Report Homework I

Network Dynamics and Learning

Luca Benfenati
Politecnico di Torino
Torino, Italy
s286582@studenti.polito.it

***Abstract*—The following paper addresses three different exercises related to graph theory, network connectivity and network flow optimization. The aim of this report is to propose solutions to specific problems, dealing with both the theoretical and the practical side of the subject, in order to present an approach which remains as general and broad as possible.**

INTRODUCTION

Before starting with the practical part, we want to provide the theory elements which will be used in the following exercises, so to be, once again, as complete and exhaustive as possible. For each section, a initial part regarding theory is proposed, for then proceed with the specific exercise. Graph theory, network connectivity and network flow optimization topics will be explored.

I. EXERCISE 1

In the first exercise we face a first task to get familiar with network connectivity and flows, and three other tasks to address different maximization flow problems.

For the first task, we need to outline **Menger's Theorem**, which is exploited to understand how many nodes and links must we remove from a graph to disconnect two nodes. Menger's Theorem states that, given a weighted multi-graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{W}$, and for $i \neq j \in G$, then:

- min #{nodes to remove for j not to be reachable from i} = $c_{node}(i, j)$
- min #{links to remove for j not to be reachable from i} = $c_{link}(i, j)$
- min #{nodes to remove for G not to be connected} = $c_{node}(G)$
- min #{links to remove for G not to be connected} = $c_{link}(G)$

where $c_node(i, j)$ is the number of node-independent $i - j$ paths, and $c_link(i, j)$ is the number of $i - j$ paths.

For the remaining three tasks, we need to firstly outline the **Max flow problem**. Consider a directed graph $G = (\mathcal{V}, \mathcal{E})$, its node-link incident matrix B, a link capacity vector $C \in \mathbb{R}^{\mathcal{E}}, C > 0$, a link flows vector $f \in \mathbb{R}^{\mathcal{E}}_{+}$, and throughput $\tau \geq 0$, which is the total flow through the network from node o to node d, associated with f. Now consider two distinct nodes o and d, which we call respectively source and destination, we can define the maximum flow problems as a linear program defined by:

$$\begin{aligned} \min \quad & \tau^*_{o,d} = \max \tau \\ \text{s.t.} \quad & \tau \geq 0 && \text{throughput nonnegativity,} \\ & 0 \leq f \leq C && \text{nonnegativity and capacity constr.,} \\ & Bf = \tau(\delta^{(o)} - \delta^{(d)}) && \text{mass conservation} \end{aligned}$$

A flow f that satisfies the constraints is called *feasible flow*.

Then, we need to define the **min-cut capacity**: consider a so-called $o - d$ cut, which is a partition of the nodes set $\mathcal{V}$ in two subsets, $\mathcal{U}$ and $\mathcal{V}\backslash\mathcal{U}$ with $o \in \mathcal{U}$ and $d \in \mathcal{V}\backslash\mathcal{U}$. We say that the capacity of an $o - d$ cut $\mathcal{U}$ is the aggregate capacity of the links from $\mathcal{U}$ to $\mathcal{V}\backslash\mathcal{U}$:

$$C_{\mathcal{U}} := \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{V}\backslash\mathcal{U}} C_{ij} \tag{1}$$

Consequently, we can specify the min-cut capacity, i.e. the minimum capacity among all $o - d$ cuts as:

$$C^*_{o,d} = \min_{\substack{\mathcal{U} \subseteq \mathcal{V} \\ o \in \mathcal{U}, d \notin \mathcal{U}}} C_{\mathcal{U}} \tag{2}$$

Lastly, the **Max-flow min-cut theorem** states that the maximum throughput $\tau^*_{o,d}$ from o to d (solution of the linear program) coincides with the minimum cut capacity $C^*_{o,d}$ among all $o - d$ cuts. So, given a capacitated multigraph $G = (\mathcal{V}, \mathcal{E}, c)$, for $o \neq d \in \mathcal{V}$:

$$\tau^*_{o,d} = C^*_{o,d} \tag{3}$$

Finally, we can address the most practical part of the exercise, thus taking into account a unitary $o - d$ network flows on the graph $G = (\mathcal{V}, E)$ in Figure 1, where:

- $\mathcal{V} = \{o, a, b, d\}$ is the set of nodes, with node o identified as source and node d identified as destination/sink;
- $\mathcal{E} = \{e_1, e_2, e_3, e_4\}$ is the set of edges and each edge e_i has integer capacity C_e.

Point a)

We want to find the infimum of the total capacity that needs to be removed for no feasible unitary flows from o to d to exist. In order to have verify this condition, we are asked to search for those edges or those combinations of edges that, if removed, prevent the flow to go through the network. Once

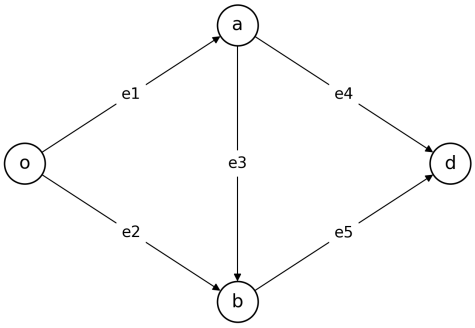

Fig. 1.

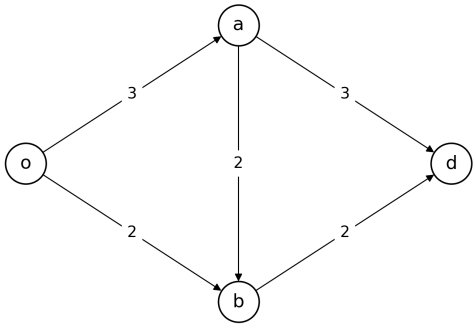

Fig. 2.

we have found them, we need to consider the sum of their respective capacities and to select the minimum out of it.

First of all, we determine all the possible paths between the source and the destination. In particular we can say that:

$$\Gamma_{o,d} = \{\gamma_1 = (o, a, d), \gamma_2 = (o, a, b, d), \gamma_3 = (o, b, d)\}$$

Now, exploiting Menger's theorem, we can easily say that the minimum number of link to be removed in order to have d not reachable from o is 2, since $c_link(o, d) = 2$, i.e. we have just 2 link-independent $o - d$ paths, in particular γ_1 and γ_2. Identified three distinct $o - d$ paths, we can find the edges we are interested in by just detaching all paths $\gamma_1, \gamma_2, \gamma_3$ at once. In particular we find the following combinations of edges to be disconnected:

- e_1, e_2
- e_4, e_5
- e_1, e_5
- e_2, e_3, e_4

Notice that the last combination of edges e_2, e_3, e_4 is considered for two reasons: firstly, we know for hypothesis that capacities are integer value (and not unitary), and then, none of the other three possible pairs of edges is already included in one of previous combinations. Hence, we consider the respective capacities, in order to find the infimum of the total capacity. It is immediate to state that the infimum of the total capacity is:

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

Point b)

As already anticipated in the Introduction of Exercise I, we address the maximization of the feasible throughput. In this particular point, link capacities are assumed to be:

- $C_1 = C_4 = 3$
- $C_2 = C_3 = C_5 = 2$

The new graph is displayed in Figure 2 In order to have a broader view on the current state of play, we compute the cut capacities:

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

and, thus, the min-cut capacity, $C^*_{o,d} = \min\{C_U\} = \min\{5, 7, 5, 5\} = 5$. As the Max-flow min-cut theorem states (Equation 3), we then know that the maximum throughput is $\tau^*_{o,d} = 5$, i.e. it coincides with the minimum cut capacity.

We are then provided with 1 unit of additional capacity to be allocated on one edge in order to maximize the feasible throughput $\tau_{o,d}$. After having computed all the cut capacities, it is immediate to notice that is not possible to increase the min-cut capacity with just 1 unit of additional capacity. Let's take for example e_1 and add 1 unit of capacity on it. Now if we compute the new cut capacities we obtain:

- $U = \{o\}, U^C = \{a, b, d\} \rightarrow C_U = 6$
- $U = \{o, b\}, U^C = \{a, d\} \rightarrow C_U = 6$
- $U = \{o, a\}, U^C = \{b, d\} \rightarrow C_U = 7$
- $U = \{o, a, b\}, U^C = \{d\} \rightarrow C_U = 5$

Even if we are able to increment by 1 two different cut capacities, the the min-cut capacity $C^*_{o,d} = min\{C_U\} = \min\{6, 7, 6, 5\} = 5$ is not increased. This reasoning can be carried out for any edge of the graph.

Point c)

For the third task, we are asked to carry out the same problem already seen in the previous point. However, now we are provided with 2 units of additional capacity rather than only 1. It is immediate to notice that it is useless to add the whole 2 units on just one edge, because it would turn out to be the same situation already seen in point b), where we may be able to increase the capacity of certain cuts, but, overall the min-cut capacity will remain the same. Instead, if we assign 1 unit of capacities to two different edges we can find 3 different combinations of edges able to increase the min-cut capacity and, consequently, the maximum throughput. In particular, we compute the cut capacities of all the combinations, which are given by:

1) adding 1 unit of capacity on edges e_1 and e_4:
   - $U = \{o\}, U^C = \{a, b, d\} \rightarrow C_U = 6$
   - $U = \{o, b\}, U^C = \{a, d\} \rightarrow C_U = 6$
   - $U = \{o, a\}, U^C = \{b, d\} \rightarrow C_U = 8$
   - $U = \{o, a, b\}, U^C = \{d\} \rightarrow C_U = 6$

2) adding 1 unit of capacity on edges e_2 and e_5:
   - $U = \{o\}, U^C = \{a, b, d\} \rightarrow C_U = 6$
   - $U = \{o, b\}, U^C = \{a, d\} \rightarrow C_U = 6$
   - $U = \{o, a\}, U^C = \{b, d\} \rightarrow C_U = 8$
   - $U = \{o, a, b\}, U^C = \{d\} \rightarrow C_U = 6$
3) adding 1 unit of capacity on edges e_1 and e_5:
   - $U = \{o\}, U^C = \{a, b, d\} \rightarrow C_U = 6$
   - $U = \{o, b\}, U^C = \{a, d\} \rightarrow C_U = 6$
   - $U = \{o, a\}, U^C = \{b, d\} \rightarrow C_U = 7$
   - $U = \{o, a, b\}, U^C = \{d\} \rightarrow C_U = 6$

As we can easily see, for these three combination the min-cut capacity is:

$$C^*_{o,d} = \min_{\substack{\mathcal{U} \subseteq \mathcal{V} \\ o \in \mathcal{U}, d \notin \mathcal{U}}} C_{\mathcal{U}} = \tau^*_{(o,d)} = 6$$

which once again, for the Max-flow min-cut theorem, corresponds to the optimal throughput.

Point d)

For the last task of Exercise 1, we are provided with 4 additional unit of capacity. Once again in order to maximize the throughput, we need to select some edges whose capacity will be increased. We are able to find 6 possible combinations of allocation of the 4 units on edges, in particular:

1) adding 1 unit of capacity on edges e_1, e_2, e_3 and e_4
2) adding 2 units of capacity on edges e_1 and e_4
3) adding 2 units of capacity on edges e_2 and e_5
4) adding 2 units of capacity on edges e_1 and e_5
5) adding 1, 1, 2 units of capacity respectively on edges e_1, e_2 and e_5
6) adding (2, 1, 1) units of capacity respectively on edges e_1, e_4, e_5, we obtain cut capacities given by:
   - $U = \{o\}, U^C = \{a, b, d\} \rightarrow C_U = 7$
   - $U = \{o, b\}, U^C = \{a, d\} \rightarrow C_U = 8$
   - $U = \{o, a\}, U^C = \{b, d\} \rightarrow C_U = 8$
   - $U = \{o, a, b\}, U^C = \{d\} \rightarrow C_U = 7$

Note that, for brevity's sake, not all the cut capacities for all combinations have been reported here, but just the last one, thus not compromising the report with verbosity. However, complete results are available and well-displayed in the Jupyter Notebook [2] on which this report is based. The min-cut capacity can be computed as:

$$C^*_{o,d} = \min_{\substack{\mathcal{U} \subseteq \mathcal{V} \\ o \in \mathcal{U}, d \notin \mathcal{U}}} C_{\mathcal{U}} = \tau^*_{(o,d)} = 7$$

Among the optimal allocations, we are then asked to select the one that maximizes the sum of the cut capacities. As we can see from the only cut capacities reported here, and from the others in the paper, all of combinations maximize the sums of the cut capacities.

As a last note, it is important to notice that, with 4 units, there are some alternatives of allocations for which the min-cut capacity is increased, but it is not maximized. These combinations have not been considered.

II. Exercise 2

In the second exercise, we confront bipartite graphs and, in particular, multi origin-destination maxflow problems: the perfect matching problem will be solved exploiting the analogy between maximal flows problem, already seen in Exercise 1. Since we are committed to our "theoretical than practical" approach, we need firstly to introduce those theory concepts that we will exploit in the different tasks.

First of all, we can define **bipartite graph** as:

$$G = (\mathcal{V}_0 \cup \mathcal{V}_1, \mathcal{E}), \mathcal{E} \subseteq (\mathcal{V}_0 X \mathcal{V}_1) \cup (\mathcal{V}_1 X \mathcal{V}_0) \tag{4}$$

i.e. those graphs whose nodes can be divided into two disjoint and independent sets $\mathcal{V}_0$ and $\mathcal{V}_1$ such that every edge connects a node in $\mathcal{V}_0$ to one in $\mathcal{V}_1$.

Then, we define a **multi origin-destination maxflow** problems as maximum flow problems where we have multiple origin-destination. One of the possible application of this type of problem is the **perfect matching**. Given a graph $G = (\mathcal{V}, \mathcal{E})$, a matching is a subset of edges $M \subseteq \mathcal{E}$ such that no two edges in $\mathcal{M}$ share a common node (head or tail). Now we can say that, considered a simple bipartite graph, whereby the node set can be partitioned as $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1$, with $\mathcal{V}_0 \cap \mathcal{V}_1 = \emptyset$, and no edges between $\mathcal{V}_0$ and $\mathcal{V}_1$ exist, for $h = 0, 1$, we shall refer to a matching $\mathcal{M}$ in G as V_h-perfect (or V_h-saturating) if every node in V_h is matched in $\mathcal{M}$. If we are dealing with a bipartite graph $G = (\mathcal{V}_0 X \mathcal{V}_1, \mathcal{E})$, a V_0-perfect matching, is a matching that matches every node in V_0.

Finally, we can also introduce **Hall's theorem**, which will be prove useful to establish whether a perfect matching exists or not. In particular, 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

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

where

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

is the neighborhood of $\mathcal{U}$ in G. In other words, for every subset U of V_0, we require that U possesses a number of neighboring nodes which is at least the number of nodes in U.

For this exercise, we consider 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:

$$p_1 \rightarrow (b_1, b_2), p_2 \rightarrow (b_2, b_3),$$
$$p_3 \rightarrow (b_1, b_4), p_4 \rightarrow (b_1, b_2, b_4)$$

Point a)

The first task asks us to represent the interest pattern using a simple bipartite graph. Our graph is shown in Figure 3.

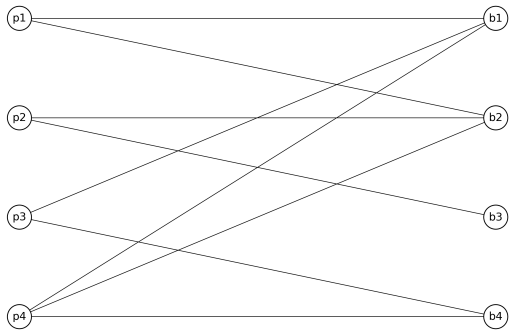

Fig. 3. Bipartite graph

Point b)

Now, we are asked to exploit max-flow problems to establish whether there exists a perfect matching that assigns to every person a book of interest. First of all we need to establish if a perfect matching exists or not. For this purpose, we exploit Hall's theorem. If we consider $V_0 = (p_1, p_2, p_3, p_4)$ and $V_1 = (b_1, b_2, b_3, b_4)$, and we take $\mathcal{U} = V_0$, we can see that $|V_0| = |V_1|$, so a perfect matching exists.

Given a simple bipartite graph $G = (V, E)$, in order to exploit analogy between the $o - d$ max-flow problem and the perfect matching, we need to consider a directed capacitated graph $G1$, with node set $V \cup s \cup t$, and edge set constructed as follows:

- for every node $p_i \in (p_1, p_2, p_3, p_4) = V_0$, add an edge (s, p_i), with capacity 1;
- for every node $b_i \in (b_1, b_2, b_3, b_4) = V_1$, add an edge (b_i, t), with capacity 1;
- for every undirected edge (i, j) in BG, add a directed edge (i, j) in $G1$ with capacity 1.

This new auxiliary graph $G1$ is shown in Figure 4.

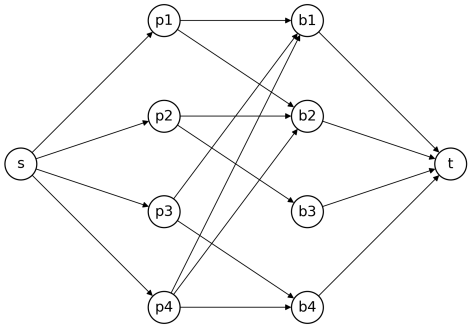

Fig. 4. Auxiliary graph

Now we need to exploit the analogy between perfect matching and maxflow problem on this auxiliary network $G1$. In particular, in this form we can say that a V_0-perfect matching on G exists if and only if it there exists a flow with throughput $|V_0|$ on the network $G1$. Solving the maxflow problem on this auxiliary network we are able to find our perfect matching. Our solution is shown in Figure **??**.

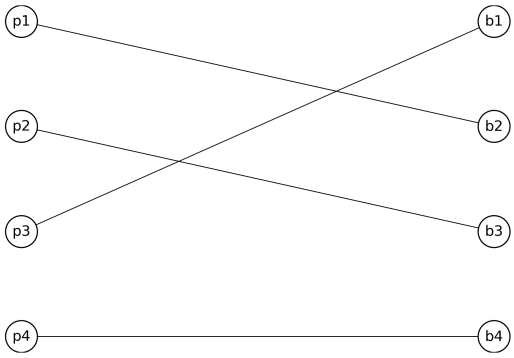

Fig. 5. Perfect Matching

Point c)

Now, we assume that there are multiple copies of book, specifically the distribution of the number of copies is $(2, 3, 2, 2)$. Then, we need to use, once again, the analogy with max-flow problems to establish how many books of interest can be assigned in total, considering this time different kind of constraints. We consider the graph already built in point (b), in particular $G1$, but we need to modify its link capacities in order to satisfy the new constraints. We codify these ones as follows:

1) firstly, we know that each person has no constraint on the number of books he can buy:
   - this is codified setting the capacities of the edges between source *s* and people $p_i \in (p_1, p_2, p_3, p_4)$ to infinite (at least for now);
2) secondly, we know that each person can not take more copies of the same book has two effects:
   - since there are only 4 books available, it is sufficient to set the capacity to each edge $s - p_i$ with $p_i \in V_0$ to 4 (not infinite anymore);
   - it is necessary to set the capacities on the edges between people $p_i \in V_0$ and books $b_i \in V_1$ to 1;
3) lastly, to codify the number of copies $(2, 3, 2, 2)$, we need to set the capacities of the edges between the books $b_i \in V_1$ and the destination t to, respectively, $(2, 3, 2, 2)$.

The new graph that is used to solve this problem is shown in Figure 6. We can now find the maximum throughput in order to establish how many books of interest can be assigned in total considering these new capacities on edges. Results shows that a total of 8 book of interest can be be assigned, distributed as follows:

- book b_1 has been assigned to p_3 and p_4
- book b_2 has been assigned to p_1, p_2 and p_4
- book b_3 has been assigned to p_2
- book b_4 has been assigned to p_3 and p_4

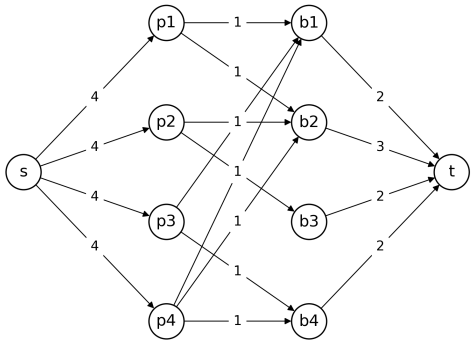

Fig. 6. Capacitated Auxiliary graph

Point d)

It is immediate to notice that the maximum throughput found in the point (c) is not maximal: this can be simply seen looking at the residual capacity on edge $b_3 - t$. Even if there were 2 copies of book b_3 avaiable, only 1 copy has been assigned. In this case, on edge $b_3 - t$ we still have one unit of capacity to be saturated.

As a consequence, it is reasonable to think selling only one copy of the book b_3, since we have two copies of these books available, but only one is sold. Instead, it is better to buy a copy of the book b_1, since we have 3 people interested in it, but only 2 copies available. From a graphical point of view, this can easily see if we compute the in-degree of node b_1, i.e. the number of head ends adjacent to a node. In particular, the in-degree of node i is $w_i = \sum_{j \in \mathcal{V}} W_{ij}$, where W is the weight matrix. So for node b_1 we have $w_{b_1} = 3$. Therefore, the new assignment for capacities on the edges between books b_i and the destination t are $(3, 3, 1, 2)$. Note that these logical statements have been verified with the appropriate maximum algorithm.

III. Exercise 3

For this last exercise we are working with different kind of optimization problems. Starting from the shortest path problem, that will be solved both with convex optimization and with more efficient techniques, then we move on to deal with maximum flow and external inflow, thus concluding with System and User Optimum Traffic Assignment problem, working with Wardrop equilibrium, tolls and cost functions.

First of all, we need to introduce **convex separable network flow optimization** and the theory behind this type of problem. Given a multigraph (V, E), an exogenous network flow is a vector $\nu \in \mathrm{R}^V$ such that:

$$\sum_{i \in V} \nu_i = 0. \tag{5}$$

A network flow is a vector $f \in \mathrm{R}^E$ satisfying a positivity constraint and a mass conservation constraints, i.e.,

$$f \geq \mathbf{0}, \quad Bf = \nu. \tag{6}$$

Every edge is endowed with a separable non-decreasing convex cost function $\psi_e(f_e)$ such that $\psi_e(0) = 0$. Given an exogenous flow ν and a network with node-edge matrix B, we study the following optimization problem:

$$f^* \in \underset{\substack{f \in \mathrm{R}_+^E \\ Bf = \nu}}{\arg\min} \quad \sum_{e \in E} \psi_e(f_e). \tag{7}$$

The ratio $\psi_e(f_e)/f_e$ may be interpreted as the cost per unit of flow sent along the edge e. The convexity of $\psi_e(f_e)$ is thus equivalent to requiring that the marginal cost for sending some flow on each edge is non-decreasing in the flow itself.

Then we define the **Wardrop equilibrium** is a path flow distribution such that if a path is used, then the cost of the path is minimal, i.e.,

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

where $\mathcal{P}$ is the path set, $f = Az$ is the projection of the path flow on the link set, and $c_p(z) = \sum_{e : e \in p} d_e(f_e)$, i.e., the cost of a path is the sum of the delays that compose the path.

We say that f is a Wardrop equilibrium if it is induced by a Wardrop equilibrium z via $f = Az$. We can also say that the flow vector $f^{(0)}$ corresponding to a Wardrop equilibrium can be obtained as a solution of a network flow optimization given that the cost functions $\psi_e(f_e)$ are chosen as

$$\psi_e(f_e) = \int_0^{f_e} d_e(s)\ \mathrm{d}s.$$

It is important to notice that to solve this type of optimization problems we exploit CVXPY [4], which is a Python-embedded modeling language for convex optimization problems, that allows to express problems in a natural way that follows the math. We use to solve separable convex network flow optimization arises in several applications and, in particular the shortest paths between origin and destination and system optimum traffic flows problem. The rest of the theory elements describing the three different kind of problems (i.e. Shortest path, System optimal traffic assignment and User optimal traffic assignment) will be displayed and analyzed in the related task.

For the exercise, we are given an approximate highway network in Los Angeles, which covers part of the real highway network: each node represents an intersection between highways (and some of the area around), and each link represent an highway. Along with this network, we are given different files, in particular:

- the node-incidence matrix B,
- the capacities C_{e_i} of each link $e_i \in e_1, ..., e_2 8$,
- the minimum travelling times l_e for each edge, values that are simply retrieved by dividing the length of the highway segment with the assumed speed limit 60 miles/hour,
- and the flow vector.
- a delay function, given by:

$$d_e(f_e) = \frac{l_e}{1 - \frac{f_e}{C - e}}$$

Our network is shown in Figure 7

Point a)

The first task asks us to find the shortest path between node 1 and node 17. In this case,we need to consider the fastest path, i.e. the path with the shortest travelling time in an empty network (so without considering the capacities). As already anticipated, we will exploit convex separable network flow optimization to solve this problem. Thus it is crucial to firstly define the optimization problem, especially defining which is our function to be minimized, i.e. our $\psi_e(f_e)$. When this class of problems are applied for shortest paths algorithms in a two-terminal network with origin and destination (o, d), our linear cost function

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

where each edge is endowed with a length l_e which differs from edge to edge and, in our case, is represented by that travelling time. Even in this formulation, the cost for sending a unit of flow on the edge e does not depend on f_e, which means that congestion effects are not taken into account. Congestion effects appear when considering strictly convex cost functions. We now use network flow optimization to find the shortest path from 1 to 17. Since the problem is linear, the solution will be to allocate the flow on a convex combination of the optimal paths. The optimal paths can be thus deduced from the non-zero components of the optimal flow.

As optimal result we obtain:
$1.00000000e{+}00, 1.00000000e{+}00, 1.19595586e{-}115.07129566e{-}12,$
$1.49988426e{-}10, 1.33658919e{-}10, 8.54100951e{-}113.78266693e{-}11,$
$9.99999998e{-}01, 8.90389208e{-}12, 3.91753215e{-}111.00000000e{+}00,$
$6.88981434e{-}12, 5.07752202e{-}12, 1.62878397e{-}115.42208640e{-}11,$
$1.91831469e{-}11, 5.71274798e{-}11, 8.67469798e{-}111.99611946e{-}09,$
$1.29358670e{-}10, 2.12547935e{-}09, 4.14939343e{-}111.16133834e{-}11,$
$1.00000000e{+}00, 4.65871714e{-}11, 3.08330688e{-}113.08494486e{-}11$
The following characterization can be used to individuate the shortest path: if the flow on a path p is positive (i.e., if the flow on all the edges that compose the path is positive), then p is a shortest path. In this case, since all the values are positive, we search for the biggest ones, in particular, the ones which tend to 1. From this last observation, we can easily see that the shortest path is identified by edges $(e_1, e_2, e_{12}, e_9, e_{25})$. The shortest path is shown in the Figure 8.

In the Jupyter Notebook [2], it is possible to see that the shortest path has been also computed with the NETWORKX [5] built-in function, which should be more efficient than the optimization problem.

Point b)

In this point, we need to compute the maximum flow between node 1 and 17. We have already explained how we can solve it, therefor we just report the results. The maximum network flow is given by 22448, and the flow is distributed among edges and nodes as follows:

- 1: 2: 8741, 6: 13707,
- 2: 3: 8741, 7: 0,
- 3: 4: 0, 8: 0, 9: 8741,
- 4: 5: 0, 9: 0,
- 5: 14: 0,
- 6: 7: 4624, 10: 9083,
- 7: 8: 4624, 10: 0,
- 8: 9: 4624, 11: 0,
- 9: 13: 6297, 12: 7068,
- 13: 14: 3835, 17: 10355,
- 14: 17: 3835,
- 10: 11: 825, 15: 8258,
- 11: 12: 825, 15: 0,
- 15: 16: 8258,
- 12: 13: 7893,
- 17: ,
- 16: 17: 8258

Point c)

Given the flow vector, we are required to find the external inflow ν which satisfies $Bf = \nu$. We know that, given a graph $G = (\mathcal{V}, \mathcal{E}, s, t)$ an *exogenous net flow* in G is a vector $\nu \in \mathbb{R}^{\mathcal{V}}$ satisfying the constraint

$$\sum_i \nu_i = 0$$

. The positive part $[\nu_i]_+ = max\{0, \nu_i\}$ and the negative part $[\nu_i]_- = max\{0, -\nu_i\}$ of the net flow in a node i are to be interpreted as the exogenous inflow in and, respectively, the external outflows from i. This is the equivalent to requiring that the total exogenous inflow matches the total external outflow.

Exploiting the flow balance equation $Bf = \nu$ (i.e. mass conservation constraint), we can retrieve the external inflow as:

ν = [16806, 8570, 19448, 4957, -746, 4768, 413, -2, -5671, 1169, -5, -7131, -380, -7412, -7810, -3430, -23544].

For the followings points we assume that the exogenous inflow is zero in all the nodes except for node 1, for which ν_1 has the same value just computed, and node 17, for which $\nu_{17} = -\nu_1$.

Point d)

In this section we face the so-called **System-Optimum Traffic Assignment Problem (SO-TAP)**, which concerns optimal traffic assignments in transportation networks. In this context, links $e \in \mathcal{E}$ represent (portions of) roads and nodes represent junctions, while the cost function is defined in terms of a delay function $d_e(f_e)$ that returns the delay encountered by any user traversing that link as a function of the flow on that link.

In this case, we need to find the social optimum f^* with respect to the delays, on the different links $d_e(f_e)$, where the delay function is defined as:

$$d_e(f_e) = \frac{l_e}{1 - \frac{f_e}{C_e}}, \qquad 0 \leq f_e \leq C_e$$

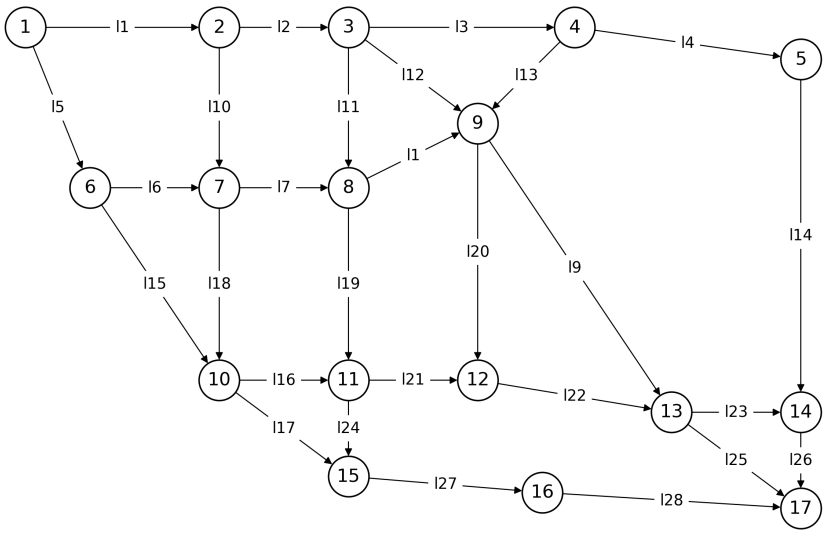

Fig. 7. Approximate highway network of Los Angeles

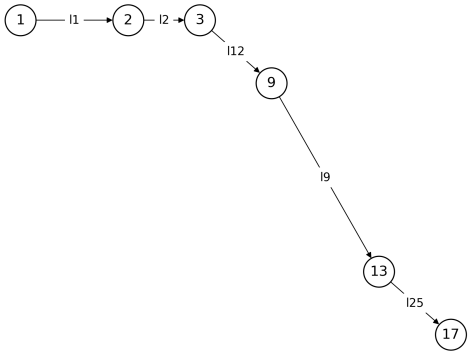

Fig. 8. Shortest path

This social optimum f^* is found minimizing the cost function defined as:

$$\sum_{e\in\mathcal{E}} f_e d_e(f_e) = \sum_{e\in\mathcal{E}} \frac{f_e l_e}{1-\frac{f_e}{C_e}} = \sum_{e\in\mathcal{E}} \left(\frac{l_e C_e}{1-\frac{f_e}{C_e}} - l_e C_e \right) \quad (8)$$

To solve this problem, we need to address once again a convex separable optimization problem. This means that the function to be minimized will be our cost function. Developing the cost function we are able to obtain:

$$\sum_{e\in\mathcal{E}} \left[l_e C_e \left(\frac{1}{1-\frac{f_e}{C_e}} \right) - l_e C_e \right] \quad (9)$$

Minimizing this function with the same procedure as before, we can obtain as result the social optimal cost: $6.64219910e+03, 6.05893789e+03, 3.13232779e+03, 3.1323258e+03, 1.01638009e+04, 4.63831664e+03, 3.00634073e+03, 2.5426346e+03, 3.13154448e+03, 5.83261212e+02, 1.45164550e-02, 2.9265955e+03, 1.89781986e-03, 3.13232589e+03, 5.52548426e+03, 2.8542726e+03, 4.88644874e+03, 2.21523712e+03, 4.63720641e+02, 2.3376876e+03, 3.31799129e+03, 5.65567890e+03, 2.37310712e+03, 1.9956728e-03, 6.41411626e+03, 5.50543301e+03, 4.88645073e+03, 4.8864507e+03$ and the social optimal cost, which is 25943.62261121287.

Point e)

In this following point, we are asked to find the Wardrop equilibrium for $f^{(0)}$, which is equivalent to ask for the solution of the **User-Optimum Traffic Assignment Problem (UO-TAP)**. This time, we need to minimize the cost function given by:

$$\sum_{e\in\mathcal{E}} \int_0^{f_e} d_e(s)\, ds \quad (10)$$

Now we have that our convex function is given by $\psi_e(f_e) = \int_0^{f_e} d_e(s)\, ds$.

The idea is to model flows resulting not from a centralized optimization, but rather as the outcome of selfish behaviors of drivers. Such selfish behavior is modeled by assuming that drivers choose their route so as to minimize the delay they experience along it. This is formalized by the Wardrop equilibrium, which should be interpreted as the result of a rational, and selfish, behavior of drivers: none of them would choose a sub-optimal route (i.e., an o-d path with larger delay) if a better route is available.

Developing our cost function, we found that the convex function to be minimized is reduced to:

$$\psi_e(f_e) = -\sum_{e\in\mathcal{E}} l_e c_e \left(\ln(c_e - f_e) - \ln(c_e)\right)$$

Solving the optimization we found our Wardrop equilibrium, which is:

$6.71564933e{+}03, 6.71564464e{+}03, 2.3674091e{+}03, 2.3674087e{+}03$
$1.00903506e{+}04, 4.64539440e{+}03, 2.80384556e{+}03, 2.28355729e{+}03$
$3.41848027e{+}03, 4.68630001e{-}03, 1.76815987e{+}02, 4.17141953e{+}03$
$3.88215600e{-}04, 2.36740874e{+}03, 5.44495615e{+}03, 2.35317274e{+}03$
$4.93333694e{+}03, 1.84155353e{+}03, 6.97104258e{+}02, 3.03649694e{+}03$
$3.05027647e{+}03, 6.08677341e{+}03, 2.58651317e{+}03, 5.27016717e{-}04$
$6.91874051e{+}03, 4.95392191e{+}03, 4.93333747e{+}03, 4.93333747e{+}03$

Since we found our wardrop equilibrium, we need to find the cost of the wardrop equilibrium. The function of the delay was already given as $d_e(f_e) = \frac{l_e}{1-f_e/C_e}$ with $0 \leq f_e \leq C_e$, so it is just necessary to find the total cost adding the delay contribution of each edge at Wardrop. Once we found the total delay at user optimum, we can compute the **Price of Anarchy (PoA)** as the ratio between the total delay at the Wardrop equilibrium and the minimum possible total delay (i.e. the total delay at social optimum).

The rationale behind PoA is that selfishly and rationally choosing a route that minimizes their own delay, drivers do end up minimizing some global objective: the price of anarchy is then implied by the fact that the global objective implicitly optimized by the drivers does not coincide in general does not coincide with the social objective.

Remember that the PoA represent the ratio between the Wardrop cost (found with UO-TAP) and the optimal cost (found with SO-TAP), so it should be included in $[1, 4/3]$. As a result we obtained:

- Wardrop cost: 26292.963306012705
- Price of anarchy 1.0134653783719798

Point f)

Point f) Then, we are asked to introduce tolls such that the toll on link e is $w_e = f_e^* d_e'(f_e^*)$, where f_e^* is the flow at the system optimum (point *(d)*). Now the delay on link e is given by $d_e(f_e)+\omega_e$, so we need to compute the new Wardrop equilibrium at ω, $f^{(\omega)}$.

Toll design in one way to sort of influence the rational and selfish behavior of drivers that minimize their own delay. Tolls are, indeed, used to align the user optimal flow to the system optimal one.

Back at our exercise, we solve the derivative in order to find the toll ω_e and we are able to obtain that :

$$\omega_e = f_e \left(\frac{c_e l_e}{(c_e - f_e)^2} \right) \tag{11}$$

As toll vector we obtain:

$1.92210631e{+}00, 1.85064818e{-}01, 5.1669498e{-}021.0517637e{-}01$
$1.44072968e{+}00, 4.69146428e{-}01, 1.07571171e{-}01, 5.69255527e{-}02$
$2.78803191e{-}01, 6.15501126e{-}03, 1.74005540e{-}07, 7.53489887e{-}02$
$2.18581142e{-}08, 1.26658219e{-}01, 4.81817175e{-}01, 8.20545054e{-}02$
$6.85716144e{-}02, 1.73792008e{-}02, 1.50892092e{-}03, 1.38933840e{-}02$
$6.58964442e{-}02, 2.63303327e{-}01, 6.70070611e{-}02, 1.14962944e{-}08$
$4.09840401e{-}01, 2.87273604e{-}01, 1.91578098e{-}01, 5.27755447e{-}01$

With these new tolls we compute the new wardrop equilibrium: thus, we need to add the toll contribution in the convex function to minimize in the optimization problem. New Wardrop equilibrium with tolls is given by:

$[6.64297447e + 03, 6.05907644e + 03, 3.13247229e + 03, 3.13247200e + 03$
$1.01630254e{+}04, 4.63825875e{+}03, 3.00632655e{+}03, 2.54233808e{+}03$
$3.13148797e{+}03, 5.83898031e{+}02, 1.14255443e{-}03, 2.92660301e{+}03$
$2.86108919e{-}04, 3.13247200e{+}03, 5.52476667e{+}03, 2.85422630e{+}03$
$4.88637059e{+}03, 2.21583023e{+}03, 4.63989614e{+}02, 2.33745341e{+}03$
$3.31821555e{+}03, 5.65566896e{+}03, 2.37303537e{+}03, 3.63323218e{-}04$
$6.41412156e + 03, 5.50550737e + 03, 4.88637096e + 03, 4.88637096e + 03]$ and so:

- New wardrop cost with tolls is: 25943.622350312628
- New price of anarchy is: 0.9999999899435693

It is immediate to notice that the new wardrop cost, which has been updated with the tolls, is equal to the initial social optimum cost: the Wardrop equilibrium flow $f^{(\omega)}$ coincides with the system optimum flow f^*. In this observation shows perfectly which is the role of link tolls, and why are them so important: the Wardrop equilibrium can indeed be seen as the optimal network flow provided that we consider the appropriate costs.

Point g)

Point g) Finally, instead of the total delay $d_e(f_e)$, we consider the cost to be the total additional delay compared to the total delay in free flow be given by:

$$c_e(f_e) = f_e \left(d_e(f_e) - l_e \right) \tag{12}$$

subject, as always, to the flow constraints. With this cost function, we need to firstly compute system optimum f^* for the cost above. Following the same procedure already explained in point (d), we found that the new optimal cost is equal to 15095.513228359 (note: new system optimum vector is not reported here because it is already a bit verbose, refer to the Jupyter notebook for extended results).

Then, we need to construct tolls $\omega_e^*, e \in \mathcal{E}$ such that the new Wardrop equilibrium with the constructed tolls $f^{(\omega^*)}$ coincides with f^* and compute the new Wardrop equilibrium. So:

1) first we compute the new Wardrop equilibrium considering this new delay function, in particular $d_e(f_e) - l_e$. The function to minimize is:
$$\psi_e(f_e) = -l_e C_e * (\log(c_e - f_e) - \log(c_e)) + l_e C_e$$
2) then, we construct the tolls with same function already used in point (e);
3) thus, we add the tolls to the new Wardrop equilibrium, in order to satisfy the requirements that the new Wardrop equilibrium with the constructed tolls $f^{(\omega)}$ coincides with the new system optimum f^*;
4) finally I need to compute the new Wardrop equilibrium with tolls, which take has been built considering the new cost function

CONCLUSION

We have addressed different problems and we have tried to propose an answer that could give an overview also on a more general subject. Topics as graph theory, network flows,

optimization problems have been explored and a panoramic on the subjects in the first part of the Network Dynamics and Learning course has been carried out.

REFERENCES

[1] G. Como and F. Fagnani, "Lecture on Network Dynamics"
[2] L. Benfenati, "Homework 1 - Network Dynamics an Learning, Jupyter Notebooks (.ipynb)"
[3] https://networkx.org/
[4]
[5] https://networkx.org/