Network Dynamics and Learning, Homework 1

Giovanni Cadau *Politecnico di Torino*
Turin, Italy
s304861@studenti.polito.it

November 13, 2022

1 Exercise 1

Fig 1 shows a network with set of nodes $\mathcal{V} = \{o, a, b, c, d\}$, set of links $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ and corresponding capacities $c_1 = c_3 = c_5 = 2$ and $c_2 = c_4 = c_6 = 1$.

1.1 Point a

To determine the minimum aggregate capacity that needs to be removed for no feasible flow from node o to node d to exist, it is possible to exploit the *Max-Flow Min-Cut Theorem*, which states that, given a multigraph $\mathcal{G} = (\mathcal{V}; \mathcal{E}; c)$ with a capacity vector c > 0, and two distinct nodes o and d, the *maximum throughput* from o to d equals the *min cut capacity* of the network.

As a consequence, the minimum total capacity to remove from the network in order to make node d not reachable from node o coincides with the min-cut capacity $c^*_{o,d}$ of the network. If removing all the links in a minimum capacity o-d cut (thus removing a total capacity of $c^*_{o,d}$), then the min-cut capacity of the resulting network would be 0 and no feasible flow from node o to node d in such resulting graph there would exist expect for the all-zero one: d would not be reachable from o. If a total capacity of less than $c^*_{o,d}$ were removed in any possible way, the min-cut capacity of the resulting graph would still be positive: there would exist a feasible flow with positive throughput from the two nodes.

In the network of Fig 1, the maximum throughput $\tau^*_{o,d}$ from o to d is 3, the corresponding minimum cut capacity $c^*_{o,d}$ is (as expected) 3 and there exists a cut $\mathcal{U}$ with capacity $\mathcal{C}_\mathcal{U} = c^*_{o,d}$. The partition of the node set $\mathcal{V}$ related to the cut $\mathcal{U}$ is the following: $\mathcal{U} = \{b, o, a, c\}$, $\mathcal{U}^c = \{d\}$. Fig 2 shows in green all links from a node in $\mathcal{U}$ to a node in $\mathcal{U}^c$ with their capacity: $\{(o, d), (b, d), (c, d)\}$, each with capacity 1.

The capacity of each link belonging to the set of links with a node in $\mathcal{U}$ to a node in $\mathcal{U}^c$ has been decreased step by step (i.e., a unit of capacity each step is removed). As expected, as long as the total removed capacity is different from 3, the maximum throughput and minimum cut capacity are different from 0 and there is a flow from o to d different from the all-zero one. When the total capacity removed reaches $c^*_{o,d} = 3$, only the all-zero flow exists in the network: no feasible flow from node o to node d exists.

1.2 Point b

As a consequence of the *Max-Flow Min-Cut Theorem* previously stated, it is possible to remove capacity from the network without affecting the maximum throughput value, if the capacity is removed from a link not belonging to a minimum o-d cut (i.e., links from a node in $\mathcal{U}$ to a node in $\mathcal{U}^c$, where $\mathcal{U}$ and $\mathcal{U}^c$ represent the subsets belonging to the partition of the node set such that o belongs to $\mathcal{U}$ and d to $\mathcal{U}^c$).

To determine the maximum capacity that can be removed without affecting the maximum throughput value from o to d, the following recursive algorithm has been implemented. Firstly, all links not belonging to a minimum cut are extracted. If the set is not empty, one unit of capacity is removed from a link in the set. The total capacity removed is updated and then the procedure is recursively repeated in the resulting graph. The recursion stops when the set of edges belonging to a min cut is empty. When a recursive descent ends, a backtrack phase occurs (the unit of capacity previously

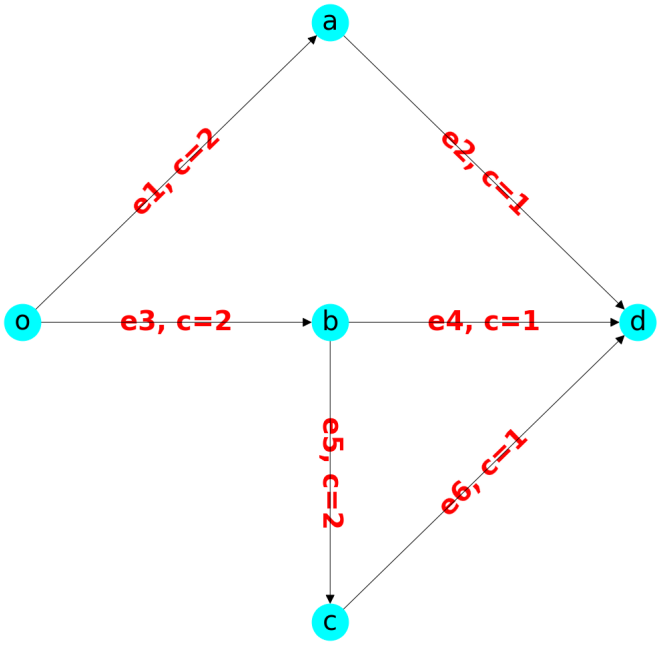

Figure 1: Network of exercise 1 with link capacities c

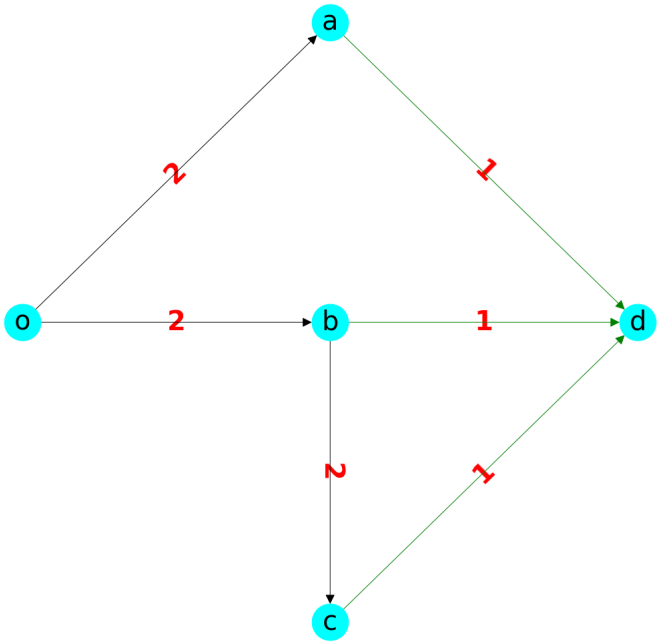

Figure 2: Network of exercise 1 with link capacities on each link and links belonging to a minimum o-d cut in green

removed is assigned to the link from which it had been removed and the total capacity removed is incremented) and another edge in the set is considered.

In the following, 1 recursive descent is reported as an example in the network reported in Fig 1. In the attached code the entire output of the algorithm is available.

In the first step of the algorithm, the following *o-d* cuts are in the network.

o-d cuts, with their capacity:
cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 3
cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 3
cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 4
cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 4
cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 4
cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 4
cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 5
cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 5
edges not belonging to an o-d cut with minimum capacity:
('o', 'a')
('b', 'c')

The capacity of edge ('o', 'a') is decreased by 1 and the procedure is repeated in the resulting graph. The cuts in the network are now the following:

o-d cuts, with their capacity:
cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 3
cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 3
cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 3
cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 3
cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 4
cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 4
cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 4
cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 4
edges not belonging to an o-d cut with minimum capacity:
('b', 'c')

The capacity of edge ('b', 'c') is decreased by 1 and the procedure is repeated in the resulting graph. The cuts in the network are now the following:

o-d cuts, with their capacity:
cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 3
cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 3
cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 3
cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 3
cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 3
cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 3
cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 4
cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 4
edges not belonging to an o-d cut with minimum capacity:
– no edges.

Since the set is empty, the recursive descent ends: the maximum capacity removed is 2.

The maximum aggregate capacity that can be removed in the overall algorithm found is 2: if an additional unit is removed, the maximum throughput decreases.

1.3 Point c

If extra units of capacity are available, they can be distributed among the links of the network to increase the throughput that can be sent from node *o* to node *d*. In order to maximize such value, the extra unit of capacity must be assigned to links in such a way to increase the capacity of the minimum cut. This can be seen as consequence of the *Max-Flow Min-Cut Theorem* previously stated: if the capacity of the min cut is increased, than also the throughput increases.

In the network of Fig 1, if no extra unit of capacity is available, the maximum throughput equals the minimum cut capacity, which is equals 3, as computed before.

If 1 extra unit of capacity is available, it must be assigned in order to increase the capacity of the min cut. The following are the cuts in the network:

o-d cuts, with their capacity and the set of edges from a node in the set containing o to a node in the set containing d:

cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 3 ; edges of the cut: [('a', 'd'), ('b', 'd'), ('c', 'd')]
cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 3 ; edges of the cut: [('o', 'b'), ('a', 'd')]
cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 4 ; edges of the cut: [('a', 'd'), ('b', 'c'), ('b', 'd')]
cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 4 ; edges of the cut: [('o', 'b'), ('a', 'd'), ('c', 'd')]
cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 4 ; edges of the cut: [('o', 'a'), ('b', 'd'), ('c', 'd')]
cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 4 ; edges of the cut: [('o', 'a'), ('o', 'b')]
cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 5 ; edges of the cut: [('o', 'a'), ('b', 'c'), ('b', 'd')]
cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 5 ; edges of the cut: [('o', 'a'), ('o', 'b'), ('c', 'd')]

edges belonging to an o-d cut with minimum capacity:

('a', 'd')
('b', 'd')
('c', 'd')
('o', 'b')
('a', 'd')

From the powerset of the set of edges belonging to an o-d cut with minimum capacity, it is chosen the set of minimum cardinality (except from the empty set) such that, by increasing capacity of edges in that set, the capacity of the min cut is increased. All sets belonging to the powerset are generated (stopping if a set with lower cardinality satisfying the previous condition has already been found). In the code attached it is possible to find all sets with the corresponding min cat capacity, here it is reported the only set satisfying the condition:

capacity of: ('a', 'd') edge is increased by 1.

o-d cuts, with their capacity after extra capacity assigned.

cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 4
cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 4
cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 4
cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 4
cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 5
cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 5
cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 5
cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 5

This happens since edge (a, d) is contained in both sets of edges of min *o-d* cuts: by increasing only its capacity, the minimum cut capacity becomes 4, which is the maximum throughput if 1 extra unit of capacity is available.

If 2 extra units of capacity are available (1 extra unit with respect to the previous case), the additional extra unit must be assigned in order to increase the capacity of the min cut, which is different from that computed at previous step. The following are the cuts in the network after the modification made in the previous step:

o-d cuts, with their capacity and the set of edges from a node in the set containing o to a node in the set containing d:

cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 4 ; edges of the cut: [('a', 'd'), ('b', 'd'), ('c', 'd')]
cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 4 ; edges of the cut: [('o', 'b'), ('a', 'd')]
cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 4 ; edges of the cut: [('o', 'a'), ('b', 'd'), ('c', 'd')]
cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 4 ; edges of the cut: [('o', 'a'), ('o', 'b')]
cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 5 ; edges of the cut: [('a', 'd'), ('b', 'c'), ('b', 'd')]
cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 5 ; edges of the cut: [('o', 'b'), ('a', 'd'), ('c', 'd')]
cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 5 ; edges of the cut: [('o', 'a'), ('b', 'c'), ('b', 'd')]
cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 5 ; edges of the cut: [('o', 'a'), ('o', 'b'), ('c', 'd')]

edges belonging to an o-d cut with minumum capacity:

('a', 'd')
('b', 'd')
('c', 'd')
('o', 'b')

('a', 'd')
('o', 'a')
('b', 'd')
('c', 'd')
('o', 'a')
('o', 'b')

As in previous step, from the powerset of the set of edges belonging to an o-d cut with minimum capacity, it is chosen the set of minimum cardinality (except from the empty set) such that, by increasing capacity of edges in that set, the capacity of the min cut is increased. All sets belonging to the powerset are generated (stopping if a set with lower cardinality satisfying the previous condition has already been found). In the code attached it is possible to find all sets with the corresponding min cat capacity, here it is reported the only sets satisfying the condition:

capacity of:
('o', 'a') edge is increased by 1.
('a', 'd') edge is increased by 1.
o-d cuts, with their capacity after extra capacity assigned.
cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 5
cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 5
cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 5
cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 5
cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 6
cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 6
cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 6
cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 6
capacity of:
('b', 'd') edge is increased by 1.
('o', 'b') edge is increased by 1.
o-d cuts, with their capacity after extra capacity assigned.
cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 5
cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 5
cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 5
cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 5
cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 6
cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 6
cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 6
cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 6
capacity of:
('c', 'd') edge is increased by 1.
('o', 'b') edge is increased by 1.
o-d cuts, with their capacity after extra capacity assigned.
cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 5
cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 5
cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 5
cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 5
cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 5
cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 5
cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 7
cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 7

No sets with lower cardinality than 2 are available: 1 additional extra unit of capacity doesn't change the maximum throughput; if there are 2 additional extra units of capacity, 3 possible sets with cardinality equals 2 are able to generate a minimum cut with capacity equals 5, which is the maximum throughput with 3 total extra units of capacity available.

- First and second possibilities are equivalent: they leads to the same situation. If one of them is chosen and 4 extra units of capacity are available (1 extra unit with respect to the previous case), the same procedure applied in previous steps can be applied. All detailed outputs are reported

in the code attached. Here is reported only the sets of edges of minimum cardinality such that, by increasing capacity of edges in that set, the capacity of the min cut is increased:

capacity of:

('o', 'a') edge is increased by 1.

('a', 'd') edge is increased by 1.

o-d cuts, with their capacity after extra capacity assigned.

cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 6

cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 6

cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 6

cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 6

cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 7

cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 7

cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 7

cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 7

capacity of:

('b', 'd') edge is increased by 1.

('o', 'b') edge is increased by 1.

o-d cuts, with their capacity after extra capacity assigned.

cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 6

cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 6

cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 6

cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 6

cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 7

cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 7

cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 7

cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 7

capacity of:

('c', 'd') edge is increased by 1.

('o', 'b') edge is increased by 1.

o-d cuts, with their capacity after extra capacity assigned.

cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 6

cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 6

cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 6

cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 6

cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 6

cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 6

cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 8

cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 8

All possibilities are exactly the same of step before, with min capacity increased by 1: all future steps choosing these edges with 2 additional capacity available, will lead to the same results: each step min capacity will be added by 1.

- If third possibility is chosen and 4 extra units of capacity are available (1 extra unit with respect to the previous case), the same procedure applied in previous steps can be applied. All detailed outputs are reported in the code attached. Here is reported only the sets of edges of minimum cardinality such that, by increasing capacity of edges in that set, the capacity of the min cut is increased:

  capacity of:

  ('o', 'a') edge is increased by 1.

  ('a', 'd') edge is increased by 1.

  o-d cuts, with their capacity after extra capacity assigned.

  cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 6

  cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 6

  cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 6

  cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 6

  cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 6

  cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 6

  cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 8

  cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 8

  capacity of:

  ('b', 'd') edge is increased by 1.

  ('o', 'b') edge is increased by 1.

  o-d cuts, with their capacity after extra capacity assigned.

  cut: [['o', 'a', 'b', 'c'], ['d']] ; cap: 6

  cut: [['o', 'a', 'b'], ['d', 'c']] ; cap: 6

  cut: [['o', 'a'], ['b', 'd', 'c']] ; cap: 6

  cut: [['o', 'b', 'c'], ['a', 'd']] ; cap: 6

  cut: [['o', 'b'], ['a', 'd', 'c']] ; cap: 6

  cut: [['o'], ['a', 'b', 'd', 'c']] ; cap: 6

  cut: [['o', 'a', 'c'], ['b', 'd']] ; cap: 8

  cut: [['o', 'c'], ['a', 'b', 'd']] ; cap: 8

  All possibilities are equivalent and lead to the same configuration of previous step, with capacities increased by 1: all future steps choosing these edges with 2 additional capacity available, will lead to the same results: each step min capacity will be increased by 1.

To sum up, both 1st (or 2nd) and 3rd possibilities lead to the following scenario: each time the capacity is increased by 2 (min possible cardinality of the edge set such that, by increasing capacity of such edges, the capacity of the min cut is increased by 1), the throughput is increased by 1.

Fig 3 summarizes all considerations made above: it shows the maximum throughput from node o to node d as a function of the total extra units of capacity available.

2 Exercise 2

Two following sets represent a list of people p and a list of books b: $p = \{p_1, p_2, p_3, p_4\}$, $b = \{b_1, b_2, b_3, b_4\}$. Each person p_i is interested in a subset of books $\{b_j\}$, specifically: $interests = \{p_1 \to \{b_1, b_2\}, p_2 \to \{b_2, b_3\}, p_3 \to \{b_1, b_4\}, p_4 \to \{b_1, b_2, b_4\}\}$.

The scenario presented can be represented by a simple bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with node set $\mathcal{V}$ partitioned as $\mathcal{V} = V_0 \cup V_1$, with $V_0 \cap V_1 = \emptyset$, and no edges between nodes of the same set. Let V_0 be the people set p, V_1 be the books set b and $\mathcal{E}$ be constructed with elements reported in $interests$.

Fig 4 shows the graph $\mathcal{G}$.

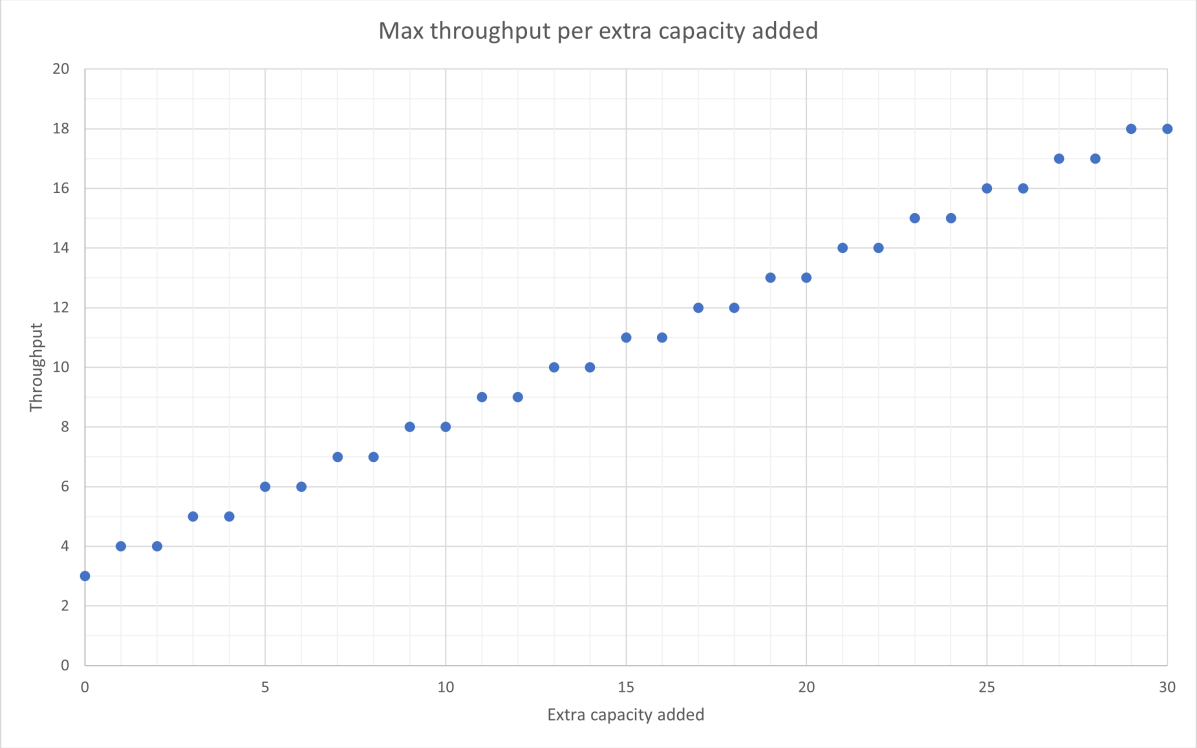

Figure 3: Maximum throughput with respect to total extra units of capacity available

2.1 Point a

In a simple bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *matching* is a subset of edges $M \subseteq E$ such that no two edges in M share a common node. In particular, in a simple bipartite graph with node subsets V_0 and V_1, for $h = 0, 1$, a matching M is defined V_h-perfect if every node in V_h is matched in M.

In the situation described, an interesting question is the following: *Is there a way to assign to every person a book of interest?*, which can be addressed by finding a V_0-perfect matching.

Firstly, a V_0-perfect matching *can* exist, since $|V_0| = |V_1|$. In order to find a perfect matching, if any, the following analogy between maximal flows and perfect matching can be exploited.

Given a simple bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $\mathcal{G}_1$ be the directed capacitated graph, with node set $V \cup s \cup t$, and edge set constructed as follows:

- for every node $n \in V_0$, an edge (s, n) is added, with capacity 1
- for every node $n \in V_1$, an edge (n, t) is added, with capacity 1
- for every undirected edge (i, j) in $\mathcal{G}$, a directed edge (i, j) in $\mathcal{G}_1$ is added, with capacity 1

A V_0-perfect matching on $\mathcal{G}$ exists if and only if it there exists a flow with throughput $|V_0|$ on the network $\mathcal{G}_1$. If it exists, given a maximum flow with throughput $|V_0|$, the associated V_0-perfect matching can be found by selecting all the edges (i, j), $i \in V_0, j \in V_1$ such that $f_{(i,j)} = 1$.

Fig 5 shows the graph $\mathcal{G}_1$, constructed from $\mathcal{G}$ as explained before.

The maximum throughput in $\mathcal{G}_1$ from node s to node t is 4. Since there exists a flow with throughput $|V_0| = 4$ on the network $\mathcal{G}_1$, a V_0-perfect matching on $\mathcal{G}$ exists. Given the maximum flow with throughput equals 4 in $\mathcal{G}_1$, the associated V_0-perfect matching in $\mathcal{G}$ is the following: $M = \{p_1 \to b_1, p_2 \to b_3, p_3 \to b_4, p_4 \to b_2\}$. Fig 6 shows the matching M in the graph $\mathcal{G}$. Each colored edge (i.e., with a color different from black) is an element belonging to a V_0-perfect matching M.

2.2 Point b

The following changes are made to the situation described above:

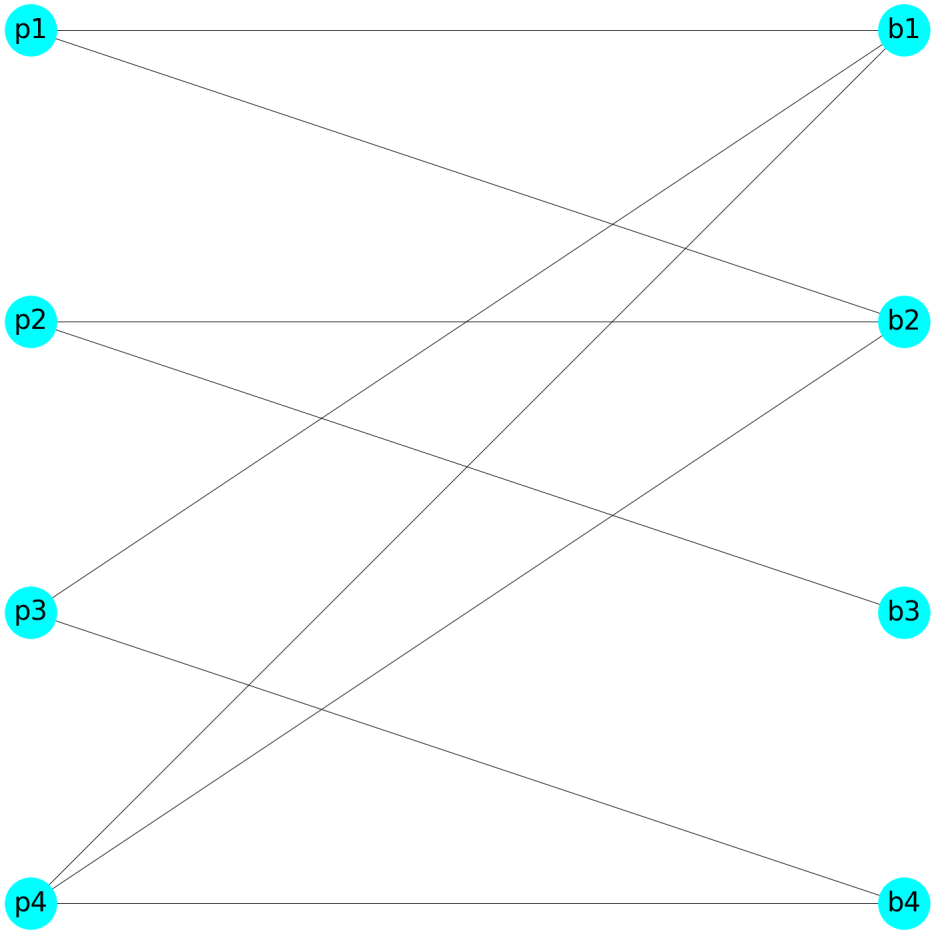

Figure 4: Network $\mathcal{G}$ of exercise 2, with set V_0 on the left, and set V_1 on the right

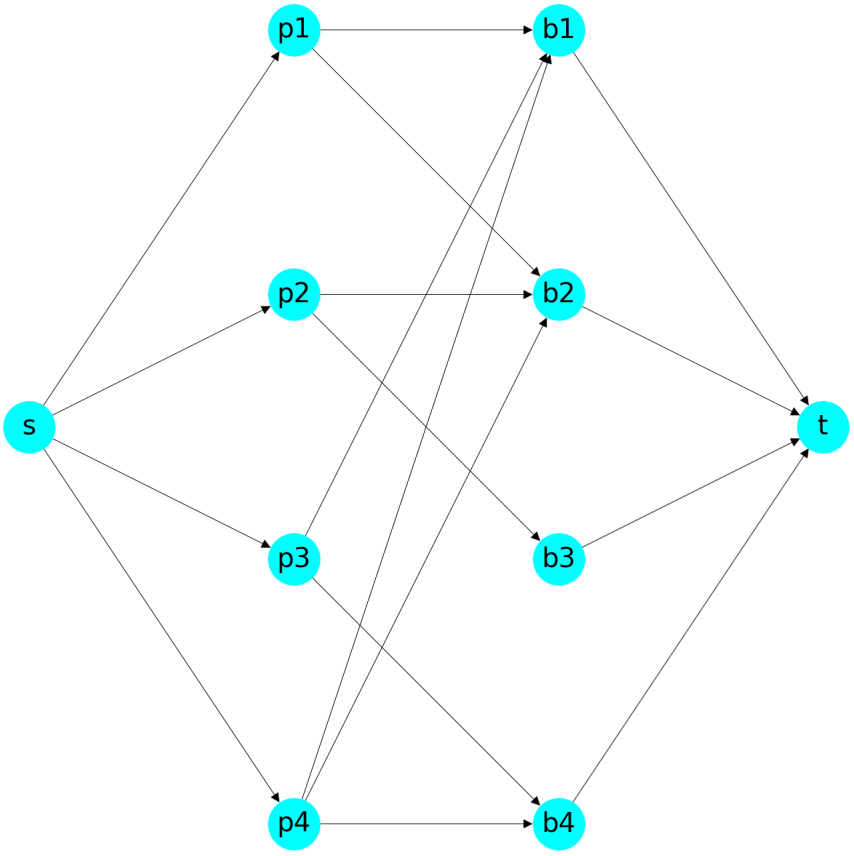

Figure 5: Network $\mathcal{G}_1$ of exercise 2, with set V_0 on the left, and set V_1 on the right

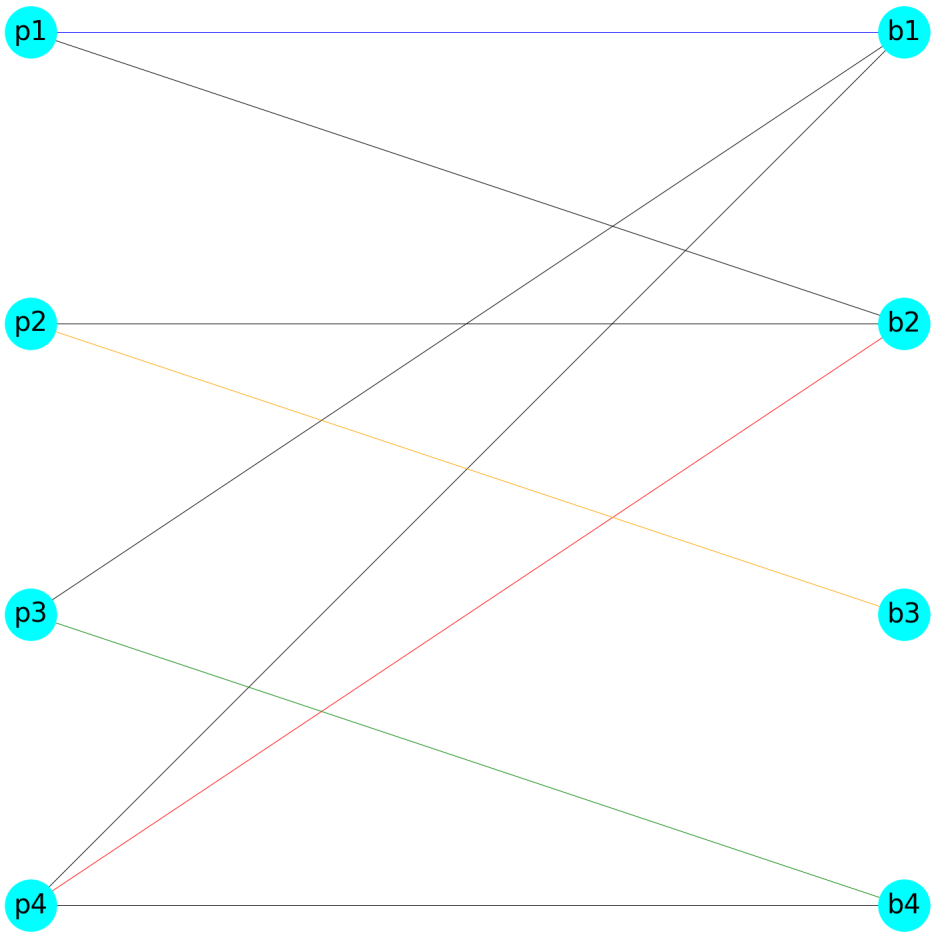

Figure 6: Network $\mathcal{G}$ of exercise 2. Each colored edge is an element belonging to a V_0-perfect matching M

- the are multiple copies of books available, with the following distribution: $copies = \{2, 3, 2, 2\}$
- each person can take an arbitrary number of different books

In order to establish *how many books of interest can be assigned in total*, the following changes in the graph $\mathcal{G}_1$ can be made.

- The capacity of each link (b_j, t) is set to $copies_j$, $j = 1, ..4$.
- The capacity of each link (p_i, b_j), $i = 1, ..4$, $j = 1, ..4$, if the link exists, is set to 1.

The maximum throughput in $\mathcal{G}_1$ from node s to node t equals the maximum number of books of interest that can be assigned in total. This can be explained with the following considerations:

- The flow on a link (p_i, b_j) represents the number of copies of book b_j assigned to person p_i.

  Due to the *flow balance equation*, the total inflow in a node a, resulting from both possible exogenous inflow and flows from incoming links, equals the total outflow from a, resulting from both possible external outflow and flows towards outgoing links.

  Since both exogenous inflow and external outflow in/from node b_j or p_i equals 0, the total flows from incoming links in node b_j equals the total flows towards outgoing links from b_j and the same happens for p_i.

  As a consequence, the total flows towards the only outgoing link from b_j, (b_j, t), equals the total flows from incoming links in node b_j and the total flows towards the only outgoing link from b_j, (b_j, t), represents the total number of copies of book b_j assigned to people in $\{p_i\}$.

  Similarly, the total flows from the only incoming link in p_i, (s, p_i), equals the total flows towards outgoing links from node p_i and the total flows from the only incoming link in p_i, (s, p_i), represents the total number of copies of books in $\{b_j\}$ assigned to person p_i.

  Since exogenous inflow in node t equals 0 and no outgoing links from t exists, the total flows from incoming links in node t, (b_j, t), equals the external outflow from node t and they represent the total number of copies of all books b assigned to all people p.

  Since external outflow from node s equals 0 and no incoming links to s exists, the exogenous inflow in node s equals total flows towards outgoing links from node s, (s, p_i), and they represent the total number of copies of all books b assigned to all people p.

  As a consequence, the throughput τ of a flow from s to t represents the total number of copies of all books assigned to all people.

- The capacity of each link (b_j, t) represents the maximum number of copies that can be taken of each book b_j. If the total flows towards the only outgoing link from b_j, (b_j, t), can not exceed the capacity of such link, then the total flows from incoming links in node b_j can not exceed the same value. Since the total flows from incoming links in node b_j represents the total number of copies of book b_j assigned to people in $\{p_i\}$, such value can not exceed the maximum number of copies that can be taken of each book b_j.

- The capacity of each link (p_i, b_j), set to 1 if the links exists, represents the maximum number of copies of a specific book b_j that can be taken by a specific person p_i: each person can not take more than a copy of the same book.

If there exist a flow in $\mathcal{G}_1$ from node s to node t with throughput τ, then there exist an assignment of books to people according to their interests and the total number of books assigned equals τ. If the flow is a maximum flow, the maximum possible number of books is assigned.

The maximum throughput in $\mathcal{G}_1$ from node s to node t is 8. The associated maximum flow carry the following information:

Book taken by people.

p1 $\rightarrow$ b1 : 1
p1 $\rightarrow$ b2 : 1
p2 $\rightarrow$ b2 : 1
p2 $\rightarrow$ b3 : 1
p3 $\rightarrow$ b1 : 1

p3 → b4 : 1
p4 → b1 : 0
p4 → b2 : 1
p4 → b4 : 1
s → p1 : 2 — number of different books taken by p1
s → p2 : 2 — number of different books taken by p2
s → p3 : 2 — number of different books taken by p3
s → p4 : 2 — number of different books taken by p4
b1 → t → 2 — number of copies of book b1 taken by people
b2 → t → 3 — number of copies of book b2 taken by people
b3 → t → 1 — number of copies of book b3 taken by people
b4 → t → 2 — number of copies of book b4 taken by people

Fig 7 shows the flow in the graph $\mathcal{G}_1$. Each link is labeled with a label in the form *number of copies/capacity of the link*.

2.3 Point c

To maximize the number of assigned books, the library can sell a copy of b_3 and buy a copy of b_1. The following statement is supported by the following considerations:

- The flow on link (b_3, t) can not be greater than 1, since b_3 has only 1 edge $e = (i, b_3)$ with capacity 1: it means that the number of copies of b_3 taken can not be greater than 1 since only 1 copy has been requested.

  Since the capacity of (b_3, t) is 2, 1 unit of capacity can be removed: 1 copy of b_3 can be sold by the library since no one has requested it.

- The flow on edge (b_1, t) can't be greater than 2, since its capacity is 2: it means that the number of copies of b_1 taken can not be greater than 2 since there are only 2 copies available. b_1 has 3 edges $e = (i, b_1)$ with capacity 1 and only 2 of them will have flow equals 1 (only 2 people can take a copy of b_1).

  The capacity of (b_1, t) can be increased by 1 unit: 1 copy of b_1 can be bought by the library since there are 3 people requesting a copy but only 2 copies available.

The maximum throughput in $\mathcal{G}_1$ from node s to node t is now 9, which is the maximum number of copies available among all books.

Fig 8 shows the new flow in the graph $\mathcal{G}_1$. Each link is labeled with a label in the form *number of copies/capacity of the link*. In this graph, each flow of each link equals the capacity of its link.

3 Exercise 3

Fig 9 shows a network $\mathcal{G}$ representing part of the highway network in Los Angeles, with set of nodes $\mathcal{V}$ labeled from 1 to 17, a set of links $\mathcal{E}$ labeled from 1 to 28 (e_i), with corresponding maximum flow capacity c_{e_i}, minimum travelling time l_{e_i} experienced when the road is empty.

3.1 Point a

The shortest path between node 1 and 17 with respect to l_{e_i} is equivalent to the path with shortest travelling time in an empty network.

As a consequence, the shortest path can be computed through the following optimization problem:

$$f^* \in \underset{\substack{f \in \mathrm{R}_+^{\mathcal{E}} \\ Bf = \nu}}{\arg\min} \quad \sum_{e \in \mathcal{E}} \psi_e(f_e). \tag{1}$$

where:

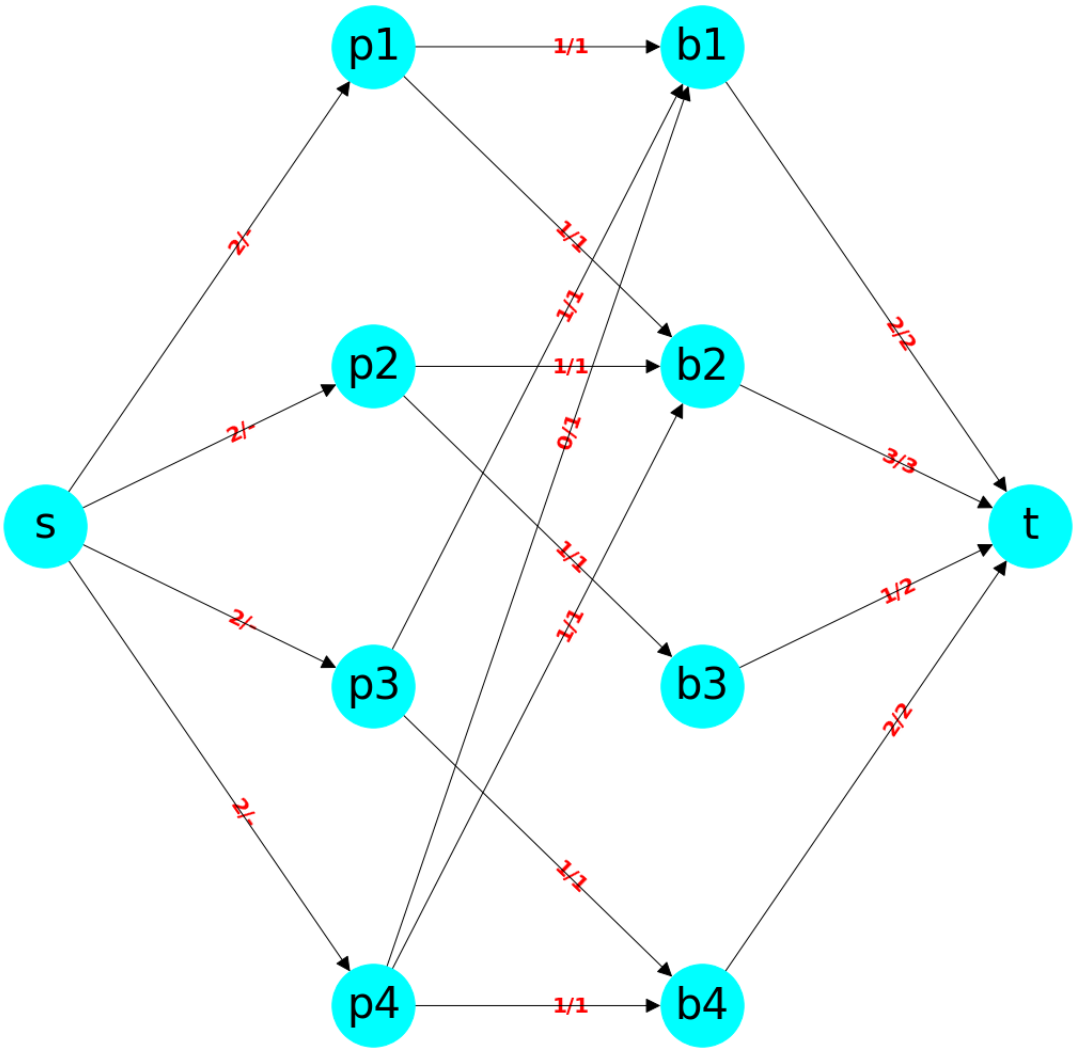

Figure 7: Flow in the network $\mathcal{G}_1$ of exercise 2.b. Each link is labeled with a label in the form *number of copies/capacity of the link*

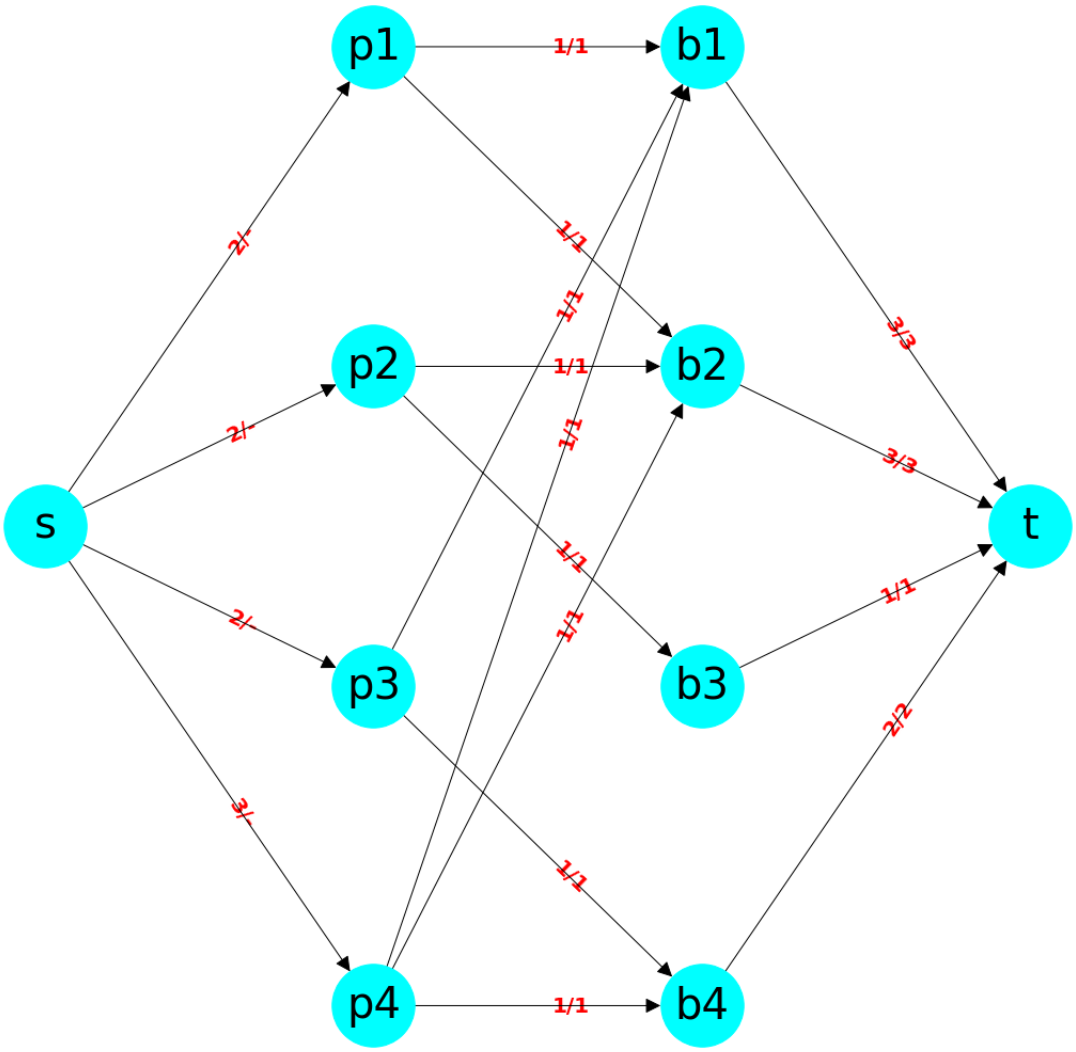

Figure 8: Flow in the network $\mathcal{G}_1$ of exercise 2.c. Each link is labeled with a label in the form *number of copies/capacity of the link*

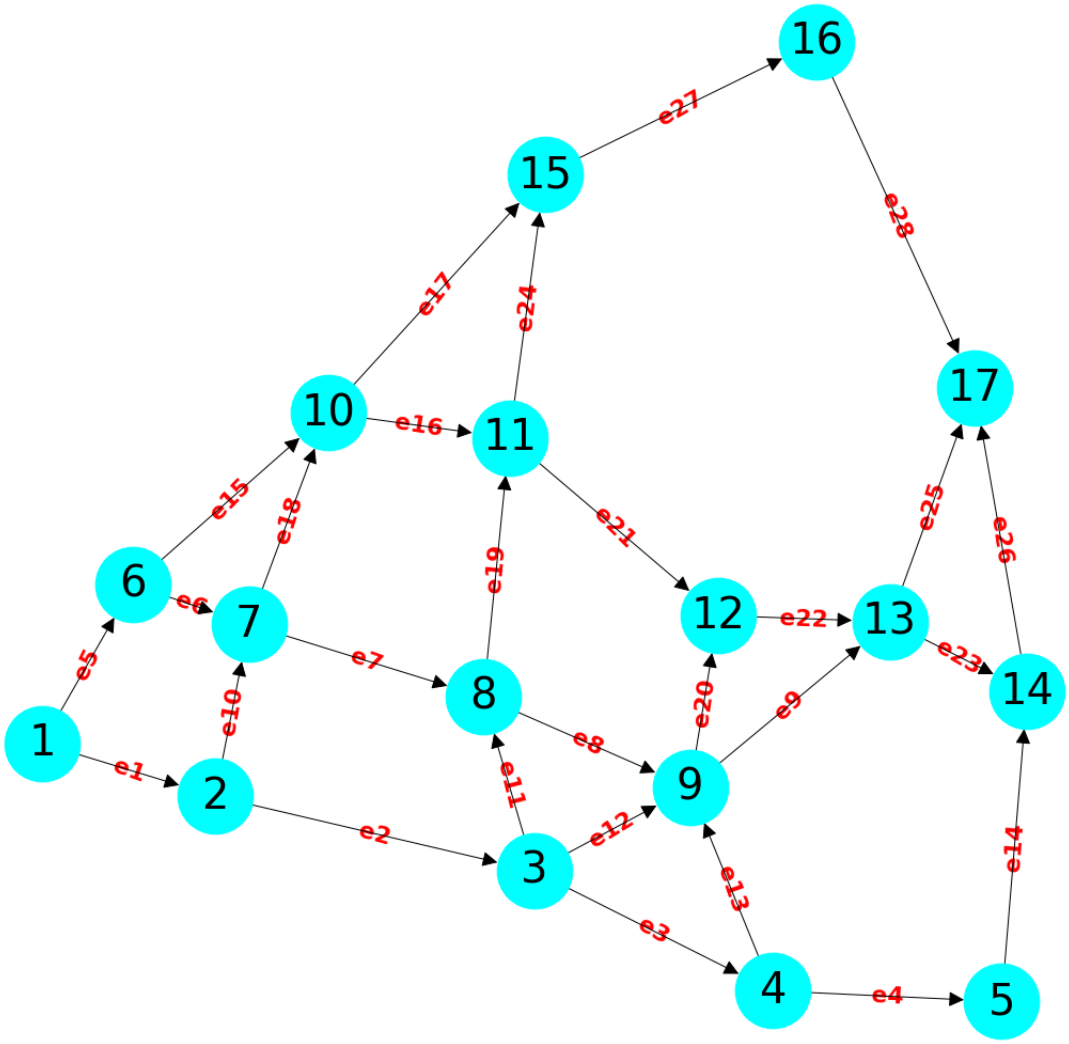

Figure 9: Network $\mathcal{G}$ of exercise 3

- $f \in \mathrm{R}^{\mathcal{E}}$ is a network flow, satisfying a positivity constraint and a flow balance equation (stated in Section 2.2) constraints, i.e.,

$$f \geq \mathbf{0}, \quad Bf = \nu. \tag{2}$$

- $B \in \mathrm{R}^{\mathcal{V} \times \mathcal{E}}$ is the node-link incidence matrix of the graph $\mathcal{G}$
- $\nu \in \mathrm{R}^{\mathcal{V}}$, $\nu = \tau(\delta^{(1)} - \delta^{(17)})$ is the exogenous network flow, $\tau = 1$ (one unit of flow enters the origin and exits the destination nodes)
- $\psi_e(f_e) = l_e f_e$ is the cost function: since the cost for sending a unit of flow on the edge e does not depend on f_e, the congestion effects are not taken into account: the network is considered as empty

The solution of the problem is the following:
Optimal value: 0.5330
$f^* = [1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$, $f^* \in \mathrm{R}^{\mathcal{E}}$
(where each position $i = 1, ..28$ in the vector refers to the corresponding link e_i)
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.
The optimal path computed is the following: $p^* = [1, 2, 3, 9, 13, 17]$ (where each element is a node), whose total travelling time is 0.5330.

3.2 Point b

The maximum flow between node 1 and 17 is the following:
$f^* = [8741, 8741, 0, 0, 13707, 4624, 4624, 4624, 6297, 0, 0, 8741, 0, 0, 9083, 825, 8258, 0, 0, 7068, 825, 7893, 3835, 0, 10355, 3835, 8258, 8258]$, $f^* \in \mathrm{R}^{\mathcal{E}}$
(where each position $i = 1, ..28$ in the vector refers to the corresponding link e_i)
Its maximum throughput is:
$\tau = 22448$

3.3 Point c

Let f' be a flow vector: $f' \in \mathrm{R}^{\mathcal{E}}$, $f' = [7524, 6537, 11139, 9282, 9282, 6398, 6728, 5988, 5951, 9557, 7423, 7423, 6814, 8536, 7652, 6537, 11924, 9640, 8161, 8603, 7974, 9446, 5562, 6719, 9455, 6686, 10833, 7403]$
(where each position $i = 1, ..28$ in the vector refers to the corresponding link e_i)
The exogenous net flow $\nu \in \mathrm{R}^{\mathcal{V}}$ satisfying the flow balance equation (stated in Section 2.2) $Bf' = \nu$ is the following:
$\nu = [16806, 8570, 19448, 4957, -746, 4768, 413, -2, -5671, 1169, -5, -7131, -380, -7412, -7810, -3430, -23544]$
(where each position $i = 1, ..17$ in the vector refers to the corresponding node i)

3.4 Points d - e

The social optimum f^* with respect to $\psi_e(f_e) = f_e \tau_e(f_e)$, with delay function $\tau_e(f_e)$:

$$\tau_e(f_e) = \frac{l_e}{1 - \frac{f_e}{c_e}}, 0 \leq f_e < c_e; \tau_e(f_e) = +\infty, f_e \geq c_e \tag{3}$$

can be computed through the following optimization problem:

$$f^* \in \underset{\substack{f \in \mathrm{R}_+^{\mathcal{E}} \\ Bf = \nu}}{\arg\min} \quad \sum_{e \in \mathcal{E}} \psi_e(f_e). \tag{4}$$

where: $\nu \in \mathrm{R}^{\mathcal{V}}$, $\nu = \tau(\delta^{(1)} - \delta^{(17)})$, $\tau = 16806$
The solution of the problem is the following:
Optimal value: 25943.62

$f^* = [6642.2, 6058.9, 3132.3, 3132.3, 10163.8, 4638.3, 3006.3, 2542.6, 3131.5, 583.3, 0, 2926.6, 0, 3132.3, 5525.5, 2854.3, 4886.4, 2215.2, 463.7, 2337.7, 3318., 5655.7, 2373.1, 0, 6414.1, 5505.4, 4886.5, 4886.5]$, $f^* \in \mathrm{R}^{\mathcal{E}}$
(where each position $i = 1, ..28$ in the vector refers to the corresponding link e_i)

The Wardrop equilibrium $f^{(0)}$ is a network flow associated with an $o - d$ path distribution z such that, if $z_\gamma > 0$ for some $o - d$ path (if someone choose γ as route from o to d), then the total travel time $T_\gamma(z)$ associated with cannot be worse than the total travel time $T_{\gamma'}(z)$ associated with any other $o - d$ path γ'.

$f^{(0)}$ is a Wardrop equilibrium if and only if it is solution of the optimization problem 5, given that the cost functions $\psi_e(f_e)$ are chosen as

$$\psi_e(f_e) = \int_0^{f_e} \tau_e(s)\ \mathrm{d}s. \tag{5}$$

The solution of 5 is

$$\psi_e(f_e) = l_e c_e \log(1 - \frac{f_e}{c_e}), f_e < c_e$$

The solution of the problem is the following:
$f^{(0)} = [6715.6, 6715.6, 2367.4, 2367.4, 10090.4, 4645.4, 2803.8, 2283.6, 3418.5, 0, 176.8, 4171.4, 0, 2367.4, 5445, 2353.2, 4933.3, 1841.6, 697.1, 3036.5, 3050.3, 6086.8, 2586.5, 0, 6918.7, 4953.9, 4933.3, 4933.3]$, $f^{(0)} \in \mathrm{R}^{\mathcal{E}}$
(where each position $i = 1, ..28$ in the vector refers to the corresponding link e_i)
The social cost under Wardrop equilibrium is $cost_w = \sum_e f_e^{(0)} \tau_e(f_e^{(0)}) = 26292.96$
It leads to a Price of Anarchy $PoA = \frac{\sum_e f_e^{(0)} \tau_e(f_e^{(0)})}{\sum_e f_e^* \tau_e(f_e^*)} = 1.01347$

3.5 Points e bis

Tolls on each link are introduced, in order to make users pay not only for their delay, but also for the cost that make other users pay due to their choice. The toll on a link e is $\omega_e = f_e^* \tau_e'(f_e^*)$, where f_e^* is the flow at the system optimum. The delay function on each link is now $\tau_e^{(\omega_e)}(f_e)$:

$$\tau_e^{(\omega_e)}(f_e) = \frac{l_e}{1 - \frac{f_e}{c_e}} + \omega_e, 0 \leq f_e < c_e; \tau_e^{(\omega_e)}(f_e) = +\infty, f_e \geq c_e \tag{6}$$

and the new Wardrop equilibrium $f^{(\omega)}$ is computed as solution of the optimization problem whose cost function $\psi_e(f_e)$ is

$$\psi_e(f_e) = \int_0^{f_e} \tau_e^{(\omega_e)}(s)\ \mathrm{d}s \tag{7}$$

The solution of 7 is

$$\psi_e(f_e) = l_e c_e \log(1 - \frac{f_e}{c_e}) + \int_0^{f_e} \omega_e\ \mathrm{d}s = l_e c_e \log(1 - \frac{f_e}{c_e}) + \omega_e f_e, f_e < c_e$$

with $\omega_e = f_e^* \tau_e'(f_e^*) = f_e^* \frac{l_e c_e}{(c_e - f_e^*)^2}$

The solution of the problem is the following:
$f^{(\omega)} = [6643, 6059.1, 3132.5, 3132.5, 10163, 4638.3, 3006.3, 2542.3, 3131.5, 583.9, 0, 2926.6, 0, 3132.5, 5524.8, 2854.2, 4886.4, 2215.8, 464, 2337.5, 3318.2, 5655.7, 2373, 0, 6414.1, 5505.5, 4886.4, 4886.4]$, $f^{(\omega)} \in \mathrm{R}^{\mathcal{E}}$
(where each position $i = 1, ..28$ in the vector refers to the corresponding link e_i)
The social cost under Wardrop equilibrium is $cost_{wtoll} = \sum_e f_e^{(\omega)} \tau_e(f_e^{(\omega)}) = 25943.62$
It leads to a Price of Anarchy $PoA_{toll} = \frac{\sum_e f_e^{(\omega)} \tau_e(f_e^{(\omega)})}{\sum_e f_e^* \tau_e(f_e^*)} = 1$
It can be noticed that:

- The wardrop equilibrium $f^{(\omega)}$ is the same as the social optimum f^*, with a precision of 0.7761
- The cost related to the wardrop equilibrium $f^{(\omega)}$ is the same as the optimal cost, with a precision of 0.00018
- The Price of Anarchy related to the Wardrop equilibrium $f^{(\omega)}$ is 1, with a precision of 10^-7

3.6 Point f

Let the delay function be $\tau_e(f_e)$:

$$\tau_e(f_e) = \frac{l_e}{1 - \frac{f_e}{c_e}} - l_e, 0 \leq f_e < c_e; \tau_e(f_e) = +\infty, f_e \geq c_e \tag{8}$$

with corresponding cost function $\psi_e(f_e) = f_e\tau_e(f_e)$ representing the total additional delay compared to the total delay in a free flow.

The social optimum f^* is:
$f^* = [6653.3, 5774.7, 3419.7, 3419.7, 10152.7, 4642.8, 3105.8, 2662.2,$
$3009.1, 878.6, 0, 2354.9, 0, 3419.7, 5509.9, 3043.7, 4881.8, 2415.6,$
$443.7, 2008., 3487.4, 5495.4, 2203.8, 0., 6300.7, 5623.5, 4881.8, 4881.8]$, $f^* \in \mathrm{R}^{\mathcal{E}}$
(where each position $i = 1, ..28$ in the vector refers to the corresponding link e_i)
with
Optimal value: 15095.51

The Wardrop equilibrium $f^{(0)}$ is computed as solution of the optimization problem whose cost function $\psi_e(f_e)$ is

$$\psi_e(f_e) = \int_0^{f_e} \tau_e(s)\ \mathrm{d}s \tag{9}$$

The solution of 9 is

$$\psi_e(f_e) = l_e c_e \log(1 - \frac{f_e}{c_e}) - \int_0^{f_e} l_e\ \mathrm{d}s = l_e c_e \log(1 - \frac{f_e}{c_e}) - l_e f_e, f_e < c_e$$

The Wardrop equilibrium $f^{(0)}$ is:
$f^{(0)} = [6781.7, 6004.9, 3266.3, 3266.3, 10024.3, 4665.6, 3028.9, 2641,$
$3038.8, 776.8, 240.3, 2498.3, 0, 3266.3, 5358.7, 2881.5, 4890.7, 2413.5,$
$628.2, 2100.6, 3509.7, 5610.2, 2262.6, 0, 6386.5, 5528.8, 4890.7, 4890.7]$, $f^{(0)} \in \mathrm{R}^{\mathcal{E}}$
(where each position $i = 1, ..28$ in the vector refers to the corresponding link e_i)
with
social cost under Wardrop equilibrium $cost_w = \sum_e f_e^{(0)}\tau_e(f_e^{(0)}) = 15148.09$
and
Price of Anarchy $PoA = \frac{\sum_e f_e^{(0)}\tau_e(f_e^{(0)})}{\sum_e f_e^*\tau_e(f_e^*)} = 1.00348$

To design tolls $w^* \in \mathrm{R}^{\mathcal{E}}$ such that the Wardrop equilibrium $f^{(\omega^*)}$ coincides with the social optimum f^*, the following *theorem* can be used: an optimal toll configuration such that the corresponding Wardrop equilibrium $f^{(\omega^*)}$ coincides with the social optimum f^* is $\omega_e^* = \psi_e'(f_e^*) - \tau_e(f_e^*)$ for all links e; if $\psi_e(f_e) = f_e\tau_e(f_e)$, $\omega_e^* = f_e^*\tau_e'(f_e^*)$. Such tolls are called marginal pricing tolls.

The delay function on each link is now $\tau_e^{(\omega_e^*)}(f_e)$:

$$\tau_e^{(\omega_e^*)}(f_e) = \frac{l_e}{1 - \frac{f_e}{c_e}} - l_e + \omega_e^*, 0 \leq f_e < c_e; \tau_e^{(\omega_e^*)}(f_e) = +\infty, f_e \geq c_e \tag{10}$$

and the new Wardrop equilibrium $f^{(\omega^*)}$ is computed as solution of the optimization problem whose cost function $\psi_e(f_e)$ is

$$\psi_e(f_e) = \int_0^{f_e} \tau_e^{(\omega_e^*)}(s)\ \mathrm{d}s \tag{11}$$

The solution of 11 is the same of 9
since $\omega_e^* = \omega_e$ ($l'_e = 0$).

The Wardrop equilibrium $f^{(\omega^*)}$ is:
$f^{(\omega^*)} = [6653.1, 5775.4, 3419.5, 3419.5, 10152.9, 4642.4, 3105.5, 2661.7,$
$3009.2, 877.7, 0, 2355.9, 0, 3419.5, 5510.4, 3043.4, 4881.7, 2414.6, 443.8,$
$2008.5, 3487.1, 5495.6, 2204.1, 0, 6300.7, 5623.5, 4881.7, 4881.7]$, $f^{(\omega^*)} \in \mathrm{R}^{\mathcal{E}}$
(where each position $i = 1, ..28$ in the vector refers to the corresponding link e_i)
with
social cost under Wardrop equilibrium $cost_{w^*} = \sum_e f_e^{(\omega^*)} \tau_e(f_e^{(\omega^*)}) = 15095.51$
and
Price of Anarchy $PoA = \frac{\sum_e f_e^{(\omega^*)} \tau_e(f_e^{(\omega^*)})}{\sum_e f_e^* \tau_e(f_e^*)} = 1$

As expected:

- The Wardrop equilibrium $f^{(\omega^*)}$ is the same as the social optimum f^*, with a precision of 0.9325
- The cost related to the Wardrop equilibrium $f^{(\omega^*)}$ is the same as the optimal cost, with a precision of 0.00065
- The Price of Anarchy related to the Wardrop equilibrium $f^{(\omega^*)}$ is 1, with a precision of 10^-7