Homework 1

Giuseppe Galilei s295620

14 November 2021

0 Preface

The following homework is submitted alongside three Jupyter notebooks, one for each exercise.
I exchanged ideas for the solution of the exercises 1.a, 1.b and 2.d with Beatrice Alessandra Motetti (s287618), but all the implementations have been carried out individually.

1 Exercise 1

The graph is shown in figure 1.

1.1 A

From the "max-flow min cut" theorem it is known that the maximum flow from node o to node d in the given graph G, is equal to the capacity of a minimum o-d cut. Thus a min-cut capacity equal to zero would mean that node d is not reachable from o.
From this we derive that the minimum capacity to be removed, so that no feasible unitary flow from o to d exists, is equal to the capacity of a min-cut, in this case it is equal to 5.
Moreover to disconnect the two nodes the capacity would need to be removed from links crossing a min-cut.

1.2 B

The proposed solution exploits the "max-flow min-cut" theorem: because the maximum flow from node o to node d graph in the graph G is constrained by the capacity of minimum o-d cuts, the idea is to find such min-cuts and iteratively place units of capacity on edges crossing a min-cut. This is applied recursively until there is not available capacity.

More in detail:
We start by computing all minimum o-d cuts of the graph, on each of these the recursive function $recur\_func$ is called.

e1	e2	e3	e4	e5
1	0	0	0	0
0	1	0	0	0
0	0	0	1	0
0	0	0	0	1

Table 1: capacity placements for 1 additional unit yielding max flow

e1	e2	e3	e4	e5
1	0	0	1	0
1	0	0	0	1
0	1	0	0	1

Table 2: capacity placements for 2 additional units yielding max flow

$recur\_func$ iteratively adds capacity on edges which cross the cut, then computes all new min-cuts and calls itself on each of these min-cuts with a decremented available capacity.
The recursion stops when $recur\_func$ is called with $available\_capacity = 0$, it is then computed the maximum flow between nodes o and d in graph G' obtained by adding all available capacity to G following the placement described in $added\_capacities$.
If the computed flow is equal to the maximum or represents a new maximum, information on edges to whom assign the capacity in order to obtain maximum flow is stored in the $added\_capacities\_global$ list. Maximum flow value is stored in the variable max_flow_global.

In this case adding 1 unit of available capacity doesn't impact the maximum flow, which remains equal to 5 (Table 1).

Note: The solution of this question and the following questions C and D is based on the same code.

1.3 C

Following the same idea of question B we obtain that by adding 2 units of capacity the maximum flow is equal to 6. Such result can be obtained by placing capacities following 3 different combinations (Table 2).

1.4 D

Following the same idea of question B we obtain that by adding 4 units of capacity the maximum flow is equal to 7. Such result can be obtained by placing capacities following 7 different combinations (Table 3).
The maximum sum of cut capacities is equal to 30, such value is found with all six combinations computed in the last question.

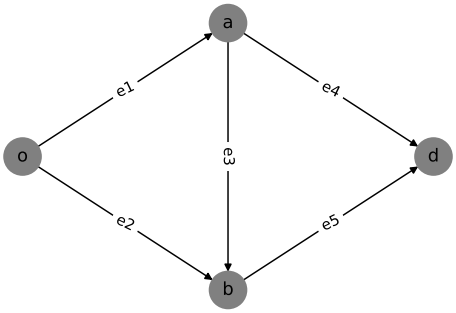

Figure 1: Graph G for exercise 1

e1	e2	e3	e4	e5
2	0	0	0	2
2	0	0	1	1
0	2	0	0	2
1	1	0	1	1
1	1	0	0	2
2	0	0	2	0

Table 3: capacity placements for 4 additional units yielding max flow

2 Exercise 2

2.1 A

The interest pattern is shown in figure 2.
In the resulting bipartite graph $G = (V, \varepsilon)$ with $V = V_0 \cup V_1$, an edge from a node p in V_0 to a node b in V_1 represents that person p is interested in book b.

2.2 B

Given that $|V_0| = |V_1|$, a perfect matching can exist. In order to find it, an analogy between maximal flows and perfect matching is exploited. Starting from G, an auxiliary graph $G' = (V', \varepsilon')$ having $V' = V \cup s \cup t$ and edge set ε' is constructed as follows:

- for every node $p \in V_0$, add an edge (s,p), with capacity 1
- for every node $b \in V_1$, add an edge (b,t), with capacity 1
- for every edge (i, j) in G, add a directed edge (i, j) in G' with capacity 1

The obtained graph G' is shown in Figure 3.
It is now possible to compute a maximum flow in G', as expected the obtained value is $4 = |V_0| = |V_1|$. A possible flow giving this result exploits edges $(p1, b1), (p2, b3), (p3, b4), (p4, b2)$, thus showing a perfect matching:

$$p1 \to b1 \;\; p2 \to b3 \;\; p3 \to b4 \;\; p4 \to b2$$

2.3 C

The proposed situation can be solved by using an approach similar to that of the previous question. Starting from G, an auxiliary graph $G'' = (V'', \varepsilon'')$ having $V'' = V \cup s \cup t$ and edge set ε'' is constructed as follows:

- for every node $p \in V_0$, add an edge (s,p), with capacity equal to the out_degree of node p (number of books each person is interested in)
- for every node $b \in V_1$, add an edge (b,t), with capacity equal to the available copies of book b
- for every edge (i, j) in G, add a directed edge (i, j) in G'' with capacity 1 (this models the constraint that each person can take at most one copy for each book)

The maximum flow on G'' has value 8, which means 8 copies of books have been assigned in total. The corresponding assignments are:

$$p1 \to b1, b2 \;\; p2 \to b2, b3 \;\; p3 \to b4 \;\; p4 \to b1, b2, b4$$

	b1	b2	b3	b4
old inventory	2	3	2	2
beginning state	-1	0	1	0
optimized inventory	3	3	1	2

Table 4:

2.4 D

A state value is computed for each book as a difference between the available copies, as assigned in question C, and the in_degree of the relative node (i.e. the number of people interested in it). Such value will be positive in case there is a surplus for such book, negative in case there is an unfulfilled demand. The idea is to use a states array to support the buying and selling of copies for the different books, while keeping the total quantity of copies constant.
In table 4 is shown the comparison between the old inventory (assigned in question C), the beginning state and the resulting optimized inventory. Graph G'' is then updated by assigning to edges $b \rightarrow t$, for each book, a capacity equal to the number of copies of book b in the optimized inventory.
As for the previous question, we determine the assignment by computing maximum flow, this time 9 copies are assigned and each person receives all books of interest:

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

3 Exercise 3

3.1 A

The shortest path problem can be solved as a flow problem, imagining of sending one unit of flow from an origin node to a destination one with the aim of minimizing a certain cost function (intuitively it could be the length of the links being used, i.e. of the links where flow is greater than zero). The highway network is modelled as a graph $G = (V, \varepsilon)$ where nodes are intersections and edges are highways, C_e represents the capacity, l_e the travel time and f_e the flow for each edge. In this case the problem is solved through cvxpy and has the following specifications:

- Objective function: $\min \sum_{e \in \varepsilon} f_e l_e$
- Constraints: $Bf = \nu, \; f \geq 0, f \leq C$

i.e. Find a flow f that minimizes the sum of delays on an empty network while satisfying flow conservation constraint, non negativity of flow and capacity constraint.

It is obtained that the shortest path has value 0.53 and uses links 1,2,9,12,25

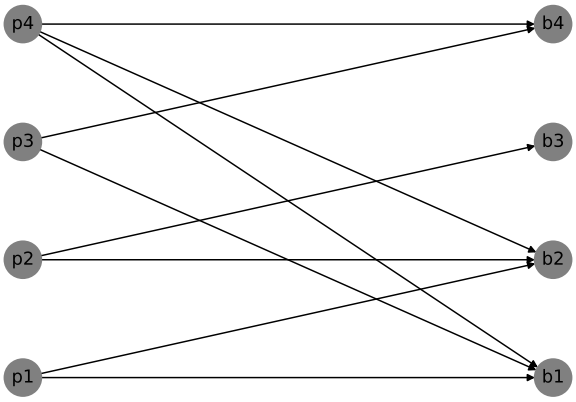

Figure 2: Bipartite graph G representing the interest pattern

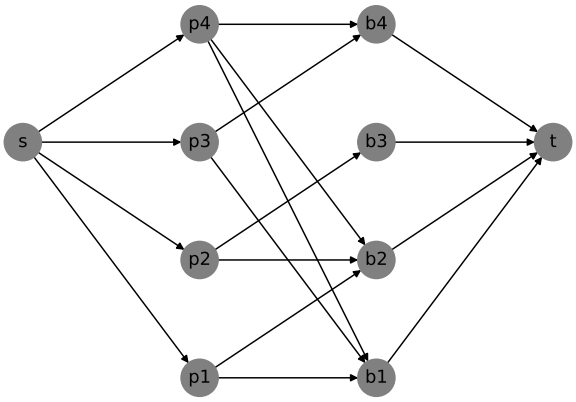

Figure 3: Auxiliary graph G'

3.2 B

The maximum flow between node 1 and 17 is found by means of a networkx function and has value 22448.

3.3 C

The requested exogenous flow vector is computed as $\nu = Bf$ and it is equal to:
$[16806,\ 8570,\ 19448,\ 4957,\ -746,\ 4768,\ 413,\ -2,\ -5671,\ 1169,\ -5,$
$-7131,\ -380, -7412,\ -7810,\ -3430,\ -23544]$

3.4 D

The social optimum flow represents the flow that minimizes the total delay of all users, in this case it would mean that highways are used at their best and provide the best service to the overall community. The problem has the following specifications:

- Objective function: $\min \sum_{e \in \mathcal{E}} \left(\frac{l_e C_e}{1 - f_e / C_e} - l_e C_e \right)$
- Constraints: $Bf = \nu,\ f \geq 0, f \leq C$

The social optimum cost is 25943.6, for the flow values refer to the Jupyter Notebook.

3.5 E

Finding a Wardrop equilibrium means finding a path flow such that if a path is used, then its cost is minimal. This way it is possible to model the selfish behaviour of drivers choosing the best path for themselves, disregarding the consequences on the rest of society. The problem has the following specifications:

- Objective function: $\min \sum_{e \in \mathcal{E}} \int_0^{f_e} d_e(s) ds = \min \sum_{e \in \varepsilon} -l_e C_e \log \left(1 - \frac{f_e}{C_e}\right)$
- Constraints: $Bf = \nu,\ f \geq 0, f \leq C$

The user optimum cost is 15729.6, for the flow values refer to the Jupyter Notebook.
The social cost at Wardrop is $f^{(0)} \frac{le}{1 - \frac{f^{(0)}}{C}} = 26293$, where $f^{(0)}$ is the flow computed for Wardrop equilibrium. The Wardrop cost is higher than the social optimum cost, as expected.
This can be shown as the price of anarchy $PoA = \frac{Wardrop\_cost}{SocialOptimum\_cost} = 1.013$

3.6 F

Introducing tolls is a way to solve the problem of selfish users in the Wardrop situation. Marginal tolls w_e are added to each link, so that selfish users pay

not only for their delay but also for the cost they induce on other users due to their choices. With the new objective function users will minimize the cost the system pays for their choices, leading to social optimum.
It is defined $w_e = f_e^*(d_e(f_e^*))' = f_e^* \frac{l_e C_e}{(C_e - f_e^*)^2}$ where f_e^* is the social optimum flow. The problem has the following specifications:

- Objective function: $\min \sum_{e \in \mathcal{E}} -l_e C_e \log\left(1 - \frac{f_e}{C_e}\right) + w_e f_e$
- Constraints: $Bf = \nu,\ f \geq 0, f \leq C$

The user cost for Wardrop with tolls is 61885, significantly higher than that for Wardrop computed before, however the system cost reduces to 25943 and coincides to the social optimum cost as expected. For the flow values refer to the Jupyter Notebook.

3.7 G

The reasoning for this question is the same as for the previous ones, it is reported a summary of the specification of the solved problems along with the results.

3.7.1 Social Optimum

The problem has the following specifications:

- Objective function: $\min \sum_{e \in \mathcal{E}} \frac{l_e C_e}{1 - f_e/C_e} - l_e C_e - f_e l_e$
- Constraints: $Bf = \nu,\ f \geq 0, f \leq C$

The social optimum cost is 15095

3.7.2 Wardrop

- Objective function: $\min \sum_{e \in \mathcal{E}} -l_e C_e \log\left(1 - \frac{f_e}{C_e}\right) - f_e l_e$
- Constraints: $Bf = \nu,\ f \geq 0, f \leq C$

The system Wardrop cost is 15148. Price of anarchy is 1.003

3.7.3 Wardrop with Tolls

It is defined $w_e = f_e^*(d_e(f_e^*))' = f_e^* \frac{l_e C_e}{(C_e - f_e^*)^2}$ where f_e^* is the social optimum flow. The problem has the following specifications:

- Objective function: $\min \sum_{e \in \mathcal{E}} -l_e C_e \log(1 - \frac{f_e}{C_e}) - f_e l_e + w_e f_e$
- Constraints: $Bf = \nu,\ f \geq 0, f \leq C$

The user cost in Wardrop with tolls is 50795 while the system "Wardrop with tolls" cost is 15095.

A price of anarchy almost equal to one indicates that this system behaves well also without tolls.