Homework 1

Grandi Francesco *Student id: 306272*
exchanged idea with Elisa Feraud for exercise 1.c

I. Exercise 1

As the first step of this exercise, the graph described was defined using NetworkX libraries. Edges, Vertexes, and capacities are plotted. The resulting graph is a simple directed graph.

A. Exercise 1.A

To find the minimum aggregate capacity that needs to be removed for no feasible flow from o to d to exist an interpretation of the min-cut is used. In particular, through the function **minimum_cut** it is possible to visualize the value of the minimum cut and the sets of vertexes resulting from it.

- Minimum cut found: $\mathcal{U} = \{a, b, c, d\}$
- Minimum cut capacity $\mathcal{C}_{\mathcal{U}} = C_1 + C_4 + C_6 = 3$

To not have any feasible flow between o and d is sufficient to remove the edges crossed by the minimum cut. By decreasing the minimum cut of three a new minimum cut will be created of capacity 0. The min-cut theorem states: the maximum flow through any network from a given source to a given sink is exactly equal to the minimum sum of a cut. This implies a new maximum capacity of 0 and no feasible flow between o and d.

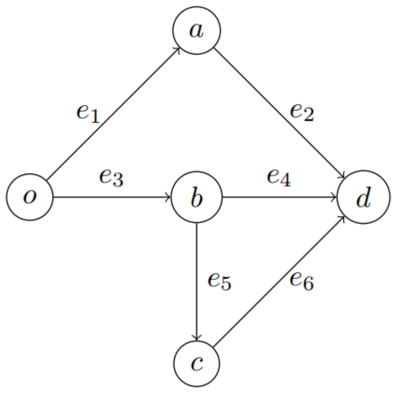

Fig. 1: $G = (V, E)$

B. Exercise 1.B

As the first step, a function called **maxCap** is defined. This function is used to retrieve the maximum capacity of one edge between all edges of the graph. Afterward, while iterating over every edge of the graph a unit of capacity is subtracted to a maximum given by the value found through the **maxCap**, and the minimum capacity is computed. If it is the same as the minimum capacity computed before then a new graph with one less unit of capacity is saved. Thanks to this algorithm every possible new capacity distribution of the graph is tested and the maximum capacity that can be removed is obtained. The result is:

- total unit of capacity removed: 2
- 1 unit of capacity removed from 'b-c'
- 1 unit of capacity removed from 'o-a'

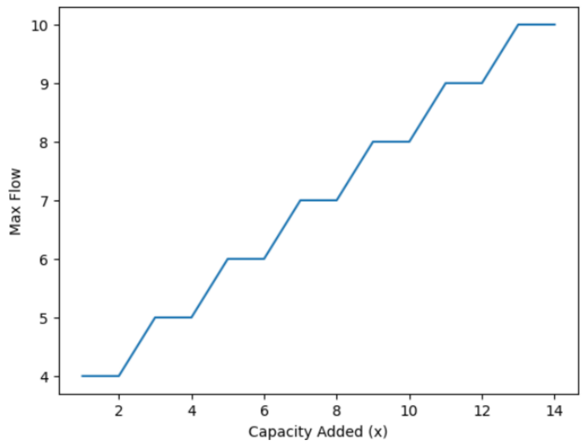

Fig. 2: Maximum flow plotted against capacity added

C. Exercise 1.C

To solve this exercise two main functions are used.

- **findEdgesMinCut**: returns the edges crossed by a minimum cut of the graph and
- **shortest_paths_edges**: return the edges of the shortest path

The first function is used to guarantee that the capacity added to the graph will increase a minimum cut removing a bottleneck of the graph in doing so.

The shortest_paths_edges function is implemented to take into account the fact that increasing the capacity of an edge of the shortest path avoids wasting in future iterations capacity units on one edge of a longer path reaching a sub optimal result in doing so.

Subsequently, the algorithm iterate on the intersection of the edges returned by the above-defined functions. At the beginning of the iteration, an edge capacity is increased and the new minimum cut is computed. To end the loop and assign the capacity one of the following condition must be satisfied:

- The new minimum cut is greater than the old one
- the last edge belonging to the edge intersection is examined

II. Exercise 2

In this exercise, we consider a matching problem where we have a set of people $\mathcal{P} = \{p1, p2, p3, p4\}$ and a set of books $\mathcal{B} = \{b1, b2, b3, b4\}$. Each person has different interests in a book. The relationship can be modeled through a simple bipartite graph where $e = (p_i, b_j) \in \mathcal{E}$ represents the interest of person i in book j.

A. Exercise 2.A

To find the perfect matching between P and B the Min-Cut Max-Flow theorem is exploited. The first step is to assign proper edge capacities. In particular, we model a new graph with edge capacity = 1 $\forall$e $\in \mathcal{E}$. Afterwards two more vertexes to $\mathcal{V}$ are added:

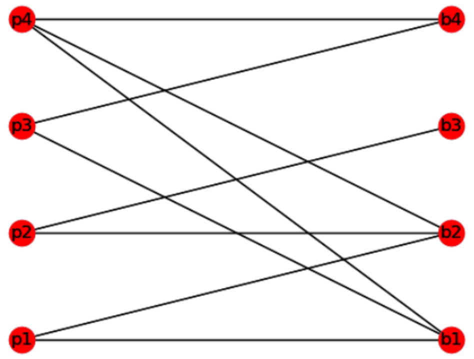

Fig. 3: $\mathcal{G} = (\mathcal{P} \cup \mathcal{B}, \mathcal{E})$

- o: The vertex through which the external flow flows in
- b: The vertex through which the flow exits the network

We then proceed adding links $(o, p_i), \forall p_i \in \mathcal{P}$, and links $(b_i, d), \forall b_i \in \mathcal{B}$. All the above-mentioned links will have an infinite capacity assigned. After calculating the minimum cut of the network it is possible to see that $|P| = |B| = 4$ which is a sufficient condition to have a perfect matching between $\mathcal{P}$ and $\mathcal{B}$. In particular:

- $(p1 \rightarrow b2)$
- $(p2 \rightarrow b3)$
- $(p3 \rightarrow b4)$
- $(p4 \rightarrow b1)$

B. Exercise 2.B

Let's change the capacity of the final edges to mimic the new number of books available. To model multiple copies of a book, a new capacity of edge (b_i, d) equal to the number of copies of b_i is added. Additionally, another capacity of edge (o, p_j) is added equal to the number of books they are interested in. One person will be satisfied if they have one copy of each book of their interest. The capacity of edge $(p_j,\ b_i)$ remains 1 as the person will only get 1 copy of the same book. Finally, the minimum cut is computed; the value obtained is 8, which means that 8 of the 9 books available are assigned. In particular, we have that:

- $p1 \rightarrow b2$
- $p2 \rightarrow b2$, $p2 \rightarrow b3$
- $p3 \rightarrow b4$, $p3 \rightarrow b1$
- $p4 \rightarrow b1$, $p4 \rightarrow b4$, $p4 \rightarrow b2$

C. *Exercise 2.C*

The library can sell 1 copy of a book and buy a copy of another one. Therefore, based on the results above, we can see that P_1 is interested in B_1 but cannot take it. Instead, B_3 has only 1 person interested in it but there is more than one copy available in the library (3). It is now easy to deduce that the library should buy another copy of B_1 and sell a copy of B_3. As expected the new maximum capacity obtained is 9 which is equal to the total number of books available.

III. Exercise 3

In exercise three a graph a simple directed graph representing the highway network in Los Angeles. More specifically the following data is given:

- Node-link **incidence matrix** $\mathcal{B} \in \{1, 0, +1\}$ of dimensions $\mathcal{V} \times \mathcal{E}$
- **Capacity vector**: describes the maximum traffic that every road can support
- **Length vector**: describes the time needed to cross a path
- **Flow vector**: represents a possible flow in the network

A. *Exercise 3.A*

To find the shortest path a flow optimization problem of the graph is considered. In particular through CVXPY, a modeling language for convex optimization problems the following one is solved:

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

having

$$0 < f < C \qquad Bf = v$$

- v is the flow vector with all 0's as values except in the origin node and the destination node, where it is equal respectively to 1 and -1.

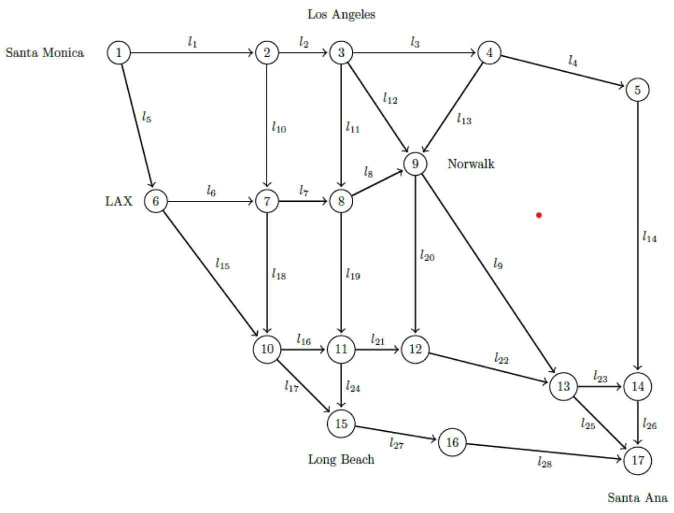

Fig. 4: Some possible paths from Santa Monica (node 1) to Santa Ana (node 17)

- f is a vector variable
- $\psi_e(f_e) = l_e f_e$ is the cost function assigned to each link e

The result is the list of zeros and ones representing the edges crossed (in case of 1) and not crossed (in case of 0). It was obtained as a result a path formed by the following edges: (1, 2), (2, 3), (3, 9), (9, 13), (13, 17).

B. *Exercise 3.B*

The maximum flow is computed by solving the same problem as before many times while increasing the exogenous flow entering vertex 1 by one unit. Following this procedure is sufficient because whenever the problem doesn't have a solution a "None" value is returned. If that is the case, since there is always a solution for flows less than the maximum flow feasible, it's enough to decrease the first non-feasible input flow by one to obtain the maximum value. The maximum flow found is 22448.

C. *Exercise 3.C*

To obtain ν it is enough to compute Bf. The result is ν = 6806 8570 19448 4957 -746 4768 413 -2 -5671 1169 -5 -7131 -380 -7412 -7810 -3430 -23544.
In the following exercises $v(0)$ will be set to 6806 (the exogenous flow entering the first vertex) while

$v(17)$ to - 6806 as the flow entering the network will outflow from it. In conclusion, all the other values of v will be set to 0 as no exogenous flow is entering or exiting from them.

D. Exercise 3D

To find the social optimum we have to solve a so-called SO-TAP problem (System-Optimum Traffic Assignment problem). That is equivalent to routing the flow in the network through the supervision of a central entity, constantly aware of that state of the network and capable of optimizing the flowing of cars in the best possible way. To do so is sufficient to solve the following problem:

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

having

$$0 < f < C \quad\quad Bf = v$$

$\tau_e(f_e)$ is the delay function defined in the exercise. By solving this problem (whose constraints are the mass conservation and the non-negativity flow) the f* obtained is: 6642, 6059, 3132, 3132, 10164, 4638, 3006, 2543, 3132, 583, 0, 2926, 0, 3132, 5525, 2854, 4886, 2215, 464, 2338, 3318, 5656, 2373, 0, 6414, 5505, 4886, 4886.

E. Exercise 3E

To find the Wardrop Equilibrium $f^{(0)}$, namely the equilibrium resulting from considering the choice that every single driver will take to minimize his own travel time. The choice will be sub-optimal as the driver does not know the state of the whole network. it is possible to solve the following optimization problem:

$$min \sum_{e \in E} \int_0^{f_e} \tau_e(t) dt$$

having

$$0 < f < C \quad\quad Bf = v$$

The new problem is obtained having the constraint that the total delay on a possible path γ chosen by the driver is less or equal than $\tilde{\gamma}$, namely every other possible path in the network from origin to destination. The f^* obtained is:
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.
To calculate the cost of the Wardrop equilibrium is enough to plug the flow values obtained into the cost function defined in the exercise. The value obtained is: 26292.357750911066

F. Exercise 3E'

The Wardrop equilibrium with tolls problem is very similar to the previous one with the only exemption that the following cost function is implemented:

$$min \sum_{e \in E} \int_0^{f_e} \tau_e(t) dt + \omega_e f_e$$

having

$$\omega_e = f_e^* \tau_e'(f_e^*)$$

and

$$0 < f < C \quad\quad Bf = v$$

The following optimal flow is obtained:
6643, 6059, 3132, 3132, 10164, 4638, 3006, 2542, 3132, 583, 0, 2927, 0, 3132, 5526, 2854, 4886, 2215, 464, 2338, 3318, 5656, 2373, 0, 6414, 5505, 4886, 4886
Accordingly, the new cost function social cost is computed and the following value is obtained: 25943.622350312624. It is possible to notice that the vector flow obtained by adding tolls and computing the wardrobe equilibrium is optimal as they are the values obtained by computing the social optimum. This makes sense as one possible purpose of adding tolls to a network is to optimize the flow pushing the driver to choose a faster route once considering the whole state of the network.

G. Exercise 3F

At this point, the cost function becomes the total additional delay compared to the total delay in free flow:

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

therefore the problem becomes:

$$min \sum_{e \in E} f_e(\tau_e(f_e) - l_e)$$

Edge	**SO-TAP**	**Wardrop**	**Wardrop, tolls**	**SO-TAP, Δ**	**SO-TAP, tolls, Δ**
1	*6642*	*6716*	*6643*	*6653*	*6653*
2	*6059*	*6716*	*6059*	*5775*	*5775*
3	*3132*	*2367*	*3132*	*3420*	*3419*
4	*3132*	*2367*	*3132*	*3420*	*3419*
5	*10164*	*10090*	*10164*	*10153*	*10153*
6	*4638*	*4645*	*4638*	*4643*	*4642*
7	*3006*	*2804*	*3006*	*3106*	*3105*
8	*2543*	*2284*	*2543*	*2662*	*2662*
9	*3132*	*3418*	*3132*	*3009*	*3009*
10	*583*	*0*	*583*	*879*	*878*
11	*0*	*177*	*0*	*0*	*0*
12	*2927*	*4171*	*2927*	*2355*	*2356*
13	*0*	*0*	*0*	*0*	*0*
14	*3132*	*2367*	*3132*	*3420*	*3419*
15	*5525*	*5445*	*5525*	*5510*	*5510*
16	*2854*	*2353*	*2854*	*3044*	*3043*
17	*4886*	*4933*	*4886*	*4882*	*4882*
18	*2215*	*1842*	*2215*	*2416*	*2415*
19	*464*	*697*	*464*	*444*	*444*
20	*2338*	*3036*	*2338*	*2008*	*2009*
21	*3318*	*3050*	*3318*	*3487*	*3487*
22	*5656*	*6087*	*5656*	*5495*	*5496*
23	*2373*	*2587*	*2373*	*2204*	*2204*
24	*0*	*0*	*0*	*0*	*0*
25	*6414*	*6919*	*6414*	*6301*	*6301*
26	*5505*	*4954*	*5505*	*5623*	*5624*
27	*4886*	*4933*	*4886*	*4882*	*4882*
28	*4886*	*4933*	*4886*	*4882*	*4882*

TABLE I: Summary results exercise 3.d, 3.e, 3.e', 3.f

while maintaining the same constraint as before. The f^* vector obtained is:
6653, 5775, 3420, 3420, 10153, 4643, 3106, 2662, 3009, 879, 0, 2355, 0, 3420, 5510, 3044, 4882, 2415, 444, 2008, 3487, 5495, 2204, 0, 6301, 5624, 4882, 4882 Then, the social optimum total cost is: 15095.513524607872. Combining tolls and the new cost function the new Wardrop equilibrium problem is obtained:

$$min \sum_{e \in E} \int_0^{f_e} \tau_e(t)dt - l_e * f_e + \omega_e * f_e$$

The f^* vector obtained is:
6653, 5775, 3419, 3419, 10153, 4643, 3105, 2662, 3009, 878, 0, 2356, 0, 3419, 5510, 3043, 4882, 2415, 444, 2009, 3487, 5496, 2204, 0, 6301, 5623, 4882, 4882
These values are rounded to integer values. The social cost under Wardrop equilibrium with tolls is: 15095.51325601912. As expected the value obtained is the same as the social optimum with the new cost function: once again adding tolls has been useful to optimize the flow.
Indeed, it is possible to calculate the Price of anarchy (PoA) defined as a measure that indicates how the efficiency of a system degrades due to the selfish behavior of its agents:

$$PoA(\omega) = \frac{\sum_{e \in E} f_e^*(\tau_e(f_e^*) - l_e)}{\sum_{e \in E} f_e^{(\omega)}(\tau_e(f_e^{(\omega)}) - l_e)} \approx 1$$

As expected the ratio is one indicating that the efficiency of the socially optimal flow is the same as the Wardrop one with tolls.