Network Dynamics - Homework 1

December 11, 2022

Abstract

In this report we present our solution to Homework 1. This report is the result of active collaboration between: Lorenzo Bergadano (s304415), Francesco Capuano (s295366), Matteo Matteotti (s294552), Enrico Porcelli (s296649) and Paolo Rizzo (s301961). For the sake of full reproducibility, we published our code on Github at: https://github.com/fracapuano/NetworkDynamics.

1 Exercise 1

Consider the network in Figure (1) with link capacities:

$$c_i = \begin{cases} 1 & \text{if } i \text{ is even} \\ 2 & \text{if } i \text{ is odd} \end{cases} \quad \forall i \in [1, 6]$$

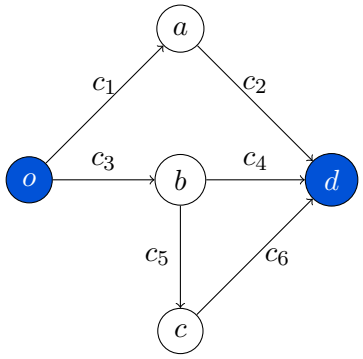


Figure 1: Exercise 1 graph

Question 1 What is the minimum aggregate capacity that needs to be removed for no feasible flow from o to d to exist?

Question 2 What is the maximum aggregate capacity that can be removed from the links without affecting the maximum throughput from o to d?

Question 3 You are given $x > 0$ extra units of capacity. How should you distribute them in order to maximize the throughput that can be sent from o to d? Plot the maximum throughput from o to d as a function of $x \geq 0$.

1.1 Question 1

Consider a capacited multigraph $\mathcal{G}(\mathcal{V}, \mathcal{E}, c)$.

Let $B \in \{-1, 0, +1\}^{|\mathcal{V}| \times |\mathcal{E}|}$ indicate the node-link incidence matrix, $f \in \mathbb{R}_+^{|\mathcal{E}|}$ the flow associated to each single edge and $\nu \in \mathbb{R}^{|\mathcal{V}|}$ the exogeneous flow, i.e. the vector such that:

$$\nu(\delta^{(o)} - \delta^{(d)}) = \tau \tag{1}$$

Where τ represents the total throughput of the network $\mathcal{G}$. The usual flow-maximization problem is expressed as:

$$\begin{aligned} \max_{f} \quad & \tau \\ \text{s.t.} \quad & Bf = \nu \quad , 0 \leq f \leq c \\ & \tau \geq 0 \end{aligned} \tag{2}$$

In this context, a flow is said to be feasible if it respects two constraints:

1. Conservation of mass ($Bf = \nu$).
2. Non negativity and feasibility capacity-wise ($0 \leq f \leq c$)

Consider now the definition of minimal-capacity cut. Given the set $\mathcal{V}$ of nodes of $\mathcal{G}$, among all the possible $2^{|\mathcal{V}|-2}$ o-d cuts $\mathcal{U}_i$, the minimal-capacity one is defined as:

$$\mathcal{U}^* : c_{\mathcal{U}^*} \leq c_{\mathcal{U}_i}, \quad \forall i = 1, \ldots, 2^{|\mathcal{V}|-2} \tag{3}$$

Since from the Max-Flow-Min-Cut theorem, the capacity of $\mathcal{U}^*$ corresponds also to the throughput corresponding to the optimal flow, if one subtracts exactly $c_{\mathcal{U}^*}$ units of aggregate capacity from the links from $\mathcal{U}^*$ to $\mathcal{V} \setminus \mathcal{U}^*$, then the throughput τ clearly becomes null (as the min-cut capacity becomes 0). This would lead to a situation in which no feasible flow exists from o to d, since the links that were crossing $\mathcal{U}^*$ would now have null capacity. Furthermore, if the aggregate capacity of links crossing the min-cut is null, by the very definition of o-d cuts, node d cannot be reached by o since the only feasible flow would be $f = 0$.

Consequently, the minimal total aggregate capacity to be removed from the network to disconnect d from o equals the capacity of the minimal-cut. For the network presented in Figure (1), this value corresponds to 3.

In Figure (2), this minimal-cut is represented coloring the nodes belonging to it.

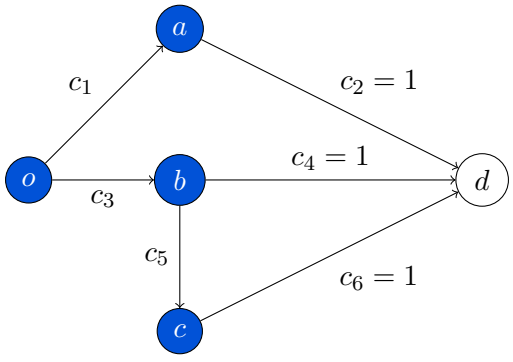


Figure 2: Minimal-Capacity Cut of Exercise 1 Graph

1.2 Question 2

From the Max-Flow Min-Cut Theorem, the throughput of a network equals the capacity of the minimal cut. This fundamental result displays to the fullest the close relationship entangling capacity and throughput. Nevertheless, modifications on capacity do not necessarily impact the throughput. Indeed, those modifications that do not change the value of the min-cut capacity do not have any impact at all on the throughput.

Figure (3) shows the flow vector f^* (in deepblue) for the given $\mathcal{G}$, related to each edge's capacity (in black). Each edge is represented with a dashed line if flow completely saturates its capacity and with a thick deepblue line when the flow on that edge is below the capacity limit.

The maximal throughput for this graph is still $\tau^* = 3$.

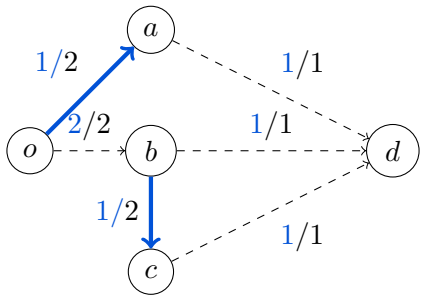


Figure 3: f^* with capacity $c = (2, 1, 2, 1, 2, 1)$

Let us define the *remaining capacity* as the residual capacity that is not used since the flow does not fully saturate the corresponding edge. In particular:

$$c_{\text{left}} \coloneqq \|c - f\|_1 \tag{4}$$

In this case, $c_{\text{left}} = 2$. If one were to remove any value of aggregate capacity larger than this value, the throughput would necessarily be negatively influenced by this choice, as this modification would inevitably impact the capacity of the minimal cut or the capacity of non-saturated edges (or both).

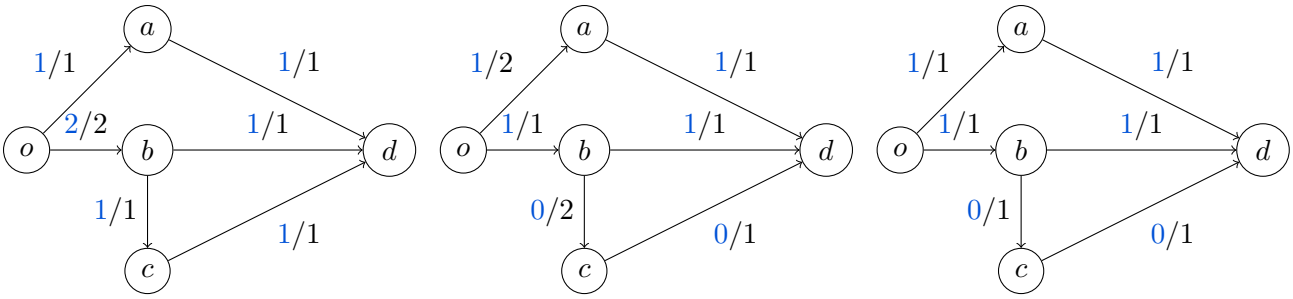
On the other hand, removing an aggregate capacity strictly smaller than c_{left} value would not necessarily result in a decrease of throughput. Since the minimal capacity cut is $\mathcal{U}^* = \{o, a, b, c\}$, e_2, e_4 and e_6's capacity is what influence the throughput. Removing any capacity $\varepsilon > 0$ from these edges would result in an ε-reduction of the throughput too, by the aforementioned Max-Flow Min-Cut Theorem.

However, removing residual capacity from non-saturated links *between* the nodes in the cut does not impact the throughput, as it does not necessarily modify the minimal cut capacity value. The upper-bound to the capacity that can be removed from non-saturated links while not reducing the throughput is clearly given by c_{left}.

Therefore, we conclude that the maximal aggregate capacity that can be removed from the graph without affecting the throughput is c_{left}.

Figure (4) shows the effects that removing capacity c_{left} can have.

In particular, Figure (4a) shows that when one removes capacity from edges e_1 and e_3 exactly c_{left} units of capacity, then the throughput is not affected at all, as the minimal cut value remains the same. On the contrary, Figure (4b) shows that removing an aggregate capacity smaller than c_{left} on saturated-edges negatively affects the throughput.



(a) Optimal f^* ($\tau^* = 3$) with capacity vector $c = (1, 1, 2, 1, 1, 1)$

(b) Optimal f^* ($\tau^* = 2$) with capacity vector $c = (2, 1, 1, 1, 2, 1)$

(c) Optimal f^* ($\tau^* = 2$) with capacity vector $c = (1, 1, 1, 1, 1, 1)$

Figure 4: Optimal flows f^* for different capacity vectors

Finally, Figure (4c) shows that any removal of larger amount of capacities would inevitably result in a deepblueuction of the throughput.

1.3 Question 3

The problem of allocating $x > 0$ extra-capacity to obtain an increase in the throughput can be split in the two sub-problems:

- *Where to allocate this extra capacity* on the network
- *How much extra capacity to allocate*, i.e. the value of x

To better highlight the relevance of the former, let us define for a given node $u \in \mathcal{V}$ in a multi-graph $\mathcal{G}$ the *out-star* and *in-star* respectively as:

$$\mathcal{S}(u) \coloneqq \{e \in \mathcal{E} : \theta(e) = \mathcal{V}_0\} \quad \mathcal{S}^-(u) \coloneqq \{e \in \mathcal{E} : \kappa(e) = \mathcal{V}_0\} \tag{5}$$

Clearly enough, an increase in the capacity of the edges in $\mathcal{S}^-(d)$ may be sufficient to trigger an increase in the throughput, albeit not necessary. For instance, taking as a reference the situation presented in Figure (3), it is clear that increasing the capacity of $e_2 \in \mathcal{S}^-(d)$ clearly triggers an increase of throughput (as well as the saturation of e_1). However, an increase in the capacity of $e_4 \in \mathcal{S}^-(d)$ does not turn into an increase of throughput, as e_3 is already saturated and any extra unit of flow that would use e_4 (triggering an increase of τ) would necessarily stop flowing in e_5 (and, consequently, e_6), thus actually reducing the throughput.

More generally, this line of thought can be re-considered in light of the possible characteristics of feasible flows. With respect to the flow vector f it is indeed possible to partition the set of edges $\mathcal{E}$ into two subsets, namely:

- $\mathcal{E}_s = \{e \in \mathcal{E} | c_e = f_e\}$, i.e. the subset of edges which are fully saturated.
- $\tilde{\mathcal{E}} = \{e \in \mathcal{E} | c_e > f_e\}$, i.e. the subset of edges which are not fully saturated.

It is easy to see that any increase in the capacity of these edges that are in $\tilde{\mathcal{E}}$ does not play out in an increase of τ, since not even the already available capacity is fully exploited.

On the other hand, unitary increments of capacity of edges in $\mathcal{E}_s$ might trigger an increase of τ.

Interestingly, the actual increments in the throughput that are induced by increase in capacity can be obtained considering the Lagrangian multipliers associated to the capacity-feasibility constraints.

In particular, in light of the *shadow price*-interpretation of the Lagrangian Multipliers, it follows that a marginal relaxation of the capacity-constraint associated to $e_i \in \mathcal{E}$ results in an increase of throughput equal to:

$$\frac{\partial \tau}{\partial c_{e_i}} = \lambda(c_{e_i}) \tag{6}$$

Where λ_{e_i} represents the Lagrangian multiplier associated to the edge e_i.

From eq. 6 and the fact that non-active constraints relate to null multipliers, it immediately follows that, if one considers infinitesimal increments of capacity x on single edges in $\mathcal{E}_s$, then the aggregate increment in throughput $\Delta\tau$ comes from:

$$\frac{\partial \tau}{\partial c_{e_i}} = \begin{cases} \lambda_{e_i} & \forall e_i \in \mathcal{E}_s \\ 0 & \forall e_i \in \tilde{\mathcal{E}} \end{cases} \implies \forall e_i \in \mathcal{E}_s \quad \Delta\tau(x) = \int_0^x \lambda(c_{e_i}(s))\ ds \tag{7}$$

Once we obtained a general expression for the increments of τ based on multipliers, we focused on solving the second sub-problem aforementioned.

If one considers a multipliers whose value does not vary with capacity in some interval $I_i = [a, b] \subseteq \mathbb{R}^+$ (i.e. $\lambda(c_{e_i}) = \lambda_i\ \forall c_{e_i} \in I$), then from equation 7 it follows that:

$$\Delta\tau(x) = \lambda_i \cdot x \quad \forall x \text{ s.t. } c_{e_i} + x \in I \tag{8}$$

We found this I to be upper-bounded by the capacity value such that the number of minimal cuts is equal to two. This implies that one can obtain the largest possible increment in τ consequent to a relaxation of $c_{e_i} \in I$ adding an amount of extra capacity equal to $x = b - U_{C^*}$, where U_{C^*} is the capacity of the current minimal cut and b is the value of capacity needed to have two minimal-cuts. In particular, in this case we found that $x = 1$.

Quantifying the capacity increase as just mentioned directly follows from the fact that the throughput is, by definition, equal to the capacity of the minimal cut.

Adding an amount of extra capacity $\tilde{x}$ to an edge e_i regardless of the impact of such update on the second minimal-cut indeed does not necessarily play out in an increase of throughput equal to $\lambda_i \cdot \tilde{x}$, as the second minimal-cut poses an upper bound to the throughput increment.

In light of these two proved results, we developed a custom algorithm that would take care of allocating extra units of capacity *on one edge at a time*. This algorithm is presented in Algorithm 1.

Algorithm 1 Allocate Extra Capacity On Single Edges

Require: U_C, b

repeat

$e_i^* \in \mathcal{E} \leftarrow \arg\max_{e \in \mathcal{E}} \lambda(c)$ ▷ Edge to allocate capacity on

$x \leftarrow b - U_C$

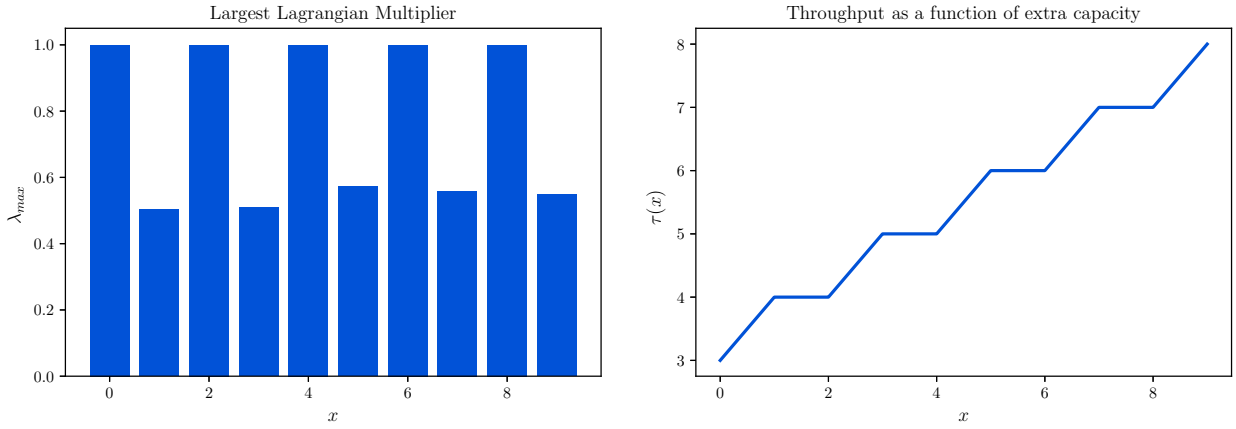
$\tau_{old} \leftarrow \tau(c)$

$c_{e_i^*} \leftarrow c_{e_i^*} + x$

$\tau_{new} \leftarrow \tau(c)$

$\Delta\tau = \tau_{new} - \tau_{old}$

$U_C \leftarrow U_C(c), b \leftarrow b(c),$ ▷ Update U_C and b in light of new c



(a) Leading Multiplier as capacity is increased

(b) Throughput as a function of capacity

Figure 5: Results for capacity-increments on single edges

With this approach, we obtained increments of throughput as the ones presented in Figure (5b). Furthermore, Figure (5a) shows the leading multiplier for various level of capacity. It is certainly interesting to note that, for instance, for $x \in [0, 1]$ the throughput grows as:

$$\underbrace{\lambda_{\max}}_{=1} \cdot x = x \tag{9}$$

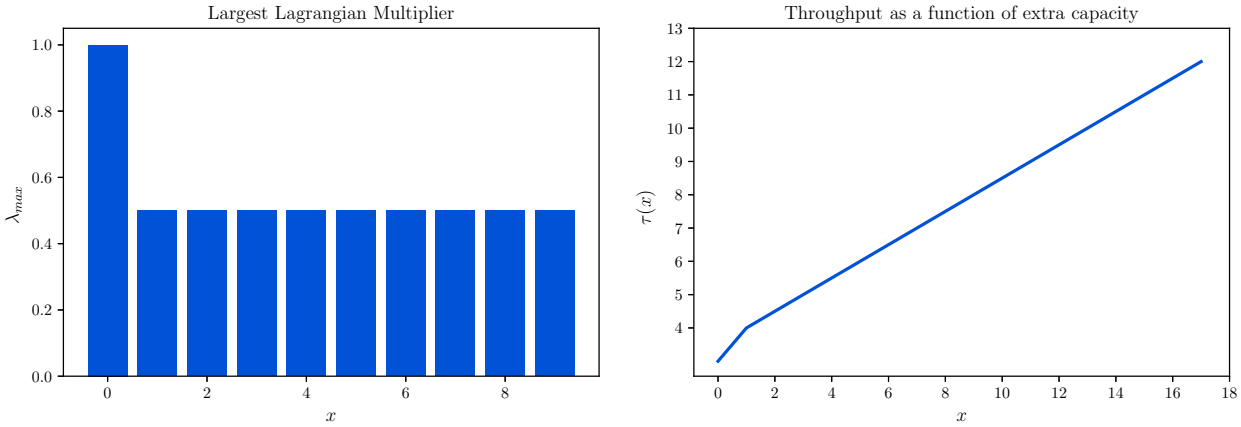
We used the same approach presented in Algorithm 1 (with just a simple tweak) to take care of the possibility of re-partitioning the extra-capacity among different edges. The tweak consisted in choosing the edge(s) where to perform the increment e_i^* according to the following condition:

$$e_i^* = \begin{cases} \arg\max_{e \in \mathcal{E}} \lambda(c) & \exists j \text{ s.t. } \lambda_j = 1 \\ e_i, e_j \text{ s.t. } \lambda_i = \lambda_j = \frac{1}{2} & \text{otherwise} \end{cases} \tag{10}$$

The results of said approach are presented in Figure (6b).

Interestingly, it is possible to see that the slope of $\tau(x)$ is always matched by the corresponding multiplier for the given level of capacity, according to eq. 8. Moreover, as the lagrangian multiplier stays constantly equal to $\frac{1}{2}$ for every value of capacity increment larger that 1, the slope of the throughput function has a slope equal to the multiplier.

As a final remark, we would like to highlight how our alternative approach matches a more algorithmic resolution of the task based on increasing the capacity of the most-common edge among all minimal cuts. Furthermore, we want to conclude highlighting that the results in 6 coincide with the ones presented in: to an increase of capacity equal to 2 units, corresponds indeed a unitary increase in the throughput, according to the actual value of the corresponding multiplier.



(a) Leading Multiplier as capacity is increased

(b) Throughput as a function of capacity

Figure 6: Results for capacity-increments on multiple edges

2 Exercise 2

There are a set of people $\{p1, p2, p3, p4\}$ and a set of books $\{b1, b2, b3, b4\}$. Each person is interested in a subset of books, specifically

$$p1 \to \{b1, b2\}, \quad p2 \to \{b2, b3\}, \quad p3 \to \{b1, b4\}, \quad p4 \to \{b1, b2, b4\}.$$

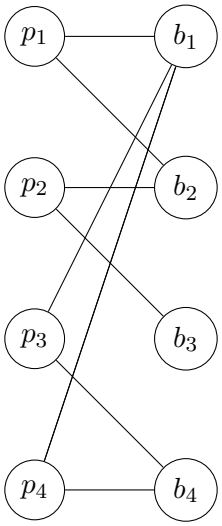


Figure 7: Graph representation

Question 1 Exploit max-flow problems to find a perfect matching (if any).

Question 2 Assume now that there are multiple copies books, and the distribution of the number of copies is $(2, 3, 2, 2)$. Each person can take an arbitrary number of different books. Exploit the analogy with max-flow problems to establish how many books of interest can be assigned in total.

Question 3 Suppose that the library can sell a copy of a book and buy a copy of another book. Which books should be sold and bought to maximize the number of assigned books?

2.1 Question 1

To check whether a perfect matching exists, Hall's theorem hypotheses must be verified. Hall's theorem states that given a bipartite graph $\mathcal{G} = (\mathcal{V}_0 \cup \mathcal{V}_1, \mathcal{E}, W)$, a perfect matching exists if and only if $|\mathcal{N}_\mathcal{A}| \geq |\mathcal{A}|$ where $\mathcal{A}$ is any subset of the set of nodes $\mathcal{V}_0$ and $\mathcal{N}_\mathcal{A}$ is the set of its neighbours.

Table 1 shows that every possible subset of $\mathcal{V}_0$ is met with a set of neighbour nodes whose cardinality is greater or equal than the subset's one. For this reason, Hall's condition is met and a perfect matching does indeed exist.

$\mathcal{A}$	$\lvert\mathcal{N}_\mathcal{A}\rvert$
$\{p1\}$	2
$\{p2\}$	2
$\{p3\}$	2
$\{p4\}$	3
$\{p1, p2\}$	3
$\{p1, p3\}$	3
$\{p1, p4\}$	3
$\{p2, p3\}$	4
$\{p2, p4\}$	4
$\{p3, p4\}$	3
$\{p1, p2, p3\}$	4
$\{p1, p2, p4\}$	4
$\{p2, p3, p4\}$	4
$\{p1, p2, p3, p4\}$	4

Table 1: Table to check Hall's condition

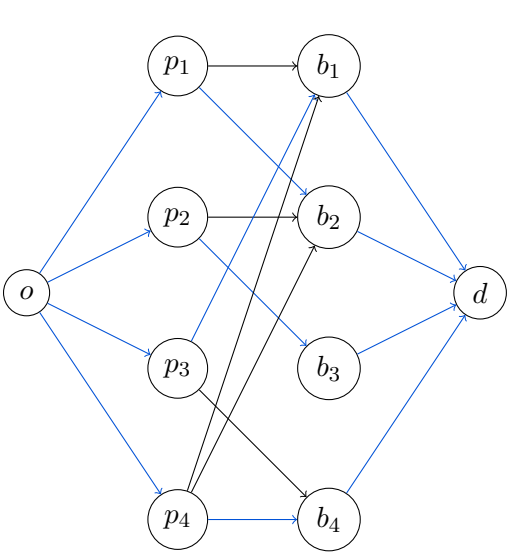
To find said matching we can expand the network into an auxiliary graph $\mathcal{G}'$ and reduce the problem to a maximum flow one.

The first step to build $\mathcal{G}'$ is adding a source and a sink node to the original graph, together with one edge from the source to each of the nodes in $\mathcal{V}_0$ and from each of the nodes in $\mathcal{V}_1$ to the sink. All of these new links will have the same capacity equal to 1.

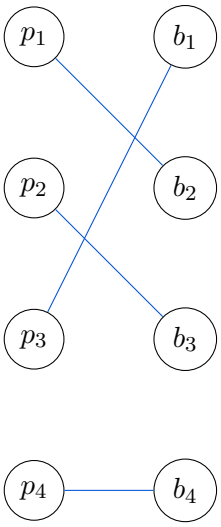
Because of the topology of the graph, every node involved in any flow vector from the source to the sink has incoming and outcoming flow equal to 1. For this reason, since there are 4 nodes in V_0 the the max-flow is at most 4. Moreover, knowing that at least one perfect matching exists, we can affirm that the max-flow will be exactly equal to 4.

Given this result we know that any max-flow vector will use a set of edges between nodes in V_0 and V_1. Said set will correspond exactly to a perfect matching.

Our results are shown in 8 and are, as expected, a optimal flow vector $f_{\mathcal{G}'}$ yielding a maximal throughput of 4 in $\mathcal{G}'$ and its correspondent perfect matching in $\mathcal{G}$. As previously mentioned, this matching is clearly a subset of the support of $f_{\mathcal{G}'}$.



(a) Max-flow in the auxiliary graph $\mathcal{G}'$



(b) Correspondent perfect matching representation in the original graph $\mathcal{G}$

Figure 8: Question 1 results

2.2 Question 2

Similarly to Question 1, the problem can be reduced to a Max-Flow problem. The only step that needs to be taken is understanding how the different book availabilities affect the capacity of the auxiliary graph.

In particular, the indication on the number of copies available translates into a change in the capacity of the edges between the nodes in $\mathcal{V}_1$ and d, which will now be equal to $(2, 3, 2, 2)$.

The fact that each person can take an arbitrary number of different books means that the capacities of the edges between $\mathcal{V}_0$ and $\mathcal{V}_1$ remain equal to 1 (since each person can only take one copy of each different book) but each person should be able to take all four different books, if they are interested. For this reason, the weights of the links between o and the nodes in $\mathcal{V}_0$ will be set equal to plus infinity.

The solution to the Max-Flow problem is a flow vector yielding a throughput equal to 8, as shown in Figure (9).

2.3 Question 3

In mathematical terms, the whole question translates into a constrained maximization problem, where the objective function is the maximum throughput of the network and the constraints are given by the conservation of mass and by capacity-feasibility.

Thanks to this reformulation of the problem, we answered this question using previously-mentioned results on the Lagrangian multipliers λ related to the Max-Flow problem on the graph presented in Figure (9).

Therefore, by looking at the four edges from b_1, b_2, b_3 and b_4 to d, one should decide to sell a copy of the book that is the head of the edge with the smallest multiplier and buy a copy of the book that is the head of the edge with the highest multiplier, i.e., respectively decrease and increase the capacity of said edges by one. In other words, the book that should be sold and bought is, respectively:

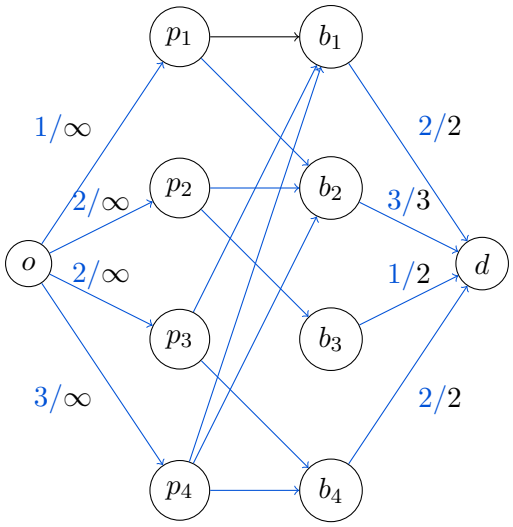


Figure 9: Max-flow in the auxiliary graph (all colored edges without indication of flow/capacity are to be considered fully saturated and with capacity 1). $f_{e_{p_1,b_1}} = 0$

Edge	$b1 \to d$	$b2 \to d$	$b3 \to d$	$b4 \to d$
Multiplier	1	0.57	$\simeq 0$	0.58

Table 2: Lagrange multipliers of the four edges directed towards node d

$$b_{\text{sold}} = \theta(\arg\min_{e\in\mathcal{E}} \lambda_e), \quad b_{\text{bought}} = \theta(\arg\max_{e\in\mathcal{E}} \lambda_e) \tag{11}$$

In principle, one should first decrease the capacity of the edge between b_{sold} and d and then, subsequently, obtain the tail of the edge corresponding to the larger multiplier, as multipliers might change when the throughput changes.

In our experiments we observed that the minimal multiplier was equal to zero (as per the fact that the capacity-constraint associated to $e_{b_3,d}$ is non-active). We believe this played a major role in the fact that the largest multiplier value did not change when capacity updates were performed.

As Table 2 shows, the optimal solution is selling a copy of book 3 to buy a copy of book 1. This will result in an unitary increment of the maximum throughput for the network considered.

3 Exercise 3

We are given the highway network in Los Angeles, which is depicted as a graph network in 10. Each node represents an intersection between two different highways and for each link $e_i \in \{e_1, ..., e_{28}\}$ the corresponding capacity c_e is given, as well as the minimum traveling time l_e, *i.e.*, the traveling time when that road is empty. Analogously, for each link, the delay function $\tau_e(f_e)$ can be computed as:

$$\tau_e(f_e) = \begin{cases} \frac{l_e}{1-f_e/c_e} & 0 \le f_e \le c_e \\ +\infty & f_e > c_e \end{cases} \tag{12}$$

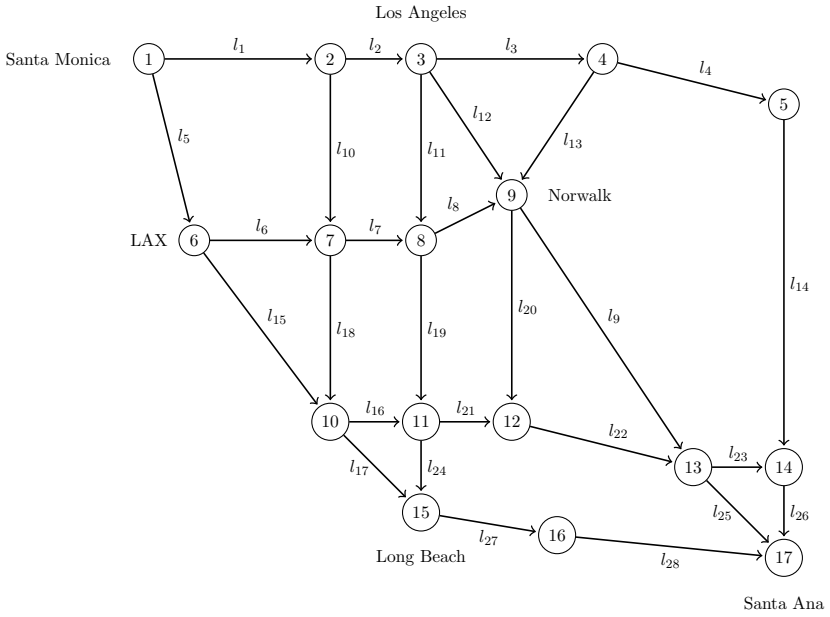


Figure 10: Approximation of a portion of the highway network in the State of California.

Question 1 Find the shortest path between node 1 and 17. This is equivalent to the fastest path (path with shortest traveling time) in an empty network.

Question 2 Find the maximum flow between node 1 and 17.

Question 3 Given the flow vector, compute the external inflow ν satisfying $Bf = \nu$.

Question 4 Find the social optimum f^* with respect to the delay on the different links $\tau_e(f_e)$. For this, minimize the cost function:

$$\sum_{e \in \mathcal{E}} \left(\frac{l_e c_e}{1 - f_e/c_e} - l_e c_e \right) \tag{13}$$

subject to the flow constraints.

Question 5 Find the Wardrop equilibrium $f^{(0)}$ For this, use the cost function:

$$\sum_{e \in \mathcal{E}} \int_0^{f_e} \tau_e(s) ds \tag{14}$$

Question 6 Introduce tolls, such that the toll on link e is $\omega_e = f_e^* \tau'(f_e^*)$, where $f^*(e)$ is the flow at the system optimum. Now the delay on link e is given by $\tau_e(f_e) + \omega_e$. Compute the new Wardrop equilibrium $f^{(\omega)}$. What do you observe?

Question 7 Instead of the total travel time, let the cost for the system be the total additional delay compared to the total delay in free flow, given by:

$$\psi_e(f_e) = f_e \left(\tau_e(f_e) - l_e \right) \tag{15}$$

subject to the flow constraint. Compute the system optimum f^* for the costs above. Construct tolls ω^* such that the Wardrop equilibrium $f^{(\omega^*)}$ coincides with f^*. Compute the new Wardrop equilibrium with the constructed tolls $f^{(\omega^*)}$ to verify your results.

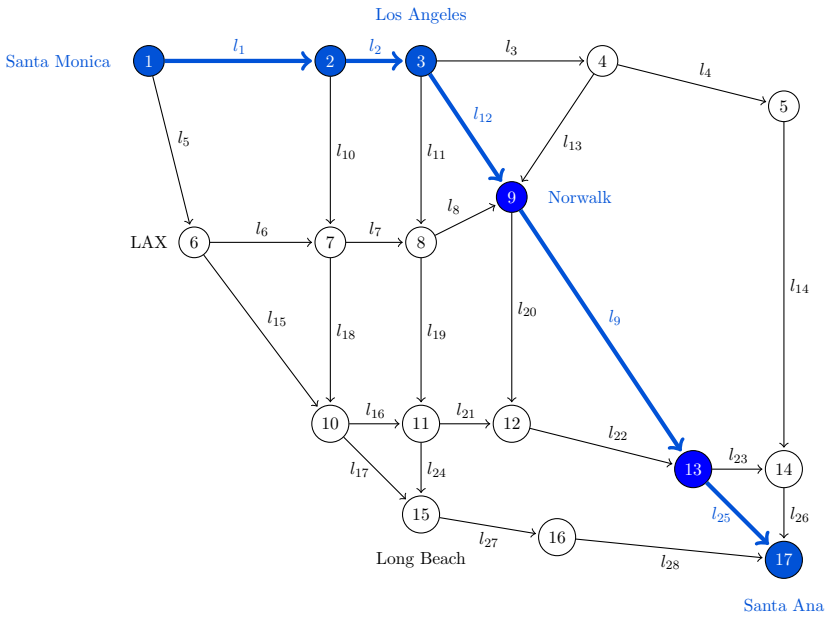


Figure 11: Shortest path connecting Santa Monica to Santa Ana.

3.1 Question 1

In principle, to solve this problem one could reinterpret it as a Network Flow optimization problem, and in particular:

$$\begin{aligned} \min_{f} \quad & l^T f \\ \text{s.t.} \quad & Bf = \nu \\ & f \geq 0 \end{aligned} \tag{16}$$

where the exogenous flow vector $\nu = (\tau, 0, \ldots, 0, -\tau)$ and where the network is assumed to be free from congestion. Once the optimal flow f^* is retrieved, the shortest path is the one whose edges are characterised by a positive optimal flow.

Despite being a Convex Optimization problem that could be solved using standard algorithms, NETWORKX comes with a built-in SHORTEST_PATH function that is more efficient than solving a convex flow-optimization problem.

Specifically, the shortest path is shown in Figure (11).

One could also see that this solution also minimizes the number of edges along the path, even though this is not necessarily true in principle since each edge has a different length.

3.2 Question 2

To solve the problem, existing algorithms such as *Ford-Fulkerson* can be used. However, NETWORKX provides us with yet another off-the-shelf function to solve the problem, called MAX_FLOW. In particular, the maximum flow between the two nodes amounts to 22448.

3.3 Question 3

The external flow can be found as:

$$\nu = Bf \tag{17}$$

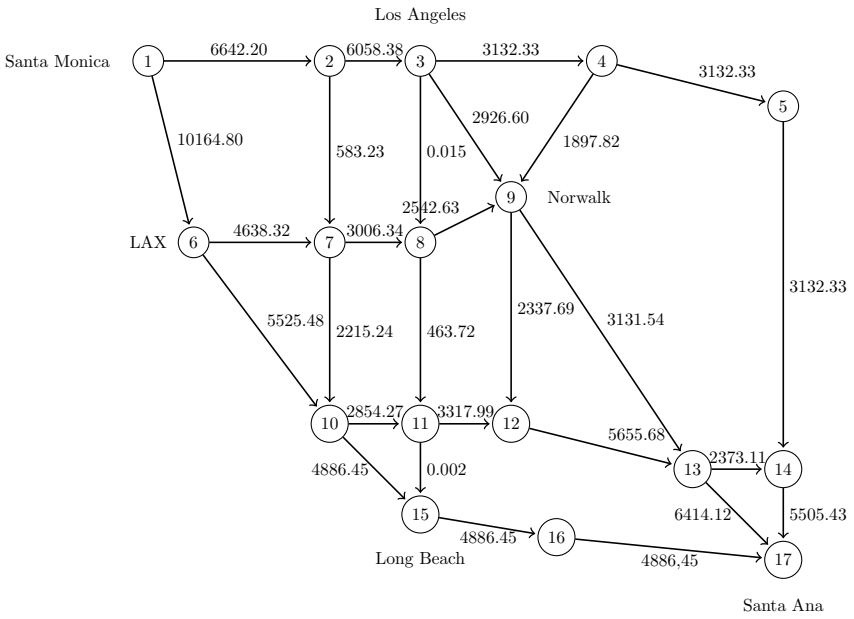


Figure 12: System optimum, solution of the SO-TAP when $\tau_e = \frac{l_e}{1-f_e/c_e}$

Solving the system, one can find the solution. The numerical solution, as a vector in $\mathbb{R}^{17}$ can be found in the code that comes with this report.

3.4 Question 4

Analogously to the first question, this problem belongs to the broader topic of network optimization, where one has to solve a minimization problem of the type:

To ease up notation, let us define:

$$\mathcal{X} \coloneqq \{f \in \mathbb{R}^{|\mathcal{E}|} : Bf = \nu \cap f \geq 0\} \tag{18}$$

Then it is immediate to define the current problem as a Flow Optimization problem in the form:

$$\min_{f \in \mathcal{X}} \quad \sum_{e \in E} \psi_e(f_e). \tag{19}$$

Problem (19) is actually a specific case of network optimization, called System Optimum Traffic Assignment Problem (SO-TAP). The objective function of Problem (19) can be re-written as:

$$\min_{f \in \mathcal{X}} \sum_{e \in \mathcal{E}} \frac{f_e l_e}{1 - f_e/c_e} \tag{20}$$

We solved this Convex Optimization problem by turning to the CVXPY library. The resulting optimal flow f^*, a vector in $\mathbb{R}^{28}$ can be found in the code that comes with this report and in Figure (12).

3.5 Question 5

In the SO-TAP framework, users will not follow the optimal flow, but will instead move along the path which optimises their selfish goal, i.e., they will move along the path which minimizes the delay

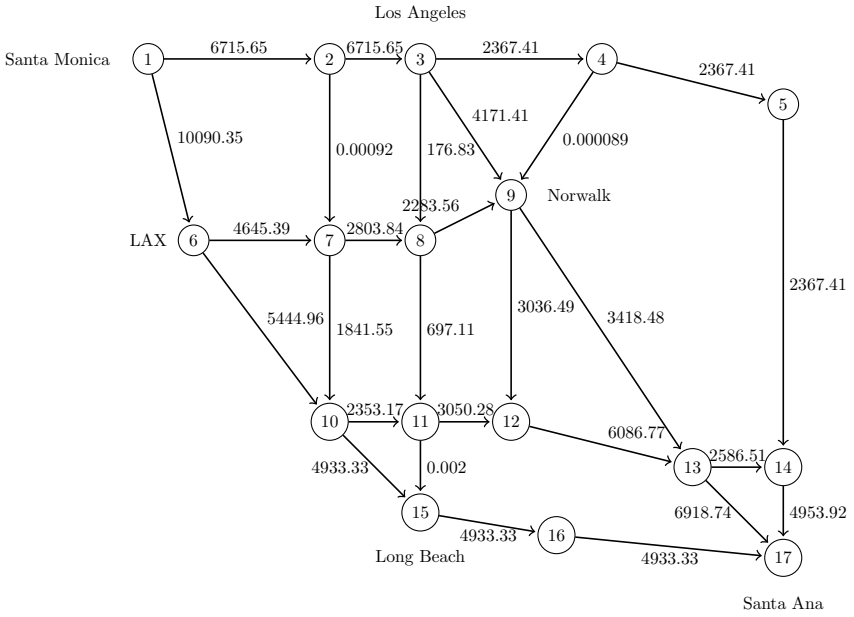


Figure 13: Flow vector at Wardrop equilibrium for the delay function $\tau_e = \frac{l_e}{1-f_e/c_e}$

they experience by themselves. This concept is formalised by the notion of *Wardrop equilibrium*. The flow vector associated to said condition can be retrieved by solving:

$$\min_{f \in \mathcal{X}} \sum_{e \in \mathcal{E}} l_e c_e \ln\left(\frac{c_e}{c_e - f_e}\right) \tag{21}$$

The flow vector $f^{(0)}$ solution of the Wardrop optimization can be found in the code that comes with this report and in Figure (13).

3.6 Question 6

The idea behind tolls is to make the users *internalize their negative externality*, i.e., directly modify the users' behavior when acting selfishly, as they will now base their actions not only upon the experienced delay but also on the tolls.

Introducing tolls allows us to rewrite the problem as:

$$\min_{f \in \mathcal{X}} \sum_{e \in \mathcal{E}} \left(\omega_e f_e + \int_0^{f_e} \tau_e(s) ds\right) \tag{22}$$

As per Corollary 4.3 of the lecture notes, if every edge's toll ω_e is set to $\omega_e = f_e^* \tau_e'(f_e^*)$, then the Wardrop equilibrium flow $f^{(\omega)}$ coincides with the system optimum f^*.

As expected, when solving the problem using CVXPY, the resulting flow vector coincides with the one found in Section 3.5.

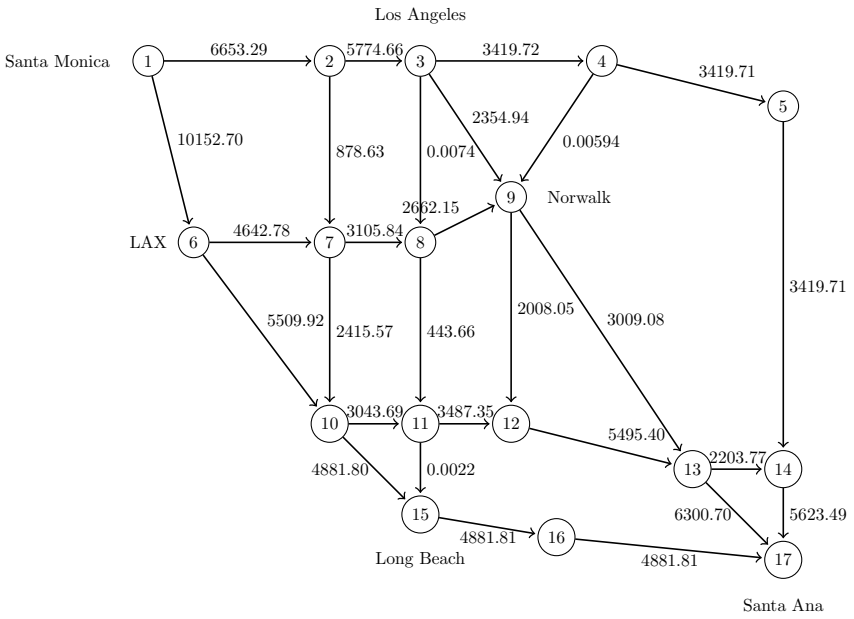


Figure 14: System optimum, minimizer of the cost function $\sum_{e\in\mathcal{E}} f_e(\tau_e(f_e) - l_e)$

3.7 Question 7

3.7.1 Social Optimum f^*

The answer to this question is the solution of the following problem:

$$\min_{f\in\mathcal{X}} \sum_{e\in\mathcal{E}} \frac{f_e l_e}{1 - f_e/c_e} - f_e l_e \tag{23}$$

Plugging the problem into CVXPY, we obtained the numerical solution, a vector in $\mathbb{R}^{28}$, presented in our code and in Figure (14).

3.7.2 Construct tolls ω^* such that the Wardrop equilibrium coincides with f^*.

Clearly, the Wardrop equilibrium did not change compared to 3.5, since the delay experienced by the users is exactly the same. What did change is the system optimum flow.

The idea is to set the vector toll ω^* as the difference between the marginal social cost $\psi'(f_e)$ and the delay τ_e, i.e. the only cost experienced by the user in the absence of tolls. Specifically, one can construct tolls so that the marginal costs are equal at the optimal f^*:

$$\omega_e^* = \psi_e{}'(f_e^*) - \tau_e(f_e^*) = \frac{l_e}{1 - f_e^*/c_e} + f_e^* \frac{l_e}{c_e(1 - f_e^*/c_e)^2} - l_e - \frac{l_e}{1 - f_e^*/c_e} = f_e^* \tau_e'(f_e^*) - l_e \tag{24}$$

Indeed, when using these tolls, the two solutions coincide, as tolls serve as a tool to make users internalize in their own decision making process their negative externalities.