Politecnico di Torino

Networks dynamics and Learning

Homework #1

Antonio Albanese, 282043

November 2021

In this homework I exchanged ideas and compared results with my colleague Nedescu Ionut Cosmin.

Please note that the report seems to be too long because text of the assignment was also reported to have simpler references to pictures and equations. Thank you.

Exercise 1

Consider unitary o-d network (i.e., integer whose integer units cannot be split) on the graph $\mathcal{G} = (\mathcal{V}; \mathcal{E})$ in Figure 1, and assume that each link l has integer capacity C_l.

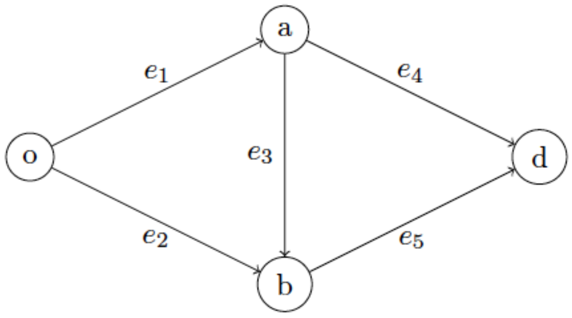


Figure 1

(a) What is infimum of the total capacity that needs to be removed for no feasible unitary flows from o to d to exist?

(b) Assume that the link capacities are

$$C_1 = C_4 = 3, \qquad C_2 = C_3 = C_5 = 2. \tag{1}$$

Where should 1 unit of additional capacity be allocated in order to maximize the feasible throughput from o to d? What is the maximal throughput?

(c) Consider link capacities (1). Where should 2 units of additional capacity be allocated in order to maximize the feasible throughput from o to d? Compute all the optimal capacity allocations for this case and the optimal throughput.

(d) Consider link capacities (1). Where should 4 units of additional capacity be allocated in order to maximize the feasible throughput from o to d? Compute all the optimal capacity allocations for this case. Among the optimal allocations, select the allocation that maximizes the sum of the cut capacities.

Answer.

(a) To answer this question we need to start from three concepts:

Definition 1 (Node-independent paths) *Two $i-j$ paths are* node-independent *if they share no intermediate node.*

Definition 2 (Link connectivity) *The* link-connectivity $c_{link}(i,j)$ *of a pair of nodes* (i,j) *is the maximum number of* link-independent $i-j$ *paths.*

Theorem 1 (Menger's Theorem) *The minimum number of links that have to be removed from a graph $\mathcal{G}$ in order for node j not to be reachable from node i equals* $c_{link}(i,j)$.

In the given graph we have $c_{link}(o,d) = 2$, meaning that we need to remove at least 2 links to have no feasible flow from o to d.

The infimum of the total capacity that needs to be removed for no feasible unitary flows from o to d to exist is

$$\min\{(\mathbf{C_1}+\mathbf{C_2}),(\mathbf{C_1}+\mathbf{C_5}),(\mathbf{C_4}+\mathbf{C_5}),(\mathbf{C_1},\mathbf{C_3},\mathbf{C_5})\}$$

(b) First of all we can redraw the graph with the given capacities (1) (Figure 2).

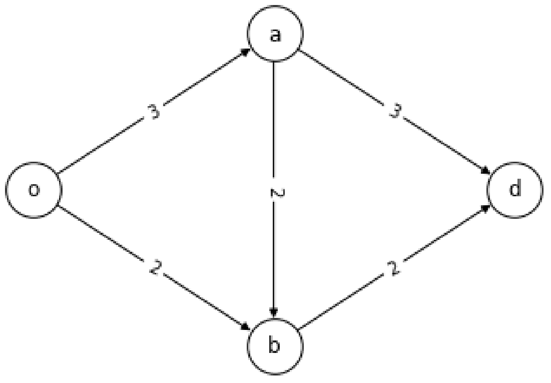


Figure 2: Redrawn graph with explicit capacities assigned to links

To answer this question we can use *max-flow min-cut theorem.* First we need a couple of definitions.

Definition 3 (Maximum throughput) *For a capacited multigraph $\mathcal{G} = (\mathcal{V};\mathcal{E};c)$ and two distinct nodes $o \neq d$ in $\mathcal{V}$, the maximum-flow problem is the constraint maximization problem*

$$\chi^*_{o,d} = \max_{\substack{\chi\geq 0\\ 0\leq f\leq c\\ Bf=\chi(\delta^{(o)}-\delta^{(d)})}} \chi$$

$\chi^*_{o,d}$ *is the* maximum throughput *from o to d.*

Definition 4 (Min-cut capacity) *We fix a multigraph $\mathcal{G} = (\mathcal{V}; \mathcal{E}; c)$ and two distinct nodes $o \neq d$ in V. The* ***min-cut capacity*** *of the network is the value*

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

Theorem 2 (Max-flow min-cut theorem) *Consider a multigraph $\mathcal{G} = (\mathcal{V}; \mathcal{E}; c)$ with a capacity vector $c > 0$, and two distinct nodes $o \neq d$ in $\mathcal{V}$. Then,*

$$\chi^*_{o,d} = c^*_{o,d}.$$

Theorem 2 tells us that the maximum throughput from o to d equals the min-cut capacity from o to d.
In the given graph we have the following cuts:

- $U = \{o, b, a\}, U^C = \{d\}$ -> $C_U = 5$
- $U = \{o, a\}, U^C = \{b, d\}$ -> $C_U = 7$
- $U = \{o, b\}, U^C = \{a, d\}$ -> $C_U = 5$
- $U = \{o\}, U^C = \{a, b, d\}$ -> $C_U = 5$

The minimum cut capacity is 5 and there are three cuts, whit that capacity.

- cut $U = \{o, b, a\}, U^C = \{d\}$,
$$C_U = c(b, d) + c(a, d) \tag{2}$$
- cut $U = \{o, b\}, U^C = \{a, d\}$,
$$C_U = c(o, a) + c(b, d) \tag{3}$$
- cut $U = \{o\}, U^C = \{a, b, d\}$,
$$C_U = c(o, a) + c(o, b) \tag{4}$$

Independently on which of those links we apply one more unit of capacity the **max throughput can not be improved**.

(c) Starting from what we know in **(b)** and equations (2), (3) and (4) we can optimize the choice of links on which to add capacities with the aim to increase the max-flow. With a recursive function it was possible to calculate all the optimal capacity allocations finding three possibilities:

- Adding 1 unit to (o, a) and 1 unit (a, d)
- Adding 1 unit to (o, a) and 1 unit (b, d)
- Adding 1 unit to (o, b) and 1 unit (b, d)

All solutions found give min-cut capacity equals to 6.
For example we could choose the configuration

$$\begin{cases} c(a,b) = 2 \rightarrow c(a,b) = 2, \\ c(a,d) = 3 \rightarrow c(a,d) = 3, \\ \mathbf{c(b,d) = 2 \rightarrow c(b,d) = 4}, \\ c(o,a) = 3 \rightarrow c(o,a) = 3, \\ \mathbf{c(o,b) = 2 \rightarrow c(o,b) = 4} \end{cases}$$

with

- $U = \{o,b,a\}, U^C = \{d\}$ -> $C_U = c(b,d) + c(a,d) = 6$
- $U = \{o,a\}, U^C = \{b,d\}$ -> $C_U = 7$
- $U = \{o,b\}, U^C = \{a,d\}$ -> $C_U = c(o,a) + c(b,d) = 6$
- $U = \{o\}, U^C = \{a,b,d\}$ -> $C_U = c(o,a) + c(o,b) = 6.$

Again, following the *max-flow min-cut theorem* we know that the **maximum throughput now is 6**.

(d) This part is similar to **(c)** and with the same recursive function it was possible to calculate all the optimal capacity allocations, resulting in six different possibilities:

1) Adding 2 units to (o,a) and 2 unit (a,d)
   - $U = \{o,b,a\}, U^C = \{d\}$ -> $C_U = 7$
   - $U = \{o,a\}, U^C = \{b,d\}$ -> $C_U = 9$
   - $U = \{o,b\}, U^C = \{a,d\}$ -> $C_U = 7$
   - $U = \{o\}, U^C = \{a,b,d\}$ -> $C_U = 7.$
2) Adding 2 units to (o,a), 1 unit to (a,d) and 1 unit to (b,d)
   - $U = \{o,b,a\}, U^C = \{d\}$ -> $C_U = 7$
   - $U = \{o,a\}, U^C = \{b,d\}$ -> $C_U = 8$
   - $U = \{o,b\}, U^C = \{a,d\}$ -> $C_U = 8$
   - $U = \{o\}, U^C = \{a,b,d\}$ -> $C_U = 7.$
3) Adding 2 units to (o,a) and 2 units to (b,d)
   - $U = \{o,b,a\}, U^C = \{d\}$ -> $C_U = 7$
   - $U = \{o,a\}, U^C = \{b,d\}$ -> $C_U = 7$
   - $U = \{o,b\}, U^C = \{a,d\}$ -> $C_U = 9$
   - $U = \{o\}, U^C = \{a,b,d\}$ -> $C_U = 7.$
4) Adding 1 unit to (o,a) and 1 unit to (o,b), 1 unit to (a,d) and 1 unit to (b,d)
   - $U = \{o,b,a\}, U^C = \{d\}$ -> $C_U = 7$
   - $U = \{o,a\}, U^C = \{b,d\}$ -> $C_U = 9$
   - $U = \{o,b\}, U^C = \{a,d\}$ -> $C_U = 7$
   - $U = \{o\}, U^C = \{a,b,d\}$ -> $C_U = 7.$
5) Adding 1 unit to (o,a) and 1 unit to (o,b) and 2 units to (b,d)

- $U = \{o, b, a\}, U^C = \{d\}$ -> $C_U = 7$
- $U = \{o, a\}, U^C = \{b, d\}$ -> $C_U = 9$
- $U = \{o, b\}, U^C = \{a, d\}$ -> $C_U = 7$
- $U = \{o\}, U^C = \{a, b, d\}$ -> $C_U = 7$.

6) Adding 2 units to (o, b) and 2 units to (b, d)
   - $U = \{o, b, a\}, U^C = \{d\}$ -> $C_U = 7$
   - $U = \{o, a\}, U^C = \{b, d\}$ -> $C_U = 9$
   - $U = \{o, b\}, U^C = \{a, d\}$ -> $C_U = 7$
   - $U = \{o\}, U^C = \{a, b, d\}$ -> $C_U = 7$.

In all the configurations shown the cut capacities sum up to 30, so we can choose any of them.

For example choosing option 4:

$$\begin{cases} c(a,b) = 2 \to c(a,b) = 2, \\ \mathbf{c(a,d) = 3 \to c(a,d) = 4}, \\ \mathbf{c(b,d) = 2 \to c(b,d) = 3}, \\ \mathbf{c(o,a) = 3 \to c(o,a) = 4}, \\ \mathbf{c(o,b) = 2 \to c(o,b) = 3} \end{cases}$$

The sum of cuts capacities is 30. And following the *max-flow min-cut theorem* **the maximum throughput is 7**.

Exercise 2

Consider the following problem. There are a set of people (p_1, p_2, p_3, p_4) and a set of books (b_1, b_2, b_3, b_4). Each person is interested in a subset of books, specifically

$$p_1 \to (b_1, b_2), \quad p_2 \to (b_2, b_3), \quad p_3 \to (b_1, b_4), \quad p_4 \to (b_1, b_2, b_4) \tag{5}$$

(a) Represent the interest pattern by using a simple bipartite graph.

(b) Exploit max-flow problems to establish whether there exists a perfect matching that assigns to every person a book of interest. If a perfect matching exists, find at least a perfect matching.

(c) Assume now that there are multiple copies of book, 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. The only constraint is that each person can not take more copies of the same book. Use the analogy with max-flow problems to establish how many books of interest can be assigned in total.

(d) Starting from point (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?

Answer.

(a) **Definition 5 (Bipartite graph)** *A simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ is bipartite if and only if every circuit in $\mathcal{G}$ has even length.*

In other words a bipartite graph is a graph whose nodes can be divided into two disjoint and independent sets $\mathcal{V}^{(1)}$ and $\mathcal{V}^{(2)}$ such that every edge connects a node in $\mathcal{V}^{(1)}$ to at least one in $\mathcal{V}^{(2)}$, and there are no edges between nodes of the same set. Given the relations in (5) we can draw a bipartite graph as in Figure 3.

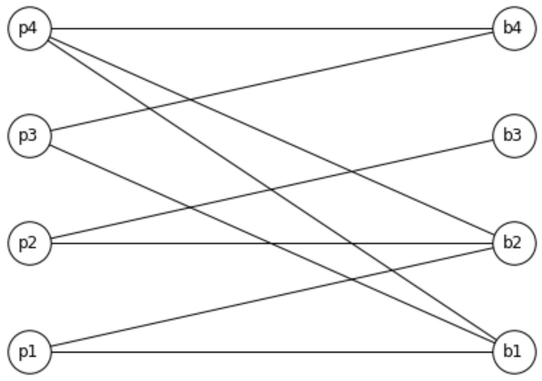
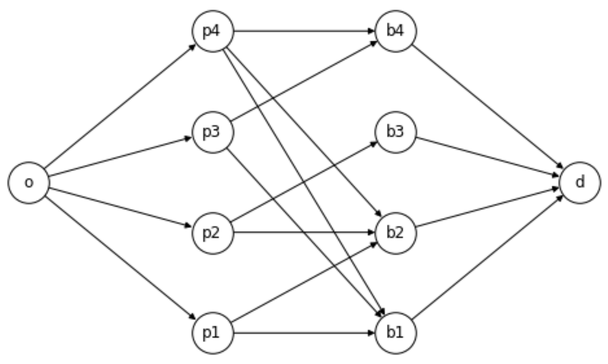


Figure 3: Relations in (5)
represented as simple bipartite graph.

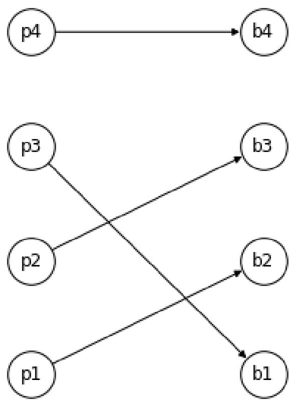
(b) **Definition 6 (Matching)** *A matching or independent link set in an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ without self loops is a subset of links $\mathcal{M} \in \mathcal{E}$ that have no node in common.*

Definition 7 (Perfect matching) *A matching is* ***perfect*** *(sometimes also called complete) if all nodes are matched.*

To exploit the max-flow problems to find a perfect matching (so to use the Hall's Marriage Theorem) we need to build the corresponding capacited directed multigraph of the given graph adding to the graph a source node o and a sink node d, and giving each link capacity 1 (Figure 4a).



(a) Graph redrawn with added nodes and directed edges



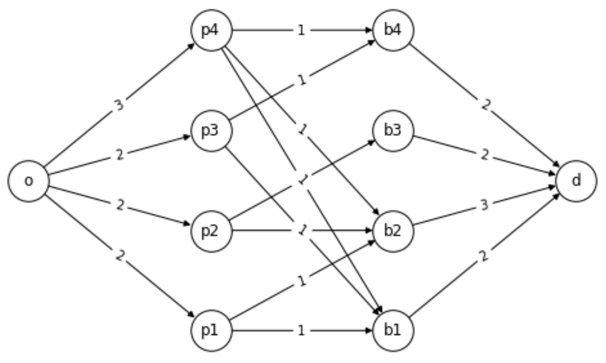
(b) The perfect matching found

Figure 4

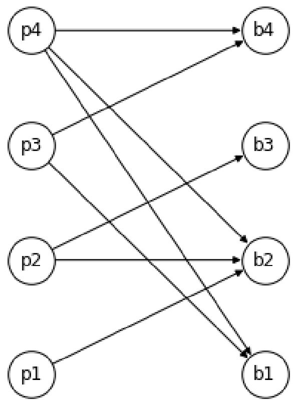
Using the max-flow algorithm it was possible to find a perfect matching, depicted in Figure 4b.

(c) We can start from the capacited multigraph found in **(b)**. We know that people have multiple preferences of books, and this time we can assign more copies of a single book, but not at the same person. We can modify the graph in Figure 4a obtaining the graph in Figure 5a, in this way:

- Set capacity of edges $(o, p_i) = w_{p_i}$, where w_{p_i} is the *out degree* of the node p_i, meaning the number of different preferences of p_i
- Set capacity of existing edges (p_i, b_j) to 1, meaning that only one copy of the same book can be assigned to the same person
- Set capacity of edges (b_j, d) equals to the number of different copies of each book that can be assigned.



(a) Graph redrawn with updated capacities



(b) The best assignment found

Figure 5

In this way **the maximum number of copies that can be assigned equals the max-flow between o and d** that is **8**. (Figure 5b)

(d) The goal at this point is to maximize the number of assigned books, in other words try to satisfy all people preferences.
Because of the manner in which we built the capacited graph (Figure 5a) in **(c)** we know that w_{p_i}, the *out degree* of each node p_i, is the number of different books choosen by p_i, and it flows directly that $w^{-}_{b_j}$, the *in degree* of each node b_j is the total number of request for each book, while w_{b_j}, the *out degree* of each node b_j, is the total number of copies that can be assigned for each book.

To miximize the number of assigned book and satisfy all users, we need that $w^{-}_{b_j} = w_{b_j}$.

- $b_1 \rightarrow w^{-}_{b_1} = 3, \quad w_{b_1} = 2 \Rightarrow$ **buy 1 copy of** b_1
- $b_2 \rightarrow w^{-}_{b_2} = 3, \quad w_{b_2} = 3$
- $b_3 \rightarrow w^{-}_{b_3} = 1, \quad w_{b_3} = 2 \Rightarrow$ **sell 1 copy of** b_3
- $b_4 \rightarrow w^{-}_{b_4} = 2, \quad w_{b_j} = 2$

In this way we can assign **9** copies in total.

Exercise 3

We are given the highway network in Los Angeles, see Figure 2. To simplify the problem, an approximate highway map is given in Figure 3, covering part of the real highway network. The node-link incidence matrix B, for this traffic network is given in the file *traffic.mat.* The rows of B are associated with the nodes of the network and the columns of B with the links. The i-th column of B has 1 in the row corresponding to the tail node of link e_i and (-1) in the row corresponding to the head node of link e_i. Each node represents an intersection between highways (and some of the area around).

Each link $e_i \in \{e_1, ..., e_{28}\}$, has a maximum-flow capacity C_{e_i} . The capacities are given as a vector C_e in the file *capacities.mat.* Furthermore, each link has a minimum travelling time, 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 in the file *traveltime.mat.* These values are simply retrieved by dividing the length of the highway segment with the assumed speed limit 60 miles/hour.

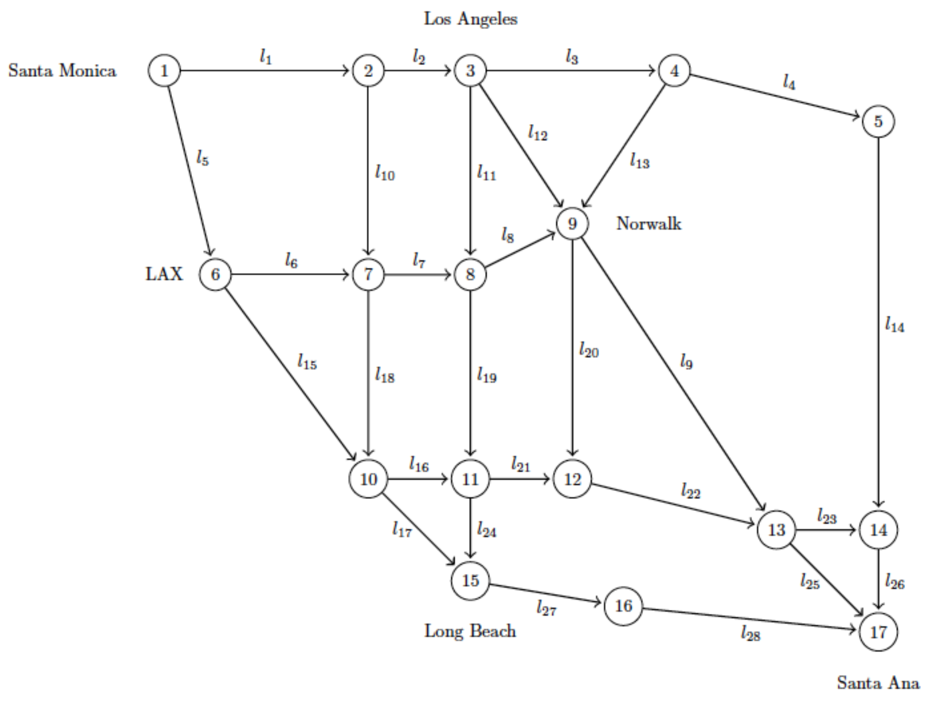


Figure 6: Some possible paths from Santa Monica (node 1) to Santa Ana (node 17).

For each link, we introduce the delay function

$$d_e(f_e) = \frac{l_e}{1 - f_e/C_e}, \quad 0 \le f_e \le C_e \tag{6}$$

For $f_e \ge C_e$, the value of $d_e(f_e)$ is considered as $+\infty$.

(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.

(b) Find the maximum-flow between node 1 and 17.

(c) Given the flow vector in *flow.mat*, compute the external inflow ν satisfying $Bf = \nu$. In the following, we assume that the exogenous inflow is zero in all the nodes except for node 1, for which ν_1 has the same value computed in the point (c), and node 17, for which $\nu_{17} = -\nu_1$.

(d) Find the social optimum f^* with respect to the delays on the different links $d_e(f_e)$. For this, minimize the cost function

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

subject to the flow constraints.

(e) 1. Find the Wardrop equilibrium f(0). For this, use the cost function

$$\int_0^{f_e} d_e(s)ds$$

2. Introduce tolls, such that the toll on link e is $\omega_e = f_e^* d_e'(f_e)$, where f_e is the flow at the system optimum. Now the delay on link e is given by $d_e(f_e) + \omega_e$. Compute the new Wardrop equilibrium $f^{(\omega)}$. What do you observe?

(f) Instead of the total delay, let the cost be the total additional delay compared to the total delay in free flow be given by

$$c_e(f_e) = f_e(d_e(f_e) - l_e)$$

subject to the flow constraints. Compute the system optimum f^* for the costs above. Construct tolls $\omega_e, e \in \mathcal{E}$ such that the new Wardrop equilibrium with the constructed tolls $f^{(\omega^*)}$ coincides with f^*. Compute the new Wardrop equilibrium with the constructed tolls $f^{(\omega^*)}$ to verify your result.

Answer.

(a) To find the short path in the graph we use several algorithms defined in *networkx library* or we can use the network flow optimization. We have a linear problem, so the solution is to allocate a unitary flow on a convex combination of optimal paths. As a result we will get an optimal flow from which we can deduce the shortest path (Non-zero component of the optimal flow).

We need to set the optimization problem:

- **Problem data:**
  - B = *incidence matrix of the graph* (given in the file traffic.mat)
  - $\nu = [1, 0, ..., 0, -1]$, *the exogneous net flow of the graph* (1 unit enters node 1 and 1 unit exits node 17)
  - l = *coefficients of the linear terms of the objective function* (length of the link, in this case we use travel times from the file traveltime.mat)
- **Variables:**
  - f = *flow vector* (in this variable we will find the optimal flow for the graph)
- **Constraints:**
  - $Bf = \nu$ (mass conservation constraint)
  - $f \geq 0$ (flow components must be ≥ 0)
- **Objective function:**
  - $\min \sum f_e l_e$

The result is the minimum total travel time, in this case $\simeq 0.533$, and from the optimum flow vector we can read the shortest path found, in this case

$$\gamma_{\mathbf{shortest}} = (\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_{12}}, \mathbf{e_9}, \mathbf{e_{25}})$$

(b) Starting from the incidence matrix B and the capacities given we can build the given graph and find the max-flow between node 1 and node 17 using the default function provided by *networkx library*. In this case we obtain

$$max - flow = 22448$$

$$f = [8741, 8741, 0, 0, 13707, 4624, 4624, 4624, 6297, 0, 0, 8741, 0, 0,$$
$$9083, 825, 8258, 0, 0, 7068, 825, 7893, 3835, 0, 10355, 3835, 8258, 8258]$$

(c) For the law of mass conservation we can find ν as

$$\nu = Bf$$

Where B is the incidence matrix in the file traffic.mat and f is the given flow in the file flow.mat and ν is the exogenous net flow. The external in-flow is the positive part of ν, in this case

$$[v_i]_+ = [16806, 8570, 19448, 4957, 0, 4768, 413, 0, 0, 1169, 0, 0, 0, 0, 0, 0, 0]$$

(d) The social optimum flow in a traffic network is the solution of a problem that want to allocate flow on the network in a way such that all users have the minimum possible delay, so trying to minimize the total delay on the network. This kind of problem are also referred as SO-TAP.
Again we can define an optimization problem to solve this part.
Setup:

- **Problem data:**
  - B = *incidence matrix of the graph* (given in the file traffic.mat)
  - $\nu = [16806, 0, ..., 0, -16806]$*, the exogneous net flow of the graph* (16806 units enters node 1 and 16806 units exits node 17, from ν found in (c))
  - l = *coefficients of the objective function* (lengths of the links, in this case we use travel times from the file traveltime.mat)
  - c = *coefficients of the objective function* (capacities of the links, in this case we use capacities from the file capacities.mat)
- **Variables:**
  - f = *flow vector* (in this variable we will find the optimal flow for the graph)
- **Constraints:**
  - $Bf = \nu$ (mass conservation constraint)
  - $f \geq 0$ (flow components must be ≥ 0)
  - $f \leq c$ (flow components must be $\leq$ capacity of the link)
- **Objective function:**
  - $$\min \sum_{e\in\mathcal{E}} f_e d_e(f_e) = ... = \min \sum_{e\in\mathcal{E}} \frac{l_e C_E}{1 - f_e/C_e} - l_e C_e$$

    where f is the optimum flow vector to be found, and d_e is the delay function defined in (6)

The aim here is to minimize the total travel time between node 1 and node 17, using as delay the delay function (6).

The optimization problem gives as a result the system optimum travel time, while in f we can find the system optimum flow that gives the optimum travel time.

Results obtained are:

$$f^* = [6642, 6059, 3132, 3132, 10164, 4638, 3006, 2543, 3132, 583, 0, 2927, 0, 3132, 5525, 2854, 4886, 2215, 464, 2338, 3318, 5656, 2373, 0, 6414, 5505, 4886, 4886]$$

$$\textit{Total travel time at system optimum} = 25944$$

(e) 1. The *Wardrop equilibrium*, is a user equilibrium in traffic networks that essentially take into account the self determination of users in the network.
i.e.: considering drivers, none of the users would choose a sub-optimal route (i.e., an o-d path with larger delay) if a better route is available.

We can find the Wardrop equilibrium again solving an optimization problem, but this time the cost function to minimize is:

$$\int_0^{f_e} d_e(s)ds \tag{7}$$

We can setup the problem as the one explained in **(d)** replacing the objective function with:

$$\min \sum_{e\in\mathcal{E}} \int_0^{f_e} d_e(s)ds = ... = \min \sum_{e\in\mathcal{E}} -l_e C_e \log\left(1 - \frac{f_e}{C_e}\right)$$

Solving the problem we find:

$$f^{(0)} = [6716, 6716, 2367, 2367, 10090, 4645, 2804, 2284, 3418, 0, 177, 4171, 0, 2367,\\ 5445, 2353, 4933, 1842, 697, 3036, 3050, 6087, 2587, 0, 6919, 4954, 4933, 4933]$$

With $f^{(0)}$ we can calculate the total travel time at Wardrop equilibrium that is:

$$\textit{Total travel time at Wardrop equilibrium} = 26293$$

2. One way to influence the self determination of users captured by the Wardrop equilibrium is to add tolls $w_e > 0$ intended as a price users have to pay to travel a specific link, in this way the cost/delay perceived by users become:

$$w_e + d_e(f_e)$$

We define the tolls as $w_e = f_e^* d_e'(f_e^*)$ again we can use the same strategy as in **(e.1)** to find the flow at Wardrop equilibrium.

We replace the objective function with:

$$\min \sum_{e\in\mathcal{E}} \left(\int_0^{f_e} d_e(s)ds + f_e w_e\right) = ... = \min \sum_{e\in\mathcal{E}} \left[-l_e C_e \log\left(1 - \frac{f_e}{C_e}\right) + \frac{l_e C_e}{(C_e - f_e^*)^2} f_e\right]$$

Result obtained are:

$$f^{(w)} = [6642, 6059, 3132, 3132, 10163, 4638, 3006, 2542, 3131, 583, 0, 2926, 0, 3132, 5524, 2854, 48$$

And the total travel time at Wardrop equilibrium with tolls is:

$$\textit{Total travel time at Wardrop equilibrium with tolls} = 25944$$

The interesting fact here is that the t.t.t. (total travel time) with tolls is the same as t.t.t at system optimum.

(f) Now we have a new delay definition as the total additional delay compared to the total delay in free flow, given by

$$c_e(f_e) = f_e(d_e(f_e) - l_e)$$

We can compute the optimum flow f^* as done in **(d)**, using as objective function:

$$\min \sum_{e \in \mathcal{E}} f_e \left(d_e(f_e) - l_e\right) = ... = \min \sum_{e \in \mathcal{E}} \left[\frac{l_e C_E}{1 - f_e/C_e} - l_e C_e\right]$$

As result we find the total cost at system optimum

$$\textit{Total cost at system optimum} = 15096$$

Now we want to define tolls in such a manner that the flow at Wardrop equilibrium with tolls $f^{(w^*)}$ coincides with the system optimum flow f^*, this lead also to the total cost at Wardrop equilibrium with tolls coincides with the total cost at system optimum.
Essentially we are talking about finding tolls such that the *Price of Anarchy (PoA)* is 1.

Definition 8 (Price of anarchy) *The price of anarchy associated to a Wardrop equilibrium $f^{(0)}$ is*

$$PoA(0) = \frac{\sum\limits_{e \in \mathcal{E}} f_e^{(0)} d_e(f_e^{(0)})}{\min\limits_{\substack{f \geq 0 \\ Bf = \chi(\delta^{(o)} - \delta^{(d)})}} \sum_{e \in \mathcal{E}} f_e^* d_e(f_e^*)}$$

and can be interpreted as the ratio between the total delay at the Wardrop equilibrium and the minimum possible total delay (i.e. at system optimum).

To define such tolls we need two more concepts:

Theorem 3 *For a multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $o \neq d$ be two nodes such that d is reachable from o. Let each link e be equipped with a non decreasing differentiable delay function d_e, such that every cycle in $\mathcal{G}$ contains a link e with $d_e(0) > 0$. Let χ in $(0, c^*_{(o,d)})$ be a feasible throughput and let w in $\mathbb{R}_+^{\mathcal{E}}$ be a vector of prices. Then,*

(i) [...]

(ii) if f^* is the solution of a network flow optimization problem with convex nondecreasing costs $\phi_e(f_e)$, and

$$\omega_e^* = \phi_e'(f_e) - d_e(f_e^*), \quad e \in \mathcal{E}$$

then

$$f^{(w^*)} = f^*$$

Corollary 3.1 *Let the assumptions of Theorem 3 be satisfied and additionally let the delay functions $d_e(f_e)$ be convex. Let f^* be a solution of the SO-TAP, and let the link tolls be chosen as*

$$w_e = f_e^* d_e'(f_e^*)$$

Then, the Wardrop equilibrium flow $f^{(0)}$ coincides with the system optimum flow f^*.

Theorem 3 and especially Corollary 3.1 say that to have the system optimum flow coincides with the Wardrop flow with tolls we must define the tolls as

$$w_e = f_e^* d_e'(f_e^*)$$

We can also notice that this is the manner in which tolls were defined in **(e2)**, and in fact the total cost in **(e2)** coincides with the total cost at system optimum found in **(e1)**.

Proceeding in this way, we solve the optimization problem with objective funtion:

$$\min \sum_{e \in \mathcal{E}} f_e \left(\int_0^{f_e} d_e'(s) ds + f_e f_e^* d_e'(f_e^*) \right) = ... =$$

$$\min \sum_{e \in \mathcal{E}} \left[-l_e C_e \log\left(1 - \frac{f_e}{C_e}\right) - l_e f_e + \frac{l_e C_e}{(C_e - f_e^*)^2} f_e \right]$$

where $f_e^* d_e'(f_e^*) = w_e$ are tolls defined according Corollary 3.1 .

Solving the problem we obtain:

$$\textit{Total cost with defined tolls} = 15096 = \textit{Total cost at system optimum}$$

that gives

$$PoA(w) = \frac{\text{tot. cost with defined tolls}}{\text{tot. cost at system optimum}} = 1$$

as required.