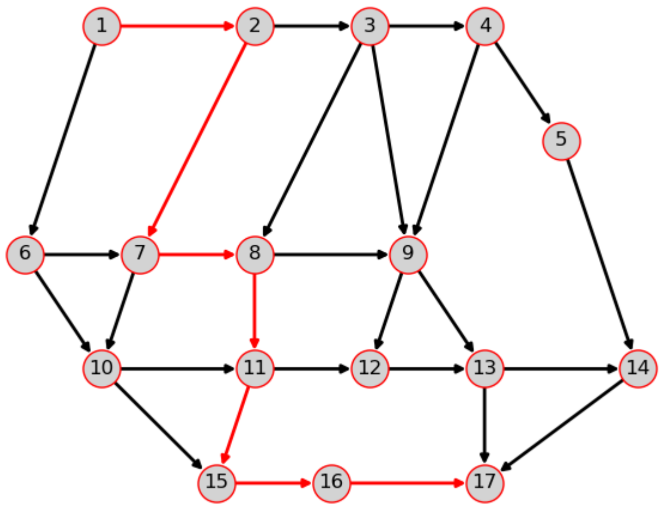

Figure 9: Graph representing the highway network of Los Angeles; the edges e colored in red are the ones with $\omega_e > 0$