First homework Networks Dynamics and Learning

Emanuele De Leo
Politecnico di Torino
s318658
s318658@studenti.polito.it

Giovanni Grossi
Politecnico di Torino
s318690
s318690@studenti.polito.it

Luca Pesce
Politecnico di Torino
s318658
s318658@studenti.polito.it

***Abstract*—In this first homework assignment, we dealt with topological and algebraic graph theory the assignment is divided into three different exercises :**

- **In the first we are given a network and its link capacities.**
- **In the second there is a list of people with characteristics to link.**
- **In the more realistic third one there is to analyze the lines of highways in the city of Los Angles.**

I. ST EXERCISE

Consider the network in Figure 1 with link capacities:

$$c1 = c3 = c5 = 3, \quad c6 = 1, \quad c2 = c4 = 2.$$

A. Minimum Aggregate Capacity Removal

The minimum aggregate capacity that needs to be removed for no feasible flow is the minimum total capacity of edges that, if removed, disconnects the source from the sink in the network, rendering it impossible to have a flow between the two nodes. To compute the minimum aggregate capacity we used the Ford and Fulkerson's algorithm finding the minimum capacity cut. The minimum capacity cut is equal to 5. Therefore, the minimum aggregate capacity that needs to be removed for no feasible flow is 5.

B. Maximum Aggregate Capacity Removal

To find the maximum aggregate capacity that can be removed from the links without affecting the maximum throughput, we iterated over each edge, removing its capacity, and computing the maximum throughput at each step. We collected the units of capacity which do not affect the maximum throughput calculated in the previous point. The maximum aggregate capacity turned out to be equal to 3.

C. Extra capacity distribution for Maximum Throughput

Given $x > 0$ extra units of capacity ($x \in \mathbb{Z}$), we should distribute them in order to maximize the throughput from 'o' to 'd'. This is achieved by iteratively adding one unit of extra capacity to the first edge in the min-cut set computed at each step. We can observe the impact on the maximum throughput as extra capacity is increased. The plot in Figure 2 illustrates the relation between maximum throughput and the extra capacity (x).

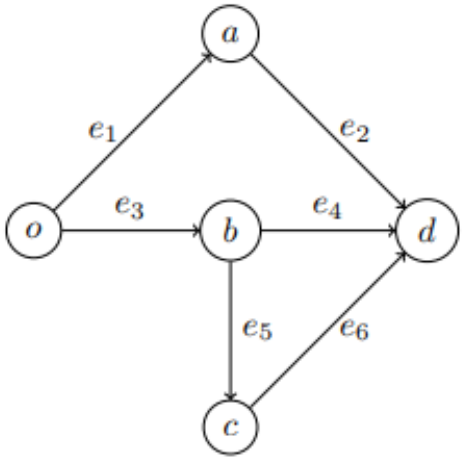

Fig. 1. Network exercise 1

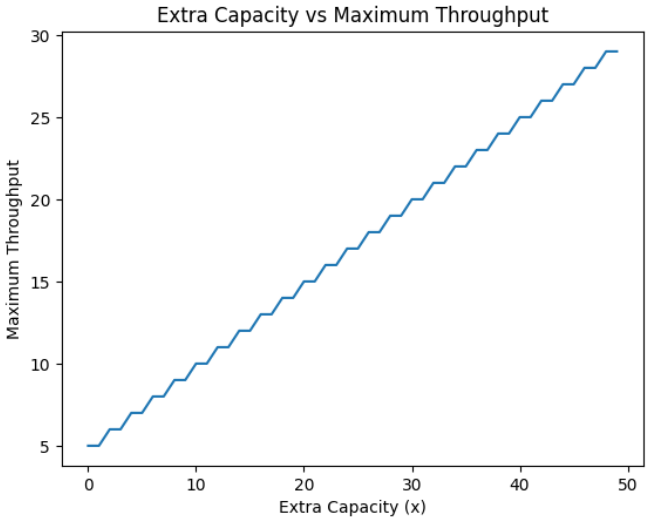

Fig. 2. Extra capacity and maximum throughput

II. ND EXERCISE

In Exercise 2, we are given a set of people $\{p1, p2, p3, p4\}$ and a set of books $\{b1, b2, b3, b4\}$. Each person has a subset of books they are interested in, forming a bipartite graph.

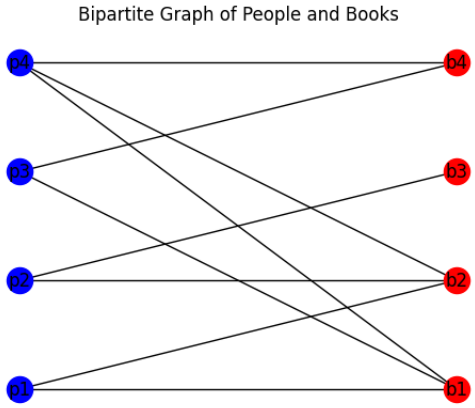

Fig. 3. Graph exercise 2

TABLE I
TABELLA RISULTATI ES2.3

Book To Sell	**B1**
Book To Buy	**B3**
Max Flow	**9**

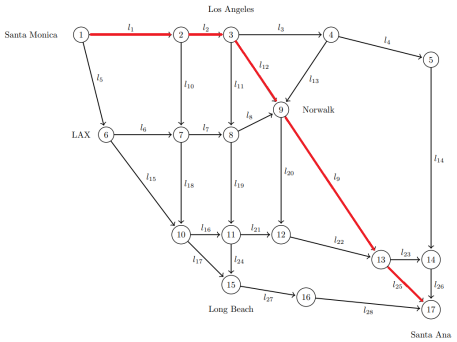

Fig. 4. Shortest path between 1 and 17

The objective is to explore scenarios exploiting max-flow problems.

A. Finding a Perfect Matching

The initial task involves exploiting max-flow problems to identify a perfect matching, if one exists. By modeling the relationships between people and books as a bipartite graph and employing NetworkX, we are able to determine whether a perfect match is achievable. We added a source node and a sink node to the graph, the first one connected to the people nodes and the last one connected to the books nodes. Then we computed the maximum throughput in the graph getting a value of 4. Since the maximum throughput turned out to be equal to the number of people, we concluded that a perfect matching could exist, but in this case it did not exist, because the number of matching people-books exceeded the throughput. The reason is given also by the graph theory; in fact there is a perfect matching if every node in the vertex set is adjacent to exactly one edge in the edges set.

B. Handling Multiple Copies of books

The exercise evolves by introducing the concept of multiple copies of books with a distribution of (2, 3, 2, 2). Each person can acquire an arbitrary number of different books basing on its interest. Leveraging the analogy with max-flow problems, the modified graph now incorporates capacities on edges from books nodes to sink node equal to the number of available book copies for each book. In this graph the maximum flow represents the total number of books of interest which can be assigned. Since the maximum flow is equal to 8, the number of books which can be assigned is 8 (the script provides also the distribution of the books assignment to each person).

C. Optimizing Book Transactions

The goal is to determine which books should be sold and bought to maximize the number of assigned books. The script employs the Ford-Fulkerson algorithm to find the maximum flow in the network. You start by creating a bipartite graph with nodes representing people, books, a source, and a sink. Edges are added between the source and people, and between people and books, with capacities set to 1. Then all the possible book distribution are generated. The script then iterates through these permutations, adding edges between books and the sink based on the proposed distribution. If the new flow value is greater than the previous maximum, the script prints the books to buy and sell, along with the maximum flow value. Additionally, the script visualizes the optimal flow by creating a new graph with positive flow edges. It removes the original arcs from the flow graph and only adds arcs with positive flow. The results are shown on table 1.

Exercise 2 showcases the versatility of network flow algorithms in solving matching problems. While no perfect match was found in Part (a), the subsequent parts demonstrate how the application of max-flow concepts can provide practical solutions and insights into the dynamics of book assignments in a bipartite graph.

III. RD EXERCISE

Exercise 3 provides a detailed examination of the transportation network in Los Angeles using a simplified representation captured in the node-link incidence matrix$B$. This matrix connects nodes and links according to the directions of the edges, each with attributes such as maximum flow capacity c_{e_i}, minimum travel time l_{e_i}, and a delay function $\tau_{e_i}(f_{e_i})$ that accounts for congestion.

A. Shortest Path

The initial task requests to discover the shortest path between two nodes (1 and 17) within the network basing on the minimum overall travelling time. This endeavor is synonymous with identifying the fastest route, considering the minimum traveling times associated with each link. Our script constructs an optimization problem where the objective is to minimize

the total traveling time by adjusting the flow vector v_{e_i} as variable. The constraints ensure that the flow aligns with the specified start and end nodes, moreover that flow values are non-negative. The optimal flow vector, represented in Figure 4, highlights the highest values corresponding to the edges of the shortest path. According to the figure the edges order of the path is:

$1 \rightarrow 2 \rightarrow 3 \rightarrow 9 \rightarrow 13 \rightarrow 17$

B. Maximum Flow

The second part of the exercise requested to compute the maximum flow between nodes 1 and 17. In the context of transportation networks, the maximum flow represents the peak transportable capacity between two specific points respecting the capacity constraint of each edge. We computed it using the Ford and Fulkerson's algorithm proving to be equal to 22448.

C. External in-flow

The external in-flow of the network satisfies the constraint Bf = v; the equation allows to have a balanced system. We easily computed the in-flow vector doing the matrix multiplication between the node-link incidence matrix B and the flow vector f.

D. Social optimum with respect to the delay

We needed to find the social optimum f* considering now the delay function on the different edges and replacing the values of node 1 and node 17 in the exogenous in-flow vector with the first value of the previous result for the external in-flow (16282). To compute f* we minimized the summation of the different delays with respect to all the links imposing the usual constraint that Bf=v. Our script provides the values for the edges and the minimum cost of social optimum that turned out to be equal to 23835,48.

E. Wardrop equilibrium

The Wardrop equilibrium is a path flow distribution such that the traffic in the network is in balance. The condition of equilibrium assumes that hypothetical users select a route that minimizes the cost incurred in its traversal, with the consequence that the cost of each used path is the same and the overall cost is minimal. We found the Wardrop equilibrium in the network minimizing the cost function given by the integral of the delay function through the flow. Our script provides the obtained values for each edge with an overall cost of 14931.38.

F. Wardrop equilibrium with tolls

We introduced specified tolls, in this way we added a variable cost on the use of a particular edge or path within the network in addition to the delay. While in the equilibrium case users choose the paths minimizing the individual cost, in this scenario, with a toll based on the product between the optimum flow and the derivative of the delay function, the cost of using a particular route may vary basing on its congestion. Adding

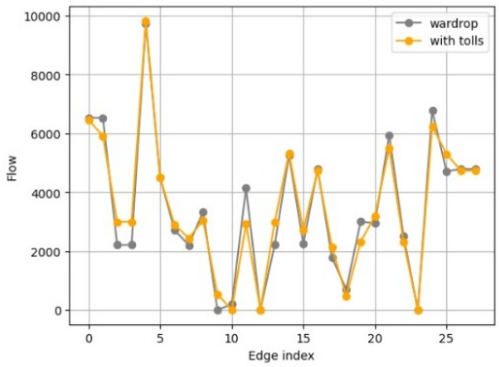

Fig. 5. Comparison of flow distribution with the equilibrium case

$$\omega = \psi' - \tau = f * \left(\frac{l \cdot C}{(C - f*)^2} \right) - l$$

Fig. 6. Construction of toll vector

the toll we expected a change in the flow distribution across the edges, because links with higher tolls could have reduced flow, while others might carry more traffic. We evaluated the Wardrop equilibrium minimizing the cost function given by the sum of the delay and the specified toll using the optimum flow computed in point d. The new overall cost is 52755.69. We observed that the optimal cost was much greater than the cost calculated in the previous point (as we expected). Despite this, the flow distribution across the links was similar to the first one as reported in the chart in Figure 5. The individual preference leads to the increase in cost, because it pushes to avoid toll selecting alternative routes, even if longer. For example we can see that:
- the edge 9 (from node 9 to 13) is crossed in the tolls case, but it is not crossed in the Wardrop equilibrium case
- the edge 10 (from node 2 to 7) is crossed only in the Wardrop case

G. Social optimum with respect to the differential travelling time

We computed the system optimum minimizing the difference between the delay function and the vector of travelling time in free flows. We obtained an overall cost of 13334,30. The next step is to construct a toll such that the Wardrop coincides with the social optimum. We thought about building the toll vector basing on the difference between the derivative of the differential cost and the initial delay (The equation is presented in Figure 6). We computed the social optimum with a cost of 13334,30 and we found the Wardrop equilibrium adding the tolls obtaining a flow distribution identical to the social optimum.