POLYTECHNIC OF TURIN

Master's Degree
in Data Science and Engineering

Network Dynamics and Learning

Homework 1

Riccardo Bosio 291299

Anno Accademico 2020-2021

Chapter 1

Introduction

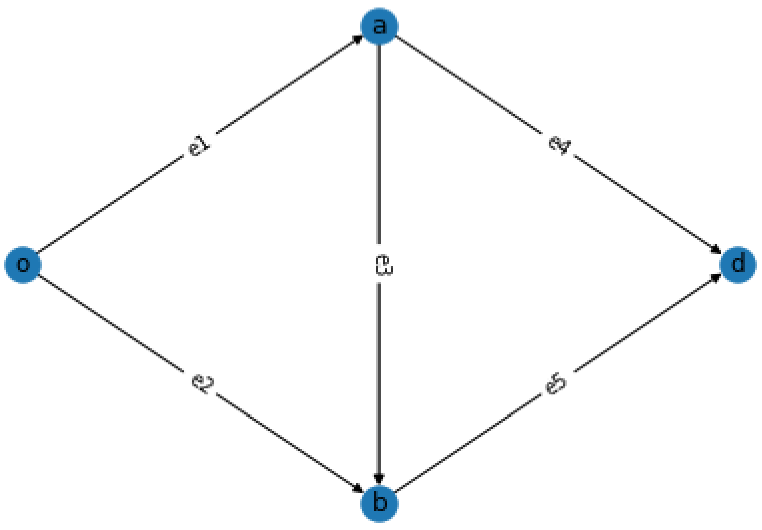
This document contains my solution to the Homework 1's exercises of the Network Dynamics and Learning course. The work is organized as follows. Each chapter corresponds to one exercise: inside the chapter, the first section contains the theory used, while in the others there is the solution of each point of the exercise. At this link https://github.com/riccardobosio/NDL, it is possible to see the text of exercises and the code used.

Chapter 2

Exercise 1

In this exercise I will consider the graph in figure 2.1.

Figure 2.1. Exercise 1 graph.



2.1 Theory used

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, s, t)$ we define an exogenous net flow on $\mathcal{G}$ as a vector $\nu \in \mathcal{R}^{\mathcal{V}}$ satisfying the constraint

$$\sum_i \nu_i = 0.$$

The positive part $[\nu_i]_+ = max\{0, \nu_i\}$ of the net flow in a node i is interpreted as the exogenous inflow in i, while the negative part $[\nu_i]_- = max\{0, -\nu_i\}$ as the external outflow from i. The constraint $\sum_i \nu_i = 0$ is then equivalent to having that the total external outflow matches the total exogenous inflow. The throughput is the total flow that goes through the network:

$$\mathcal{X} = \frac{1}{2}\sum_i |\nu_i| = \sum_i [\nu_i]_+ = \sum_i [\nu_i]_-.$$

Given a graph $\mathcal{G}$ and an exogenous net flows $\nu \in \mathcal{R}^{\mathcal{V}}$, a network flow is a nonnegative vector $f \in \mathcal{R}_+^{\mathcal{E}}$ whose entries f_e satisfy the flow balance equations

$$\nu_i + \sum_{e\in\mathcal{E}|\mathcal{K}(e)=i} f_e = \sum_{e\in\mathcal{E}|\theta(e)=i} f_e,$$

with $i \in \mathcal{V}$ and f_e represents the flow on the link $e \in \mathcal{E}$. These flow balance equations can be rewritten as

$$Bf = \nu.$$

$o - d$ flows are network flows with a single origin o and a single destination d, i.e. nonnegative vectors $f \in \mathcal{R}^{\mathcal{E}}$ such that:

$$Bf = \mathcal{X}(\delta^{(o)} - \delta^{(d)}),$$

for some throughput value $\mathcal{X}$. The capacity $c_e > 0$ of a link $e \in \mathcal{E}$ represents the maximum flow that is allowed through it. $c \in \mathcal{R}^{\mathcal{E}}$ is the vector of all link capacities. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, s, t)$, a capacity vector $c \in \mathcal{R}^{\mathcal{E}}, c > 0$, and two distinct nodes $o, d \in \mathcal{V}$, the maximum flow problem is

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

This optimization problem aims at finding the maximum throughput $\mathcal{X}$ from a given node o to another node d that can be achieved by a flow vector f without violating the link capacity constraints. An $o - d$ cut is a partition of the node set $\mathcal{V}$ in two subsets, $\mathcal{U}$ and $\mathcal{U}^c$, such that the origin o belongs to $\mathcal{U}$ and the destination node d belongs to the complementary set $\mathcal{U}^c$. The capacity of an $o - d$ cut $\mathcal{U}$ is the aggregate capacity of the links crossing it from the left to the right:

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

The Max-flow min-cut theorem tells us that the maximum throughput from o to d coincides with the minimum capacity among all $o - d$ cuts.

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

2.2 Point a

For Merger's Theorem, the minimum number of nodes (different from o and d) that have to be removed from $\mathcal{G}$ in order for d not to be reachable from o equals $c_{node}(o, d)$, which is the maximum number of node-independent $o - d$ paths.

In this case $c_{node}(o, d) = 2$, therefore 2 capacities need to be removed for no feasible unitary flows from o to d to exist.

2.3 Point b

Assuming that the link capacities are

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

and let V be the set of nodes $\{o, a, b, d\}$, I define 4 possible cuts and their corresponding capacities:

- $U_1 = \{o\}, V/U_1 = \{a,b,d\} \rightarrow C_{U_1} = e_1 + e_2 = 5;$
- $U_2 = \{o,a\}, V/U_2 = \{b,d\} \rightarrow C_{U_2} = e_2 + e_3 + e_4 = 7;$
- $U_3 = \{o,b\}, V/U_3 = \{a,d\} \rightarrow C_{U_3} = e_1 + e_5 = 5;$
- $U_4 = \{o,a,b\}, V/U_4 = \{d\} \rightarrow C_{U_4} = e_4 + e_5 = 5.$

Applying the max-flow min-cut theorem to this case, $\mathcal{X}^*_{o,d} = c^*_{o,d} = min_{\mathcal{U}\subseteq\mathcal{V}, o\in\mathcal{U}, d\notin\mathcal{U}} c_{\mathcal{U}} = 5$. If I want the maximum throughput from o to d to be bigger, I need to increase C_{U_1}, C_{U_3} and C_{U_4}. Since I have just one unit to add, I should find an edge shared by the three cuts U_1, U_3, U_4. This is impossible because such an edge does not exist. This means that wherever I allocate one additional capacity unit, the maximum throughput will remain 5.

2.4 Point c

If I had 2 units of capacity to allocate, I would face different choices:

- add one unit to e_1 and one to e_4;
- add one unit to e_1 and one to e_5;
- add one unit to e_2 and one to e_5.

In all these three cases I obtain a maximum throughput equal to 6 ($\mathcal{X}_{o,d} = 6$).

2.5 Point d

If I had 4 capacity units to allocate, I would have different options:

- add two units to e_1 and two to e_4;
- add two units to e_1 and two to e_5;
- add two units to e_1, one to e_4 and one to e_5;
- add one unit to e_1, one to e_2, one to e_4 and one to e_5;
- add one unit to e_1, one to e_2 and two to e_5;
- add two capacity units to e_2 and two to e_5.

In all the cases the maximum throughput is equal to 7 and the sum of cut capacities is equal to 30.

Chapter 3

Exercise 2

3.1 Theory used

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is simple if it is undirected, unweighted and the weight matrix $\mathcal{W}$ has zero diagonal, equivalently if $\mathcal{G}$ contains no self-loops. A simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is called bipartite if we can split the node set into two nonempty subsets so that there are no links between nodes in the same subset. If I find a partition $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1$ such that $\mathcal{W}_{ij} = 0$ for all $i, j \in \mathcal{V}_0$ and $\mathcal{W}_{ij} = 0$ for all $i, j \in \mathcal{V}_1$. A matching in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is a subset of links $\mathcal{F} \subseteq \mathcal{E}$ such that every node is either the head or the tail node of at most one link in $\mathcal{M}$. A matching in $\mathcal{G}$ is $\mathcal{V}_h - perfect$ if every node i in $\mathcal{V}_h$ is matched in $\mathcal{M}$. The Hall's Marriage Theorem states that for a simple bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{V}_0 \subseteq \mathcal{V}$, there exists $\mathcal{V}_0$-perfect matching in $\mathcal{G}$ if and only if

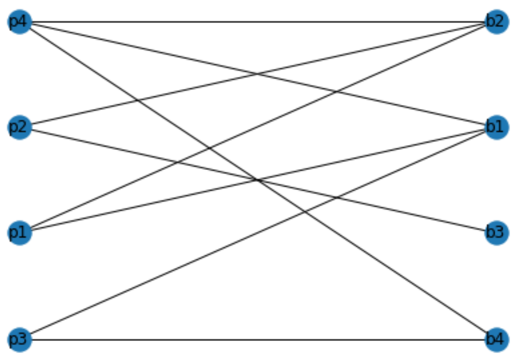
$$|\mathcal{U}| \leq |\mathcal{N}_{\mathcal{U}}|, \forall \mathcal{U} \subseteq \mathcal{V}_0,$$

where $\mathcal{N}_{\mathcal{U}} = \cup_{i \in \mathcal{U}} \mathcal{N}_i$ is the neighborhood of $\mathcal{U}$ in $\mathcal{G}$.

3.2 Point a

The interest pattern can be represented using the simple bipartite graph in figure 3.1.

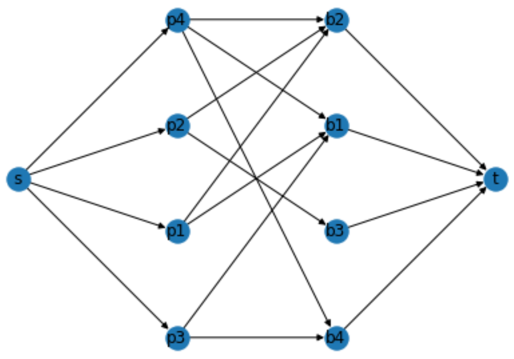
Figure 3.1. Exercise 2 bipartite graph.



3.3 Point b

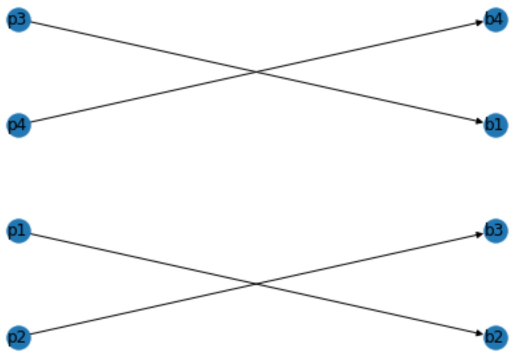
To reconduct the problem to a single source and sink, I add an additional pair of nodes s and t and I create edges from s to each person p_i and from each book b_i to t. I set the capacities of all

Figure 3.2. Modified graph.



edges to 1, because each person can take only one type of book and only one copy of each book is available. Then I compute the maximum flow using the networkx *maximum_flow* algorithm. The resulting perfect matching is shown in figure 3.3.

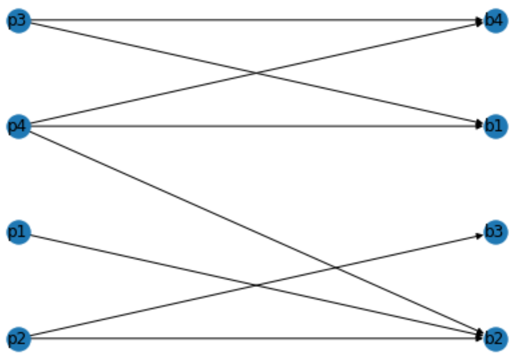
Figure 3.3. Point b result.



3.4 Point c

First of all, I increase capacities from s to $p1, p2, p3, p4$ to allow people to pick more than one book. I set the number of copies of a book changing the capacity of the edges leading to the sink t. The capacities between people and books don't change since each person can still take only one copy of a specific book. Now I can calculate the maximum flow as in point b. As you can see in 3.4, 8 out of 9 books are assigned.

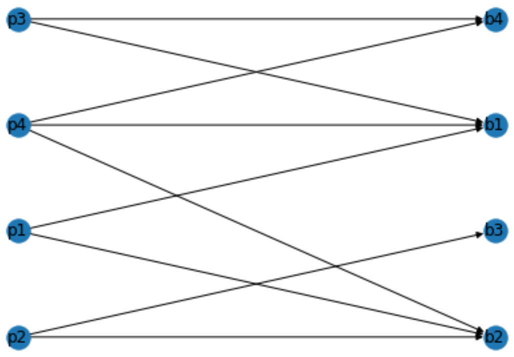
Figure 3.4. Point c result.



3.5 Point d

p_1 is interested in b_1 and b_2, but he cannot buy b_1 because there are no more copies available. At the same time there is one extra copy of b_3 and nobody is interested in it. Therefore the library should sell one copy of b_3 and buy one copy of b_1. Now all 9 copies are sold (3.5).

Figure 3.5. Point d result.



Chapter 4

Exercise 3

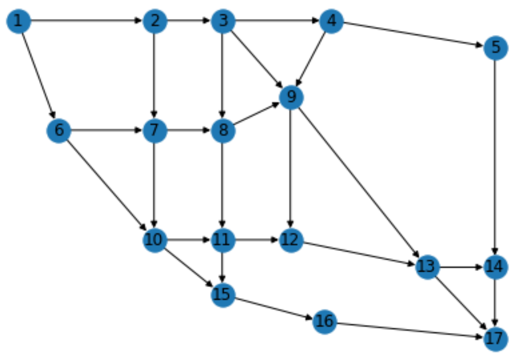
4.1 Theory used

The node-link incidence matrix of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is a matrix B in $\mathcal{R}^{\mathcal{V}x\mathcal{E}}$ whose entries are defined by

$$B_{ie} = \begin{cases} +1 & \text{if } e = (i,j) \text{ for some } j \neq i, \\ -1 & \text{if } e = (j,i) \text{ for some } j \neq i, \\ 0 & \text{if } e = (i,i) \text{ or } e = (j,k) \text{ for some } j \neq i, k \neq i. \end{cases}$$

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, a walk from a node i to a node j is a finite string of nodes $\gamma = (\gamma_0, ..., \gamma_l)$ such that $\gamma_0 = i, \gamma_l = j$, and (γ_{h-1}, γ_h) belongs to $\mathcal{E}$ for all $h = 1, ..., l$, i.e. there is a link between every two consecutive nodes. l is called length of the walk. A path is a walk $\gamma = (\gamma_0, ..., \gamma_l)$ such that $\gamma_h \neq \gamma_k$ for all $0 \leq h \leq k \leq l$ except for possibly $\gamma_0 = \gamma_l$. Let $\Gamma_{o,d}$ be the set of o-d paths and τ_e be the delay function: $\tau_e : [0, +\infty) \to [0, +\infty]$, continuously differentiable in the interval $[0, c_e)$, where $c_e = sup\{x \geq 0 : \tau_e < +\infty\}$. This function returns the delay $\tau_e(f_e)$ encountered by users traversing link e when the flow on that link is f_e. Then, for a given throughput $\nu > 0$, a Wardrop equilibrium is a flow vector $f^{(0)} = A^{(o,d)}z$ where z in $\mathcal{R}^{\Gamma_{o,d}}$ is such that $z \geq 0$, $\mathbb{1}'z = \nu$, and for every path p in $\Gamma_{o,d}$ $z_p > 0 \to \sum_{e \in \mathcal{E}} A^{(o,d)}_{ep} \tau_e(f^{(0)}_e) \leq \sum_{e \in \mathcal{E}} A^{(o,d)}_{eq} \tau_e(f^{(0)}_e)$, $\forall q \in \Gamma_{o,d}$.

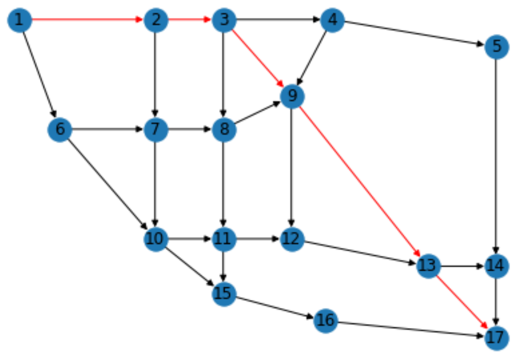
Figure 4.1. Exercise 3 graph.



4.2 Point a

To find the shortest path from 1 to 17 I use the *shortest_path* method. The resulting path is the red one in figure 4.2.

Figure 4.2. Point a resulting path.



4.3 Point b

I assign to each edge the associated capacity contained in *capacities.mat*. Then I can find the maximum flow between node 1 and 17 with the $maximum\_flow$ algorithm.

4.4 Point c

To compute the external inflow ν I do the dot product between B and f, where B is the content of the $traffic.mat$ file and f is the content of $flow.mat$.

4.5 Point d

To find the social optimum f^* with respect to the delays on the different links $d_e(f_e)$ I need to minimize the cost function

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

where C is the content of the file *capacities.mat*, f is the content of $flow.mat$ and l is the content of *traveltime.mat*. This minimization problem has to be solved under the following constraints:

- $B \cdot f = \nu$, where B is the content of $traffic.mat$;
- $f \geq 0$.

In order to use the module *cvxpy*, I write the cost function as

$$cp.multiply(l, c)@(cp.inv\_pos(1 - f/c)) - l@c.$$

4.6 Point e (1)

To find the Wardrop equilibrium, I calculate the cost function in the following way:

$$\begin{aligned}\sum_{e\in\mathcal{E}}\int_0^{f_e} d_e(s)ds &= \sum_{e\in\mathcal{E}}\int_0^{f_e}\frac{l_e}{1-\frac{s}{C_e}}ds \\ &= \sum_{e\in\mathcal{E}} l_e\int_0^{f_e}\frac{1}{1-\frac{s}{C_e}}ds \\ &= \sum_{e\in\mathcal{E}}(-l_eC_e)\int_0^{f_e}\frac{1}{1-\frac{s}{C_e}}(-\frac{1}{C_e})ds \\ &= \sum_{e\in\mathcal{E}}(-l_eC_e)[log(1-\frac{s}{C_e})]_0^{f_e} \\ &= \sum_{e\in\mathcal{E}}(-l_eC_e)[log(1-\frac{f_e}{C_e})-log1] \\ &= \sum_{e\in\mathcal{E}}(-l_eC_e)[log(1-\frac{f_e}{C_e})] \\ &= -\sum_{e\in\mathcal{E}} l_eC_elog(1-\frac{f_e}{C_e})\end{aligned} \tag{4.1}$$

I write the final function (4.2) using the *cvxpy* module:

$$-cp.multiply(l, c)@cp.log(1 - f/c)).$$

Finally I minimize the final cost function following these constraints:

$$B@f == \nu, f >= 0.$$

4.7 Point e (2)

The delay on link e is now given by $d_e(f_e) + w_e$ where $w_e = f_e^* d_e'(f_e^*)$ and f_e^* is the flow at the system optimum. First of all I calculate the derivative of the delay:

$$d_e'(f_e) = -l_e(1 - f_e/c_e)^{-2}(-\frac{1}{c_e}) = \frac{l_e}{c_e}\frac{1}{(1 - f_e/c_e)^2}.$$

Then I use it to compute w_e. Now that I have w_e I can find the new cost function:

$$\begin{aligned}\sum_{e\in\mathcal{E}}\int_0^{f_e}(d_e(s)+w_e)ds &= \sum_{e\in\mathcal{E}}(\int_0^{f_e} d_e(s)ds + \int_0^{f_e} w_eds) \\ &= \sum_{e\in\mathcal{E}}\int_0^{f_e} d_e(s)ds + \sum_{e\in\mathcal{E}}\int_0^{f_e} w_eds \\ &= -\sum_{e\in\mathcal{E}} l_eC_elog(1-\frac{f_e}{C_e}) + \sum_{e\in\mathcal{E}} w_ef_e\end{aligned} \tag{4.2}$$

I write it using the *cvxpy* module

$$-cp.multiply(l, c)@cp.log(1 - f/c) + f@omega$$

and solve the minimization problem following the constraints:

$$B@f == \nu, f >= 0.$$

4.8 Point f

The cost is the total additional delay compared to the total delay in free flow:

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

I need to minimize

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

Using *cvxpy*, I write the function as $cp.multiply(l, c)@(cp.inv\_pos(1 - f/c)) - l@c - f@l$ and I compute the system optimum f^*. Then I compute the new w_e^* as $w_e^* = f_e^* \hat{d}_e'(f_e^*)$. The delay is given by $\hat{d}_e(f_e) = d_e(f_e) - l_e$ so $\hat{d}_e'(f_e) = d_e'(f_e)$. In order to get the new Wardrop equilibrium with tolls f^{w^*}, once I have computed w_e^*, I minimize the function given by:

$$\begin{aligned}\sum_{e\in\mathcal{E}} \int_0^{f_e} (d_e(s) - l_e + w_e^*) ds &= \sum_{e\in\mathcal{E}} \int_0^{f_e} d_e(s) ds - \sum_{e\in\mathcal{E}} \int_0^{f_e} l_e ds + \sum_{e\in\mathcal{E}} \int_0^{f_e} w_e^* ds \\ &= -\sum_{e\in\mathcal{E}} l_e C_e log(1 - \frac{f_e}{C_e}) - \sum_{e\in\mathcal{E}} l_e f_e + \sum_{e\in\mathcal{E}} w_e^* f_e.\end{aligned} \tag{4.3}$$

In *cvxpy* the function to minimize is now $-cp.multiply(l, c)@cp.log(1-f/c)-f@l+f@omega\_star$.