Università degli Studi di Torino

Dipartimento di informatica

Tesi di Laurea Triennale in Informatica

Applied logistics: solving real-world network flow problems using python

Relatore:

Prof: Andrea Cesare Grosso

Candidato:

Stefano Atzeni

ANNO ACCADEMICO 2022/2023

The purpose of computation is insight, not numbers.
- Richard Hamming, 1962

Abstract

This thesis delves into the world of supply chain management, focusing on transportation logistics, and aims to design and apply optimization models for planning transportation routes that meet logistics objectives and challenges.

The first part of the thesis provides an overview of the fundamental concepts in logistics and operations research, a field that deals with making optimal decisions. Moreover, it introduces linear programming to formulate optimization problems such as minimum cost flow and multi-commodity flow problems.

The second part reports the practical implementation of the problems addressed during my internship experience. It illustrates the development of a tool to analyze a transportation network and visualize different route scenarios by solving optimization problems, in order to gain a detailed and insightful understanding.

The tool is based on Python and linear optimization libraries to analyze the historical data with the purpose of being integrated into a working outbound system. Its implementation will contribute to the field of logistics and transportation management, improving the planning process, reducing delays, and enhancing overall delivery times and costs.

This thesis demonstrates the potential of linear optimization models for transportation logistics and provides a practical and flexible tool that can be adapted to different scenarios and objectives. The results show that the tool can improve the efficiency and effectiveness of the transportation network, and suggest further improvements and extensions for future work.

Dichiarazione di originalità

Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata.

Declaration of originality

I declare to be responsible for the content I'm presenting in order to obtain the final degree, not to have plagiarized in all or part of, the work produced by others and having cited original sources in consistent way with current plagiarism regulations and copyright. I am also aware that in case of false declaration, I could incur in law penalties and my admission to final exam could be denied

Contents

Chapter 1

Logistic & Transports

This first chapter provides a brief overview of logistics and transportation [1]. After defining the main concepts in section 1.1 and understanding the importance of logistics from history to nowadays in section 1.2 we pass to a deeper analysis of transportation in section 1.3 covering key phases of its process, its various modes and the levels of abstraction in transportation networks.

1.1 Definition of logistics

Supply chain management (SCM) deals with the process that begins with raw materials and finishes with the final product in the customer's hands. It involves all the activity in sourcing, conversion, and logistics. Usually, it consists of several partner companies that cooperate towards a common goal, each one providing a key node in the supply chain.[2, 3]

The Council of Supply Chain Management Professionals (CSCMP) defines logistic management as the "part of supply chain management that plans, implements, and controls the efficient, effective forward and reversed flow and storage of goods, services, and related information between the point of origin and the point of consumption in order to meet customers' requirements."[2] Therefore it encompasses inbound and outbound transport management within supply chain facilities, transporter fleet management, logistic network design, route planning, warehousing, and inventory management.

In simple terms we can refer to logistics using the widely used seven R's concept, that is, getting the Right product, in the Right quantity, in the Right condition, at the Right place, at the Right time, to the Right customer, at the Right price.

Transportation is an integral part of logistics and covers everything that involves the movement of goods from one point to another in the supply chain. Optimizing the flow of goods to meet market demands, while minimizing costs and maximizing customer satisfaction.

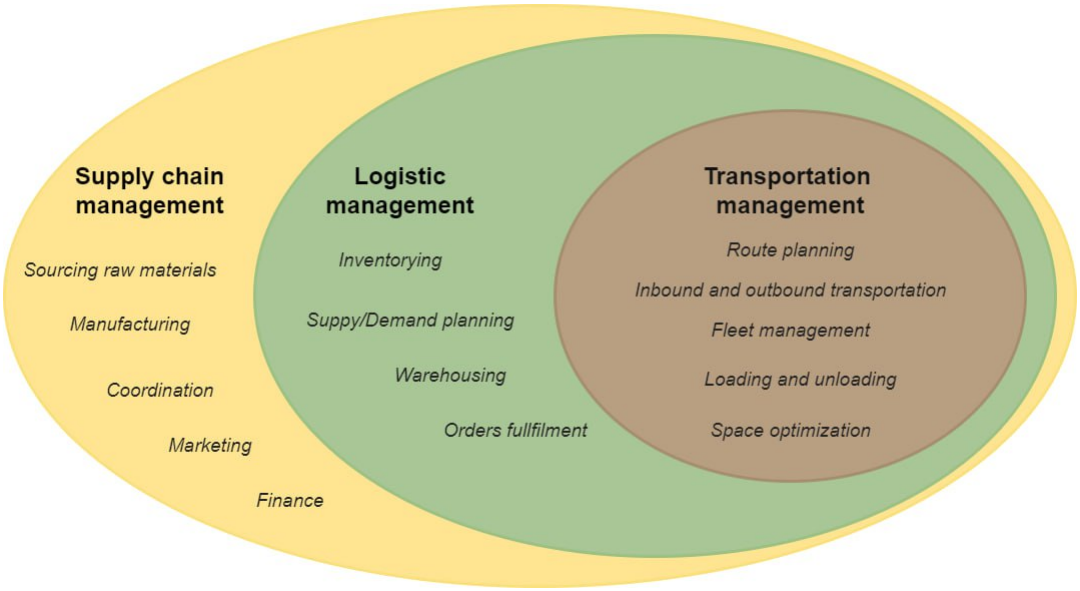


Figure 1.1: Relationship between supply chain, logistics, and transportation management. Adapted from MD Sarder. Logistics Transportation Systems 2020. [1]

1.2 Importance of logistics

Logistics has very ancient origins, with the most basic transport techniques based on wagons and animals. Its value has always been rooted in the military field: transporting troops, weapons, food, and supplies quickly and strategically are undoubtedly activities of major warfare interest. Many great strategists were in fact, logisticians.

With the advent of the Industrial Revolution, the invention of the steam engine, and the first modern technologies, logistics gained more importance. The transport network increased in both velocity and size, consolidating the backbone structure for the beginning of mass consumption.

During the 20th century, with the outbreak of the World Wars, the potential of new scientific-technological discoveries led to strong military interest, causing considerable development in all fields, including logistics, which also began to rely on theoretical aspects of optimization.

In the post-war period, the automobile became a common household commodity, and air transport was also fully established. With the onset of globalization, the transport of goods and services around the world became crucial, and the importance of logistics was recognized not only in the context of war but also, and especially, in business. The economic boom led to the creation of a massive and extensive transport network with the aim of satisfying the needs arising from increasing consumerism. As a result, logistics became essential to manage this high level of complexity.

Finally, in recent years, increasing the speed and decreasing the cost of transport has become a crucial element for the competitiveness of companies aiming to distinguish themselves in an overcrowded market where attention to customer needs is increasingly important. However, in the present day, we already have a sufficiently fast transport network. What can make the difference is the optimal use of the network by carefully planning routes in advance. Let us assume that two companies

produce the same good at the same production cost and use the same (or comparable) transport network. In this scenario, optimal logistics management becomes the factor that could result in a lower final price for the consumer, leading to an element of differentiation and a higher market acquisition. There is no doubt that customers will prefer to purchase a product with no added shipping costs or one that is available for next-day delivery.

1.3 Transportation

In this section, we will look more in depth at transportation to better understand some of its proprieties and interesting aspects.

1.3.1 Transportation phases

The transportation process consists of five fundamental phases:

1. **Route Planning**: Establishing the optimal delivery path for the transporter.
2. **Load Planning**: Efficiently arranging shipments within the transporter to minimize unnecessary handling.
3. **Truck Servicing**: Ensuring the transporter is fueled and adequately prepared for the journey.
4. **Movement**: Physically moving the transporter along the predetermined route.
5. **Delivery**: Reaching the shipment's final destination, where the consumer receives the requested goods.

We will focus on the first phase: route planning. As the upstream process in the entire transport chain, it plays a pivotal role in the field of logistics due to its profound impact on operational efficiency and cost optimization. Efficient route planning ensures that goods are transported from their origin to destination in the most time-effective and cost-efficient manner possible. This translates to reduced fuel consumption and emissions, minimized vehicle wear and tear, and ultimately, lower transportation costs.

1.3.2 Transportation modes

There are five main types of transport *roadway*, *railway*, *airway*, *waterway*, and *pipeline*. Each with different disadvantages and advantages as briefly illustrated in Table 1.1.

Multimodal transport is the transportation of a certain commodity via several modes. The advent of this type of transport was greatly facilitated by the invention of the modern container in 1956 by Malcolm McLean, which is still in use today. Before this standard, moving goods from one means of transport to another required unloading and reloading them, making multimodal transport costly in terms of time, and labor.

In the present day, however, embracing multimodal transport has become crucial to exploit the advantages of each individual transportation mode, despite the associated increase in costs and organizational complexity. Common multimodal combinations include ship-truck (fishyback), air-truck (birdyback), and rail-truck (piggyback).

Roadway Road transport usually employs trucks to carry finished or semi-finished products for long-medium distance journeys, but mainly for final deliveries. As it is based on the public road network, it allows a high degree of flexibility and diversification in the choice of routes. However, physical and governmental limitations do not allow it to adopt an economy of scale. Roadway transportation can be truckload(TL) if the truck is fully loaded or less-than-truckload(LTL) otherwise, in this case, it is usually more convenient to join more shipments into one, in a process called consolidation.

Railway One of the major advantages of rail transportation is its cost efficiency, owing to its ability to carry large quantities of goods in a single journey. Due to its lower speed, it is particularly suited for non-time-sensitive shipments, for example, it holds significant value for industries such as mining, agriculture, and manufacturing. Furthermore, rail transport is inherently more environmentally friendly than road or air transport, thanks to its lower carbon emissions per ton-mile. However, rail transport lacks flexibility due to its fixed routes.

Airway Air transport is undoubtedly the fastest mode of transportation, but it is also particularly expensive and possesses limited capacity. It is usually reserved for time-sensitive and high-value products, allowing for the rapid coverage of long distances in a short period.

Waterway Although it is relatively slow, maritime transportation can cover extensive distances around the globe and carry a vast amount of goods. It is much less expensive than air transportation, owing to its advantageous quantity/cost ratio. In the globalization era, it assumes a pivotal role in facilitating the movement of significant volumes of goods.

Pipeline The use of pipelines is reserved for the specific transportation of liquids, such as gas, oil, or water. They are employed to connect locations where using other forms of transportation is challenging. Pipelines are particularly economical due to their low operating costs, despite requiring a substantial upfront investment for their construction. The planning of these structures must be meticulous due to their intricate inflexibility. This mode of transportation becomes feasible when supply and demand need to remain in balance over an extended period, such as between an oil source and a refinery.

Table 1.1 Transportation modes comparison [1]

	Roadway	Railway	Airway	Waterway	Pipeline
Speed	Moderate	Slow	Fast	Slow	Moderate
Availability	High	Moderate	Moderate	Low	Low
Flexibility	High	Low	Moderate	Low	Low
Length	Short-Long	Medium-Long	Long	Medium-Long	Long
Operation cost	Moderate	Low	High	Low	Low
Capacity (tons)	10-25	50-12,000	5-12	1000-6000	

1.3.3 Transportation network

A *transportation network* is composed of a set of facilities like manufacturers, warehouses, and distribution centers (DCs) linked by transportation infrastructure. Transportation networks can be viewed at different levels of abstraction:

Physical network On the lowest level we are considering the actual path that commodities take from the origin to the destination, using the different transportation infrastructures, namely the physical network. It is composed of tangible elements like roads, rails, ports, vehicles, and more. It deals with low-level problems, such as traffic and weather-related issues.

Operation network If we abstract from the particular physical network issues, we can consider the facilities as decision points. Each transport can then be analyzed based on its key attributes, such as cost, distance, velocity, and capacity, forming the Operational network. In these terms, a route is just a sequence of nodes, that is decisions, that goods must pass through.

Strategic network At the highest level of abstraction, we encounter the strategic network, wherein entire routes are regarded as straightforward connections between origins and final destinations, characterized by their total cost and distance. This comprehensive perspective is indispensable for shaping business-centric decisions, including supply and demand management.

In our discussion of transport networks, we will focus on the operation network view. This approach allows us to utilize the appropriate level of abstraction for applying mathematical structures, such as graphs. Having all the necessary information to compute optimal routes and make considerations about the entire network, without getting lost in meticulous details and technicalities of the logistics transport sector.

To clarify, we are not aiming to create a navigation tool like Google Maps. We are not looking to provide a precise path on the map. We are not interested in streets, traffic lights, and crossroads. We need to abstract from these - still important - elements and focus on a larger scale to make comprehensive decisions.

Chapter 2

Operations research

This chapter provides a comprehensive understanding of Operations Research. Section 2.1 introduces the main concepts and historical context. Then we pass to essential components such as linear programming [4] in Section 2.2 and the simplex method in Section 2.3. Finally, we explore considerations related to complexity in Section 2.4.

2.1 What is operations research

Operations research (OR) is the field of applied mathematics that uses advanced analytical methods to help make efficient management decisions. In other words OR applies its scientific and technological base to resolving problems in which the human element is an active participant. As such, OR is the science of decision making, the science of choice.[5] Given its immaterial nature, it is difficult to define exactly what OR concerns because there is a lot of overlap with other scientific fields, figure 2.1 helps visualize some of the major intersections between disciplines.

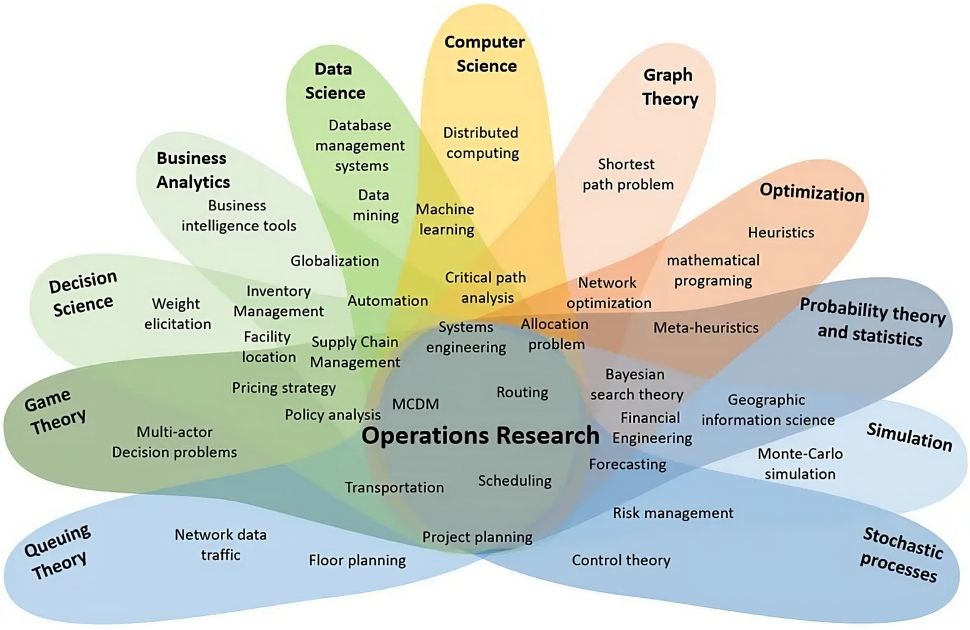


Figure 2.1: Overview of the disciplines and problems related to OR. Illustrated by Alex Elkjær Vasegaard.[6]

But what does it mean to make a decision? A decision process is composed of 5 phases:

- Problem identification;
- Reality analysis and data collection;
- Construction of the model;
- Determination of one or more solutions;
- Analysis of the results obtained

These phases are not strictly sequential and can be iterated until the desired solution is achieved.

OR origin is usually attributed to the military services during World War II. As said previously, during that time, a lot of effort was made to use scientific approaches and new technologies to solve problems, leading to advancements across various fields. OR is a comprehensive approach to problem-solving, originating from the minds of intelligent individuals who were faced with unprecedented challenges. Under the pressures of military exigencies, they developed mathematics tools, formulated theories, built models, and designed techniques to find efficient solutions. Many of the standard tools of OR, such as linear programming, dynamic programming, queuing theory, and inventory theory, were relatively well developed before the end of the 1950s[7]. One of the pivotal outcomes, the *simplex method* for solving linear programming problems, was formulated by George Dantzig in 1947.

The immense potential and vast applicability of OR led to its successful transfer to post-war commerce and industry. Also made possible by the development of computers capable of performing calculations that would have been excessively time-consuming for humans. Today OR is used everywhere, from agriculture to the space industry, and is indispensable for any business management. After all, what is management if not decision-making? It is therefore essential to use efficient tools to make informed decisions that ensure optimal solutions. Some of the applications of OR relate exactly to what we have already discussed above, such as Supply chain management, Logistics, and Routing, but there are numerous notable other uses.

2.2 Linear Programming

Linear programming (LP) is a method for modeling problems using linear relationships. Generally, it aims to maximize or minimize an *objective function*, choosing the best possible assignments for *decision variables*, but respecting certain *linear constraints*.

Decision variables, as the name suggests, correspond to the unknown values on which we can act to solve the problem, that is our decision. They often represent quantities ($x_1 \dots x_n \in \mathbb{R}$ or $x_1 \dots x_n \in \mathbb{Z}$) or binary decisions ($x_1 \dots x_n \in [0, 1]$)

The objective function is a linear function that usually represents the total cost or profit associated with the assignment of decision variables. Since we can always convert a minimization problem into a maximization problem and vice versa, we always use maximization as a convention in the standard form.

Constraints are expressed as linear equalities or inequalities and are the most characteristic part of a particular problem because they define the feasible region of possible solutions that satisfies the constraints.

$$a_{11}x_1 + \cdots + a_{1n}x_n \quad \phi \quad b_1 \quad \text{with} \quad \phi \in (\leq, \geq, =)$$

Again, it is always possible to convert the various forms of constraint, the standard form convention is to pose the inequalities as less-than and to stipulate that all the decision variables be non-negative.

Finally, we can describe a generic LP problem in standard form as follows:

$$\begin{aligned} \text{maximize} \quad & z = c_1x_1 + \cdots + c_nx_n \\ & a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1 \\ & \vdots \\ \text{subject to} \quad & a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m \\ & x1, \ldots x_n \geq 0 \end{aligned}$$

A set of assignments for the variables $(x_1 \ldots x_n)$ is called a *solution*. A solution is *feasible* if it satisfies all constraints and it is also called *optimal* if there is no other solution with a higher objective function value.

A problem is called *infeasible* if no feasible solution exists, i.e. if two or more constraints cannot be satisfied simultaneously. On the other hand, a problem is said to be *unbounded* if it is possible to give arbitrarily high values to the decision variables, i.e. there are infinitely many optimal values.

Of course, reality cannot always be formulated through linear relationships, so it is often complicated to model certain problems. Thus, linearity itself turns out to be a limitation of the approach and there are other fields of mathematical programming where other types of models are used, however, these techniques are much more complex to handle and not as mature as LP.

2.3 Simplex method

The simplex method is the primary algorithm for solving Linear Programming (LP) problems. To give a basic overview of the algorithm, we will consider a geometric interpretation of what happens during its execution. In reality, the same concepts are transposed in terms of linear algebra in order to be computed by a machine. Those details will not be discussed, as a detailed and in-depth explanation of the simplex is outside the scope of this thesis.

- **Solution Region**: The linear constraints of the problem define the set of possible solutions, also called feasible solution region, which can be geometrically interpreted as a convex polytope, i.e. n-dimension polygon. If the feasible region is empty, the problem is unsolvable.

- **Fundamental Theorem of Linear Programming**: This theorem asserts that if an optimum exists for the objective function, it resides at a vertex of the polytope. Thus, there is no need to examine all infinitely many solutions within the feasible region. Only a finite number of solutions need to be considered since the vertices of the convex polytope are also finite.

- **Optimality Condition**: If a vertex is not optimal, there must be an edge across which moving would result in higher values of the objective function; otherwise, it would already be considered optimal. If this edge is finite, we will eventually reach another vertex with a higher value, possibly optimal. Otherwise, if the edge is infinite, the problem is unbounded.

Given these premises, the intuition of the algorithm is straightforward:

Phase 1: Find an initial solution (i.e. vertex). Sometimes this phase is trivial, while in other cases, it may require solving a modified version of the problem. If an initial solution can't be found the problem is infeasible.

Phase 2:

1. **Optimality Check**: If there is no edge along which the objective function improves, the optimal solution to the problem has been found. Otherwise, proceed to step 2.
2. **Moving Along an Edge**: Move along an edge leading to higher values, if another vertex is reached, return to step 1. Otherwise, the problem is unbounded.

In short, the simplex method is an iterative approach to navigate the vertices of the convex polytope defined by an LP problem, until the optimal solution is identified, as illustrated in Figure 2.2.

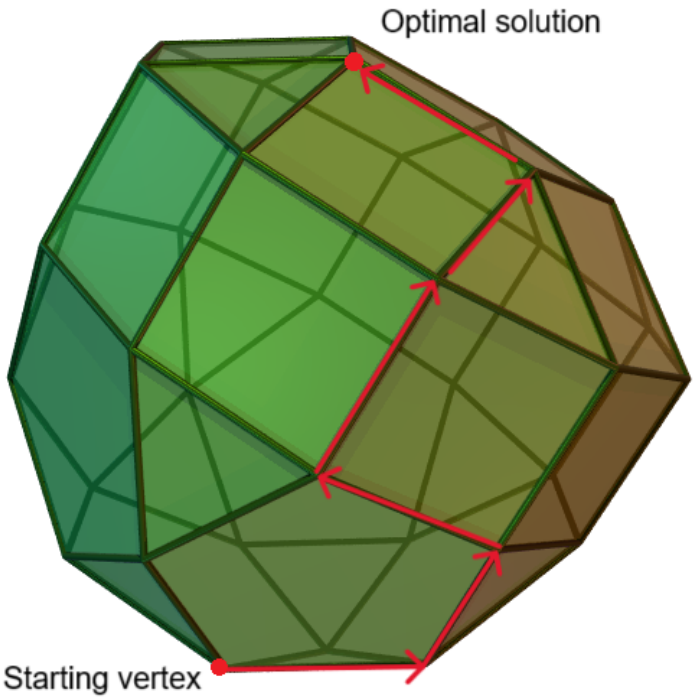


Figure 2.2: Edges traversed by the simplex algorithm in a 3D polytope: from initial vertex to optimal solution.
Based on commons.wikimedia.org/wiki/File:Elongated_pentagonal_orthocupolarotunda.png

2.4 Complexity

LP problems with real variables are in class P, meaning that they can be solved in polynomial time. However, for large-scale problems, LP problems can become difficult to handle.

The simplex method is exponential in the worst case, but in practical cases, it often finds a solution in a reasonable time, making it very valid and widely used. If integer variables are used (Integer-LP, ILP) or both integer and real variables are used (Mixed-ILP, MILP), the complexity becomes NP-hard, meaning that no polynomial-time algorithms are known for solving them. They are still tractable within certain sizes using other techniques and algorithms such as branch and bound and cutting plane.

It is not surprising that using integer variables requires more computation, as we are adding many more constraints to our solutions. Moreover, the Fundamental Theorem of Linear Programming is not directly applicable to ILP because of the discrete nature of the variable.

In general, in this field the theoretical complexity of algorithms is not so decisive. Often it is possible to work on a problem and look for specific methods to solve it. There are various techniques and heuristics to improve the performance of algorithms, however we will not go into further details of the solution methods. We will treat solvers as black boxes, given the formulation of a problem, we expect a result, without paying attention to the internal procedure.

Chapter 3

Network flow problems

We already discussed transportation networks, in this chapter we introduce their mathematical generalization in section 3.1, 3.2 and delve into some of the most significant problems [8] such as the maximum-flow 3.3, the shortest path 3.4, the minimum-cost flow 3.5, and the multi-commodity flow 3.6 problems.

3.1 Graphs

In graph theory, a *graph* G is a mathematical structure defined by a set of *vertices* (or *nodes*) V and a set of *edges* (or *arcs*) E that connect them.

$$G = (V, E)$$

Such a simple structure turns out to be a great tool for modeling real-world problems. Graphs permeate our world, manifesting in electric grids, road systems, social networks, and so on. They find vast applications and enable us to unravel complex connections and unlock insights across various domains, from studying gene interactions to optimizing neural networks.

Graphs can take various forms, each with its own characteristics. a fundamental distinction is between *directed* and *undirected* graphs:

- **Directed:** In a directed graph, each edge has a direction, meaning it goes from one vertex to another. This direction can be represented by an arrow. Formally, an edge e in a directed graph is an ordered pair of vertices, $e = (u, v)$, where u is the source vertex, and v is the destination vertex.
- **Undirected**: In an undirected graph, edges have no direction, and they simply connect two vertices. Formally, an edge e in an undirected graph is an unordered pair of vertices, $e = \{u, v\}$, where u and v are connected.

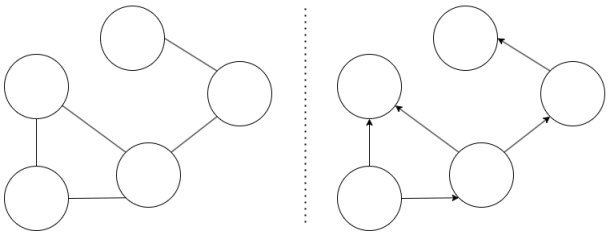


Figure 3.1: Example of an undirected graph and a directed graph.

3.2 Flow networks

To expand the expressive power of graphs, we can add attributes to their components. In this case, they are often referred to as *networks*. For example, we can label nodes with names or add numeric values to edges.

A *flow network* is a model for a network where a resource is transferred from one node to another passing through edges (e.g. water, electricity, data packages).

Flow networks are essential to solve problems involving the efficient distribution of resources or information in a network, taking into account capacity constraints and flow conservation principles. In our case study, a transportation network is a flow network where flow represents the movement of a certain commodity from one facility to another.

Formally is defined by a directed graph $G = (V, E)$, a *source* node s, a *sink* node t, and a *capacity function* $q : E \to \mathbb{R}$. (In some cases a *cost* function $c : E \to \mathbb{R}$ can be also added).

$$N = (G, s, t, q)$$

We can now introduce the concept of *flow* as a function $f : E \to \mathbb{R}$ modeling the units passing through an edge. Additionally, we define a *net-flow* function $X_f : V \to \mathbb{R}$ to represent the total flow remaining in a node.
Let IN(w) and OUT(w) be the sets of incoming and outgoing edges from node w:

$$X_f(w) = \sum_{(u,v) \in IN(w)} f(u, v) \quad - \sum_{(u,v) \in OUT(w)} f(u, v)$$

We say that a node v is *active* if $X_f(v) \geq 0$, denoting that some units remain in the node, *deficient* if $X_f(v) \leq 0$, indicating a 'debt' of units that must be created, or *conserving* if $X_f(v) = 0$, meaning that the exact amount of flow entering the node is equal to the flow leaving it.

f is a flow function if the following proprieties are satisfied:

- **Skew symmetry constraints:** The flow of an edge from u to v is the opposite of the flow from v to u, i.e. $f(u, v) = -f(v, u)$. This implies that we use negative flow to indicate the opposite direction.

- **Capacity constraints:** The flow through an edge cannot be higher than its maximum capacity. i.e. $f(u, v) \leq q(u, v)$

- **Source & Sink definition:** The source s must be deficient, i.e. $X_f(s) \leq 0$ with a negative net flow because it produces units. Similarly, the sink t must be active, i.e. $X_f(t) \geq 0$ with a positive net flow because it consumes the remaining units.

- **Flow-conservation:** All the nodes except source s and sink t must be conservative, with a zero net flow i.e. $\forall v \in V \setminus \{s, t\} \quad X_f(v) = 0$ meaning that only the source and sink can produce and consume flow, respectively.

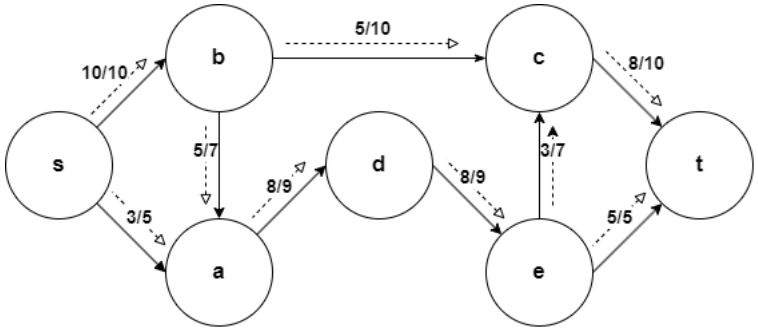


Figure 3.2: Example of a flow network, flow and capacity are denoted f/c

Sometimes it is necessary to model multiple sources. This can be made possible by introducing a *super-source* node connected with each source with infinite capacity, acting as a global source. A similar construction can be made for a *super-sink*.

3.3 Maximum-flow problem

The maximum flow problem involves finding the maximum amounts of goods that can be transferred from the source s to the sink t. In other words, finding the maximum flow passing through a network while respecting all capacities.

We define the value $|f|$ of a feasible flow as the net flow entering the sink node t, i.e. $|f| = X_f(t)$. Correspondingly, the value f is also equal to the net flow outgoing from the source s, i.e. $|f| = -X_f(t)$.

Given a flow network $N = (G, s, t, q)$, the max-flow problem can be formally stated as follows:

$$\text{maximize} \quad |f|$$

subject to:

$$\sum_{(u,v)\in IN(w)} X_{u,v} \quad - \sum_{(u,v)\in OUT(w)} X_{u,v} = \begin{cases} -|f| & \text{if } w = s \\ f & \text{if } w = t \\ 0 & \text{otherwise} \end{cases} \qquad \forall w \in V$$

$$X_{u,v} \leq q(u,v) \quad \forall (u,v) \in E$$

$$X_{u,v} \in \mathbb{Z}_0^+ \quad \forall (u,v) \in E$$

We want to maximize the flow value $|f|$, using the variables $X_{(u,v)}$ to represent the flow through the edge (u, v). We also constrain all the flow $|f|$ to originate only from the source s and be consumed only at the sink t, while ensuring that the capacity is not exceeded for any edge.

In this linear programming form, we could solve the max-flow problem using the simplex method, but there are various algorithms that solve the problem faster such as Ford–Fulkerson algorithm.

3.4 Shortest path problem

The shortest path problem has been an essential and extensively explored topic with various practical applications. It involves finding the shortest path from one node to every other node, but there are multiple variations, such as finding the best route from one node to another or from each node to every other node.

In this problem, we introduce a cost function c that represents the cost of traversing a specific edge. This cost can represent various factors, not limited to monetary cost, and it serves to weigh the edges, giving a reason to prefer one path over another. In this sense, a "shortest" or "best" route is not necessarily the one that involves fewer nodes, but we can define the length of a directed path as the sum of the lengths of edges in the path.

Additionally, there is no need to use a flow network since we do not consider capacities, but we might view the problem as sending 1 unit of flow as cheaply as possible from node s to each of the nodes in $V \setminus \{s\}$ in a flow network with infinite capacities.

Given a graph $G = (V, E)$, a source node s, and a cost function $c : E \to \mathbb{R}$, the shortest path problem can be formally stated as follows:

$$\text{minimize} \quad \sum_{(u,v)\in E} c(u,v) \cdot f(u,v)$$

subject to:

$$\sum_{(u,v)\in IN(w)} X_{u,v} \quad - \sum_{(u,v)\in OUT(w)} X_{u,v} = \begin{cases} -(|V|-1) & \text{if } w = s \\ 1 & \text{if } w \neq s \end{cases} \quad \forall w \in V$$

$$X_{u,v} \in \mathbb{Z}_0^+ \quad \forall (u,v) \in E$$

We want to minimize the cost, ensuring that each node is reached with exactly one unit, and the source s sends one unit to each node except itself, that is $|V| - 1$.

Usually, the shortest path problem is not defined and solved as an LP problem due to its particularly simple structure that has allowed researchers to develop several intuitive algorithms such as Dijkstra's Algorithm.

3.5 Minimum-cost flow problem

The minimum cost flow problem is a fundamental optimization problem in network flow theory because it can be considered a generalization of the previously mentioned problems. Previously, we have discussed the max-flow problem and the shortest path problem, each addressing a specific aspect of flow networks. The max-flow problem deals only with capacities, while the shortest path problem focuses exclusively on costs. Now, we combine both of these aspects in the min-cost flow problem, which involves finding the most cost-effective way to route resources through a flow network while respecting capacity constraints.

Given a flow network $N = (G, s, t, q)$ and a cost function $c : E \to \mathbb{R}$, the min-cost flow problem, requiring an amount of flow d to be sent from source s to sink t, can be formally stated as follows:

$$\text{minimize} \quad \sum_{(u,v)\in E} c(u,v) \cdot f(u,v)$$

subject to:

$$\sum_{(u,v)\in IN(w)} X_{u,v} \quad - \sum_{(u,v)\in OUT(w)} X_{u,v} = \begin{cases} -d & \text{if } w = s \\ d & \text{if } w = t \\ 0 & \text{otherwise} \end{cases} \qquad \forall w \in V$$

$$X_{u,v} \leq q(u,v) \quad \forall (u,v) \in E$$

$$X_{u,v} \in \mathbb{Z}_0^+ \quad \forall (u,v) \in E$$

We aim to determine the flow values for $X_{u,v}$ that minimize the overall cost, which is the result of multiplying the edge costs by the flow quantities on those edges. The constraints ensure that flow conservation is maintained at each node, with the source supplying the required flow and the sink receiving it, while the flow on each edge does not exceed its capacity.

This problem could be solved as an LP problem using the simplex, but there are other algorithms that take advantage of the fixed structure of flow networks to solve the problem faster such as the network-simplex.

3.6 Multi commodity flow problem

In previous problems, we have focused on a single type of goods, i.e. commodities that must transit through our network. The multi-commodity flow problem arises in situations where multiple commodities share common resources within the network structure. Similarly to the min-cost flow problem, our objective is to discover the most cost-effective routes to meet the various demands of commodities while respecting the constraints imposed by shared edge capacities.

When commodities are entirely independent, addressing a problem with several commodities becomes a straightforward task, it involves solving each single-commodity problem separately using techniques mentioned previously. However, in cases where commodities depend on shared facilities and their demands intersect, the individual single-commodity problems are no longer independent. To achieve an optimal flow, it is necessary to solve the problem with a different approach.

Given a flow network $N = (G, s, t, q)$ and a set of commodities K defined by $k = (s_k, t_k, D_k) \in K$ where s_k and t_k are, respectively, source and sink of commodity k with demand D_k. Let $C^k(u, v)$ be the cost of transporting a single unit of commodity k along edge (u, v). The multi-commodity flow problem can be stated as follows:

$$\text{minimize} \quad \sum_{k \in K} \sum_{(u,v) \in E} C^k(u, v) \cdot X^k_{u,v}$$

subject to:

$$\sum_{(u,v) \in IN(w)} X^k_{u,v} \quad - \sum_{(u,v) \in OUT(w)} X^k_{u,v} = \begin{cases} -D_k & \text{if w is an source for } k \\ D_k & \text{if w is a sink for } k \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in K, \forall w \in V$$

$$\sum_{k \in K} X^k_{u,v} \leq q(u, v) \quad \forall (u, v) \in E$$

$$X^k_{u,v} \in \mathbb{Z}^+_0 \quad \forall (u, v) \in E$$

Let $X^k_{u,v}$ denote the quantity of commodity k assigned to edge (u, v). Our objective is to minimize the total transportation cost for all commodities. This cost is determined by the product of the unit cost $C^k_{u,v}$ and the quantity $X^k_{u,v}$ for each commodity k, as it is transported through each arc. The capacity constraint guarantees that the sum of all commodities passing through an edge remains within its capacity.

Chapter 4

Work Report

4.1 Blue Reply

Figure 4.1: REPLY and Blue reply logos

REPLY is an Italian IT consulting company founded in Turin in 1996, and it has since expanded its presence globally. The company specializes in system integration and digital services applications, focusing on the design, implementation, and maintenance of solutions based on the Internet and social networks.

REPLY employs a network model comprising numerous subsidiary companies, all under the umbrella of a parent company. Each subsidiary is dedicated to a specific business area, operating in diverse sectors such as big data, cloud computing, Artificial Intelligence (AI), digital media, and the Internet of Things (IoT)

Thanks to the companies in its network, REPLY operates in the following sectors:

- Energy and Utilities
- Telecom, Media & Entertainment
- Industrial Products
- Distribution & Transportation
- Banking
- Insurance
- Public Sector
- Healthcare
- Cybersecurity
- Esport

Blue Reply stands out as one of the leading technology consulting companies within the REPLY group. Specializing in consulting, design, and the implementation of innovative differentiation solutions based on open-source technologies and partnerships with market-leading vendors. Blue Reply's extensive expertise spans various domains, including Low Code, Big Data & Analytics, Microservices & DevOps, Business Consulting, Commerce Data Platform, Industry 4.0 & IoT, and Composable Commerce.

Blue Reply plays a pivotal role in facilitating the digital transformation of its clients, primarily operating in the sectors of Banking, Credit, Insurance, Manufacturing, Telco & Media, and Retail. Actively supporting them in achieving their digital transformation goals.

The Logistics Unit, where I completed my internship, is responsible for managing the flow of goods and related information of its clients. This unit oversees various organizational, managerial, and strategic activities to ensure the efficient movement of products and the coordination of data associated with them. During my internship, I conducted an in-depth analysis of the transport network for a Blue Reply client. I applied linear programming techniques to explore optimal solutions and compared them with existing theoretical routes using graphical representations to enhance comprehension. As a result, I successfully developed a demo tool for analyzing general transport networks with the purpose of being integrated into a working outbound system.

4.2 Libraries and tools

In this section, I will provide an in-depth overview of the software, libraries, and tools used for the project.

4.2.1 Environment: Python,Conda,jupiterNotebook

The entire development and execution of the work was conducted within a Python-based ecosystem. This choice was made due to Python's versatility and extensive library support. Python's wide range of libraries, including those for data analysis, graph manipulation, and linear programming, made it an ideal choice for the project.

To ensure a well-organized and controlled environment, I employed Conda, a powerful package and environment management tool. Conda allowed me to create isolated environments, making it simple to manage dependencies and ensure consistency in the project. This level of control was crucial in avoiding conflicts between different packages and versions, ensuring the stability of the development environment.

For the development, documentation, and presentation of my work, I relied on Jupyter Notebook. The user-friendly interface of Jupyter Notebook made it an ideal choice, enabling me to easily integrate code, data visualizations, and explanatory text. It significantly enhanced my research and project workflow by allowing for real-time testing, debugging, and the communication of results

4.2.2 Data and Graph: NumPy, Pandas, NetworkX

I didn't have direct access to databases and all the data used were provided in CSV format. To read and manipulate this data, I utilized Pandas [9], a library for working with data frames (i.e., tables). Pandas is built on top of NumPy [10], a powerful numerical library. Both Pandas and NumPy are extensively employed and supported for data analysis in Python. Especially Pandas simplified data cleaning and manipulation, making it a fundamental tool for structuring and exploring datasets.

For graph management, I utilized NetworkX [11], a Python package designed to create, manipulate, and analyze complex networks easily. NetworkX is a widely acclaimed library known for its user-friendly features. In fact, it enables the addition of various attributes to nodes and edges, providing quick and intuitive access, which results in clean and concise Python code. Moreover, NetworkX incorporates numerous classic graph algorithms, such as max flow, min-cost flow, and much more.

4.2.3 Plotting: Matplotlib, Plotly

To visualize the generated graphs, I initially used the basic functionality of NetworkX which interfaces with Matplotlib [12], a widely adopted library to generate static, high-quality plots and charts. Almost a standard in Python for data visualization and data analysis.

One of the most valuable features was the capability to overlay the graph onto a map, enabling the precise visualization of node locations and their connections. However, due to the network's high density, it was often challenging to achieve accurate visualizations of densely clustered portions of the network. While Matplotlib provides interactive features like zooming, these interactions occurred within a static image, which didn't always guarantee satisfactory results.

For these reasons, I eventually switched to Plotly [13], a figure and data representation package that shares similarities with Matplotlib. While Plotly is less established and lacks some functionalities, it has gained increasing popularity for its ability to create clean and interactive data visualizations easily. However, due to its lack of direct integration with NetworkX, it required some additional effort for successful implementation.

4.2.4 Linear programming: PuLP

For modeling and solving linear programming problems, I utilized the PuLP [14]library, a package that extensively utilizes Python notation to effectively translate a description in mathematical language into code. Using PuLP is a common choice because it serves as a global interface for various solvers. Not only a problem can be converted into various other formats, but a licensed solver such as GUROBI or CPLEX can be directly connected by including an API key.

In my case, I used the basic open-source solver COIN-OR (Computational Infrastructure for Operations Research), which utilizes CLP (COIN LP) to solve LP problems based on the simplex method and CBC (COIN Branch & Cut) for ILP and MLP problems.

The key benefit of using PuLP is that, if future challenges require greater efficiency, it is possible to integrate a more powerful solver without needing to modify the code that models the problem.

4.2.5 Frontend: streamlit

Streamlit [15] is a powerful open-source framework designed to simplify the process of creating interactive web applications using Python. It stands out as an excellent choice for developers who want to rapidly create tools and interactive dashboards for data visualization and analysis without delving into the complexities of web development. It empowers users to seamlessly convert Python scripts into interactive web applications with minimal effort, enabling developers to concentrate on the content and functionality of their applications. Streamlit offers a diverse collection of widgets, including sliders, charts, and tables, all presented through a consistent, engaging, and interactive user interface (UI). This ease of use has made Streamlit highly popular among data scientists, engineers, and developers, allowing them to efficiently and effectively create web-based tools and prototypes.

However, it's important to note that the framework is best suited for small projects and prototypes, much like my own. The absence of a clear division between the View, Model, and Controller components does not allow the proper organization of responsibilities across different components and files, making it impractical for managing large-scale projects. The advantages of a well-structured and organized web framework are not within Streamlit's scope.

In my case, my focus was solely on data analysis and problem-solving, making the creation of a structured interface an unnecessary task. Thanks to Streamlit, I was able to enhance my project, which had initially been split across multiple Jupyter notebooks and tests, into a single proof of concept that seamlessly integrated all the work accomplished during my internship.

4.3 Data Setup

4.3.1 Preliminary Data

The initial data concerned a set of facilities, which we refer to as "compounds", and their associated transportation divided into "roadway", "railway", "waterway", and "others" (in case the transport is outsourced).

I had access to both historical data on shipments and theoretical data on the predetermined routes that carriers should take. In addition, there was information on services implemented on individual compounds during the transport process, but as they did not specifically concern the actual movement of goods they were ignored.

The historical data contained aggregated information about previous shipments and provided details such as the average transportation cost (along with its currency) and the average transport duration in days between pairs of compounds.

The theoretical data included instructions regarding all predefined paths established between the starting points and the destinations. These instructions also encompassed all the inters-shipment nodes that the carrier needed to pass through during the transportation, along with estimated lead and dwell times. There are various reasons why the carrier had to divide the delivery route into smaller segments. Firstly, carriers cannot cover extended distances without making stops; or it is possible that the transported goods require inspection during routine checks; or different routes might be managed by distinct transportation companies, driven by various factors, whether economic or political, and so on.

4.3.2 Data cleaning

The initial tasks involved exploring the dataset to assess data consistency and completeness. Notably, several compound coordinates were missing, which required using external APIs to complete them based on addresses. It was also necessary to ensure that all costs were converted to a unique currency, which was not always trivial given the fluctuation of certain currencies in recent years. Moreover, a significant amount of cost and time data was either missing or meaningless. This was because shipment data, such as departure and arrival dates, were entered manually into the system, which could lead to several human errors when loading data. All erroneous data required further research to find more consistent values, and in the worst case, resulted in their removal from the dataset.

This step was crucial and should not be underestimated, as it could lead to impractical solutions for optimization problems. For instance, if an edge has zero cost or, worse, negative cost, it could create loops in algorithms when trying to exploit an unreal resource.

Another issue relates to theoretical routes. Despite being comprehensive, they turned out to contain various routes created ad hoc for specific cases with inconsistent expected times. It was not uncommon to find routes without any logical sense, involving unnecessarily lengthy paths to connect neighboring compounds (figure 2.1). This was due to the necessity of transporting goods through specific inter-transport nodes for the services mentioned above. Without detailed information on the reasons for these choices, these routes had to be ignored.

Figure 4.2: Example of a route with no apparent logical sense, probably created to satisfy a shipment that needed to be revised in a French compound

Instead, I cross-referenced historical data and theoretical routes to obtain a list of feasible arcs, i.e. links between one compound and another for which there is historically or theoretically a contract with a carrier. This means that even if two nodes are geographically close, they may not be directly connected by any edge, and

thus a direct shipment is not possible without passing through at least one other intermediary node.

An alternative approach would have been to calculate a distance matrix between compounds using external APIs. However, this method was only applicable to roadway edges and did not cover waterways, railways, and other transport modes, excluding a substantial portion of the transport network under consideration.

4.3.3 Network visualization

Given the set of compounds and edges that connect them, I was able to plot the entire network to get an idea of its disposition and complexity as shown in figure 4.3

Figure 4.3: Visualisation of the transportation network using Matplotlib. The legend clarifies color associations and their meanings

Why do some waterway edges pass on land? Recall that we are still viewing the transportation network as an operational network; an edge is simply a straight line between two nodes. We know that these two points are connected using some form of maritime transport, but we don't know (or don't care) about specific details. Graphically, we only see straight lines, but in reality, they follow actual streets or navigation routes. It's important to note that we are not losing generality; the true essence of these edges lies within their attributes, such as cost, distance, time, and so on. Even if two edges appear identical on the map, they may contain different data reflecting real-world obstacles, and an algorithm may choose one over the other.

4.4 Implementation

This section shows the code implementation of the flow problems discussed above.

Maximum flow problem

The maximum flow problem was not addressed in the implementation despite its theoretical importance, as it was not considered to be of business relevance during the internship.

Shortest path problem

The shortest path problem has been solved trivially using Networkx's pre-built methods. As previously mentioned, being a widely addressed problem, there are many library methods that solve it very quickly.

Minimum cost flow problem

I also relied on NetworkX's pre-built method for solving the minimum cost flow problem, although it was possible to write the LP modeling using PULP. The choice to use NetworkX is due to its more immediate applicability since the data structure of the network is already implemented within this library, and the fact that it employs the network simplex algorithm to solve the problem more efficiently.

It takes as input a NetworkX directed graph `G` with edge cost and capacity attributes on the edges and demand attributes on the nodes. A negative demand indicates that the node wants to send flow, i.e., it is a starting point or a supply node. A positive demand indicates that the node wants to receive flow, i.e., it is an ending node. A demand of zero means that the flow is conserved, i.e., it is an intermediate node. A flow on the graph satisfies all demands if the net flow into each node equals the demand of that node.

Figure 4.4 shows a fragment of code that assigns demand values to nodes from a data frame that specifies only supply and demand. It ensures that all remaining nodes have zero demand and then calls the `network_simplex` algorithm to find the minimum cost flow, using the specified weight, such as cost or time. If successful, it returns a dictionary representing the amount of flow passing through each arc to satisfy all demands and the total cost of implementing those choices.

```
def solve_minimum_cost_flow(G,df_demands,weight):
    for i,row in df_demands.iterrows():
        G.nodes[row['COMPOUND']]['DEMAND']=row['DEMAND']
    for node in G.nodes:
        if 'DEMAND' not in G.nodes[node]:
            G.nodes[node]['DEMAND']=0
    try:
        flowCost,flowDict= nx.network_simplex(G,weight=weight,capacity='CAPACITY',demand='DEMAND')
    except Exception as e: return None,None,e
    return flowDict,flowCost,"OK"
```

Figure 4.4: Code snippet of `solve_minimum_cost_flow()` method

Using the min-cost flow solution the results can be displayed graphically. Figure 4.5 shows the result of a simple example with three start plants supplying part of the network. The edge's width represents the *quantity* of goods passing through. The edge's color represents the usage ratio, i.e. *quantity/capacity*. It is easy to identify edges that are over-congested or under-utilized, this insight could be exploited to manage relations with carriers and their contracts.

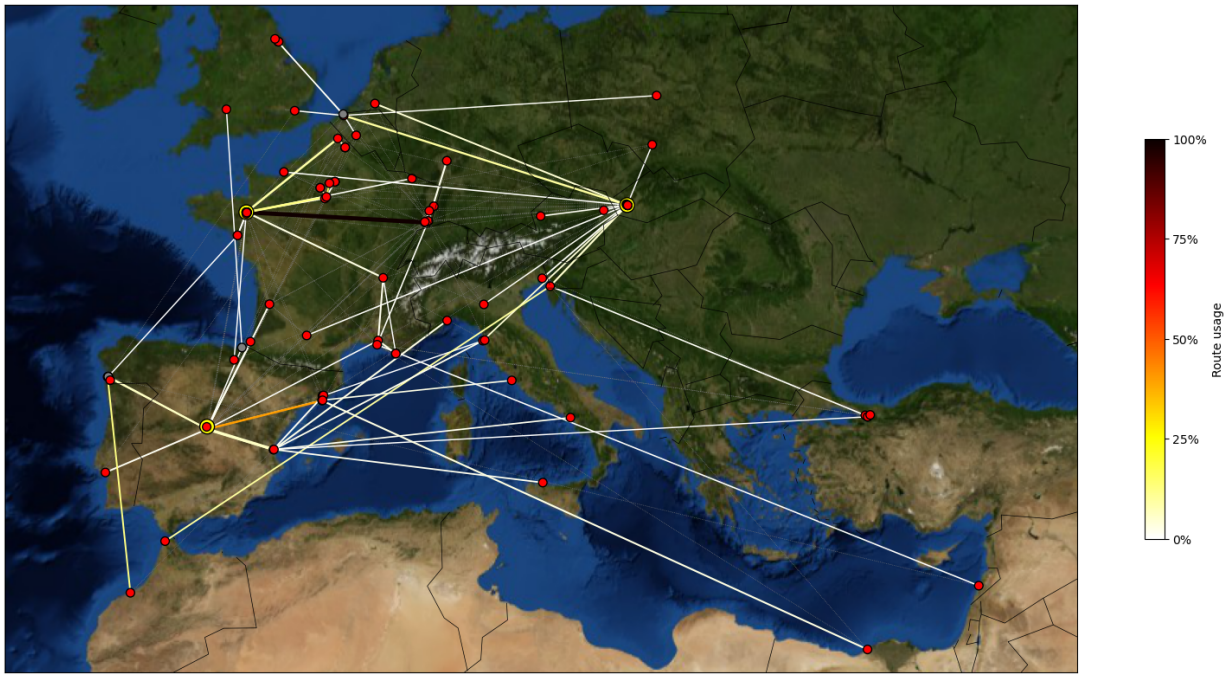


Figure 4.5: Example of minimum cost flow problem results

Multi commodity flow problem

The implementation of the multi-commodity flow problem was not so trivial, since it had to be modeled into an LP problem using PULP, However, the code implementation is straightforward, as it directly follows the mathematical formulation.

Commodities To model the set of commodity K defined by $k = (s_k, t_k, D_k)$ a `Commodity` class is defined (Figure 4.6). A `Commodity` object represents the quantity `demand` that must be shipped from `source` to `dest`. The `name` attribute was added for better comprehension in outputs, but it's not necessary and shouldn't be considered unique as there could be two flows of the same goods that must be considered distinct.

We will assume that the inputs for the multi-commodity flow problem are a graph `G` with capacity and weight attributes on edges, and a list of Commodity `commodities`.

```python
class Commodity:
    def __init__(self, name, source, dest,demand):
        self.name = name
        self.source = source
        self.dest = dest
        self.demand = demand
    def __str__(self):
        return f"[{self.name},{self.source},{self.dest},{self.demand}]"
    def __repr__(self):
        return str(self)
```

Figure 4.6: Commodity class declaration

Objective function

$$\text{minimize} \quad \sum_{k \in K} \sum_{(u,v) \in E} C(u,v) \cdot X_{u,v}^{k}$$

```python
prob=LpProblem('MCFP',LpMinimize)
#OBJECTIVE FUNCTION
sum_cost_var=[]
vars={}
for i, commo in enumerate(commodities):
    for source,dest,data in G.edges(data=True):
        vars[i,source,dest,commo.name]=LpVariable(f'FLOW_{i}_{source}_{dest}_{commo.name}',
                                                  lowBound=0,cat='Integer')
        sum_cost_var.append(data[weight]*vars[i,source,dest,commo.name])
prob+=lpSum(sum_cost_var)
```

Figure 4.7: Objective function implementation in PULP

First, we declare the `LpProblem` variable `prob` and set it as a minimization problem. This variable will contain the objective function, all the constraints, and variables. Next, for each commodity and for each edge in the network, we generate a non-negative integer `LpVariable`, representing the quantity of commodity `"i"` passing through that edge. In addition, we construct the summation for the objective function by adding the newly created variables, each multiplied by the weight of the corresponding edge. Finally, we add the objective function to the problem. Note that, unlike the general formulation for the multi-commodity flow problem, where a cost was related to a specific commodity on a given edge, in this case, it is only related to the edge itself, as no distinction is made between commodities costs.

Capacity Constraint

$$\sum_{k \in K} X_{u,v}^{k} \leq q(u,v) \quad \forall (u,v) \in E$$

```python
#CAPACITY CONSTRAINT
for source,dest,data in G.edges(data=True):
        sum_quantity=[]
        for i, commo in enumerate(commodities):
            sum_quantity.append(vars[i,source,dest,commo.name])
        prob+= lpSum(sum_quantity)<= data["capacity"]
```

Figure 4.8: Capacity constraint implementation in PULP

For each edge in the graph and for each commodity "i" we sum all the variables, i.e. quantities of commodities, passing through that particular edge. It is ensured that the total of these quantities does not exceed the edge's maximum capacity.

Flow Constraint

$$\sum_{(u,v)\in IN(w)} X^k_{u,v} \quad - \sum_{(u,v)\in OUT(w)} X^k_{u,v} = \begin{cases} -D_k & \text{if w is an source for } k \\ D_k & \text{if w is a sink for } k \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in K, \forall w \in V$$

```
#FLOW CONSERVATION CONSTRAINT
for node in G.nodes():
        for i, commo in enumerate(commodities):
                if node==commo.source: tot_flow=-commo.demand
                elif node==commo.dest: tot_flow=commo.demand
                else: tot_flow=0
                sum_quantity_in = [vars[i, source, dest,commo.name] for source, dest in G.in_edges(node)]
                sum_quantity_out = [vars[i, source, dest,commo.name] for source, dest in G.out_edges(node)]
                prob+=lpSum(sum_quantity_in)-lpSum(sum_quantity_out)==tot_flow
```

Figure 4.9: Flow constraint implementation in PULP

Firstly, for each node and commodity, we determine whether the node serves as a source or destination for the given commodity. If it's a source, it must have a total negative flow equal to the commodity's demand. If it's a destination, it should have a total positive flow equal to the demand. Otherwise, as an intermediate node, it must maintain a total flow of zero to ensure flow conservation. Then, we calculate the inflow and outflow for each node and commodity, ensuring that the difference of these flows aligns with the conservation requirements established earlier.

Results Once the problem is modeled we can use `prob.solve()` to call the solver and find a solution to the problem. If feasible, the `LpProblem` will now contain values for the variables. We can use the solution to display the result graphically as done before. Note that the visualization is equal to the min-cost flow problem because it was not possible to draw several lines for each commodity as they would have made the chart confusing and unreadable. However, we can still gain information by looking at the overused edges, as different commodities saturate their capacity as shown in Figure 4.10.

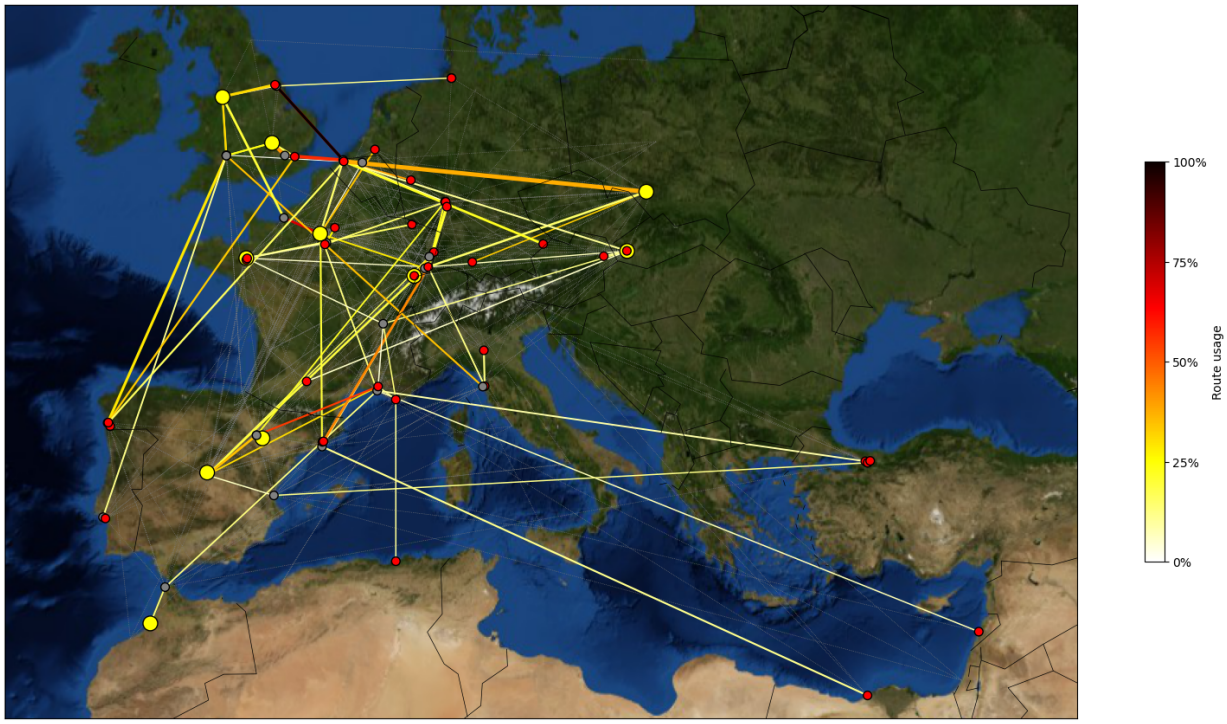


Figure 4.10: Example of multi-commodity flow problem results

4.5 Demo

Despite the work done to obtain clean and complete source data, the approach adopted to develop the tool was as generic as possible. This way, it can load any transport network or portions of it. This means that in the future, if more precise cost estimations are available or if the network is modified, these adjustments can be seamlessly implemented without the need for code modification. The tool is also flexible to the introduction of new parameters, for instance, if we wanted to include the amount of CO2 emitted as an additional weight and information on the transport edges, the tool can handle this and all functionality remains compatible. However, it does not support optimizing on more than one parameter simultaneously, as this would require a different problem modeling.

The tool is divided into three main sections:

- **Home:** Describes the tool and the required CSV file formats that must be loaded. However, the tool will always check the format of the data entered to ensure it can run properly, displaying errors if any exist.

- **Network Analysis:** Allows to load and effectively examine a network structure, visualize the transport network on an interactive map, explore attributes and relationships, and compute shortest paths.

- **Route Scenario:** Allows to load additional information regarding demands to solve minimum cost flow and multi-commodity flow problems, and provides a visualization of the results.

4.5.1 Network Analysis

On the first page (Figure 4.11) is required to upload two CSV files with compounds node and transportation edges. All other pages are disabled until the data are correctly uploaded.

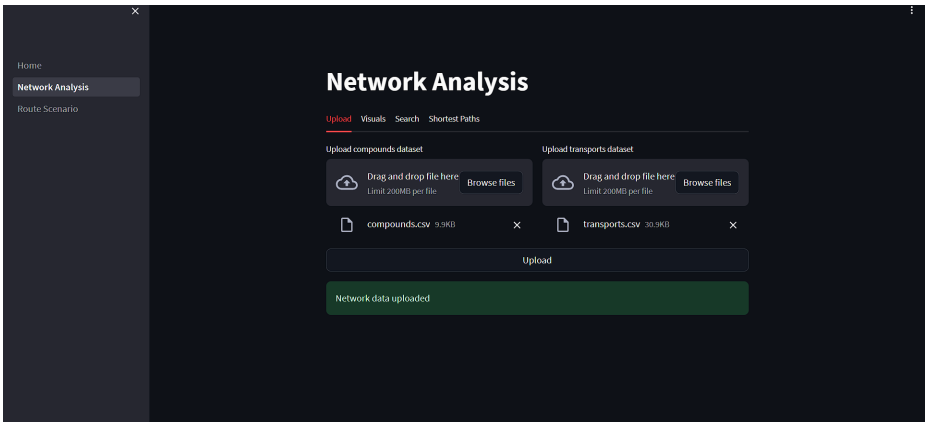


Figure 4.11: Screenshot of the page for uploading CSV files related to the network.

After the successful data loading process, it is possible to switch pages to begin network exploration. The network is visualized in a map (Figure 4.12) similar to those previously displayed but with the distinct advantage of being interactive, achieved by employing Plotly instead of Matplotlib.

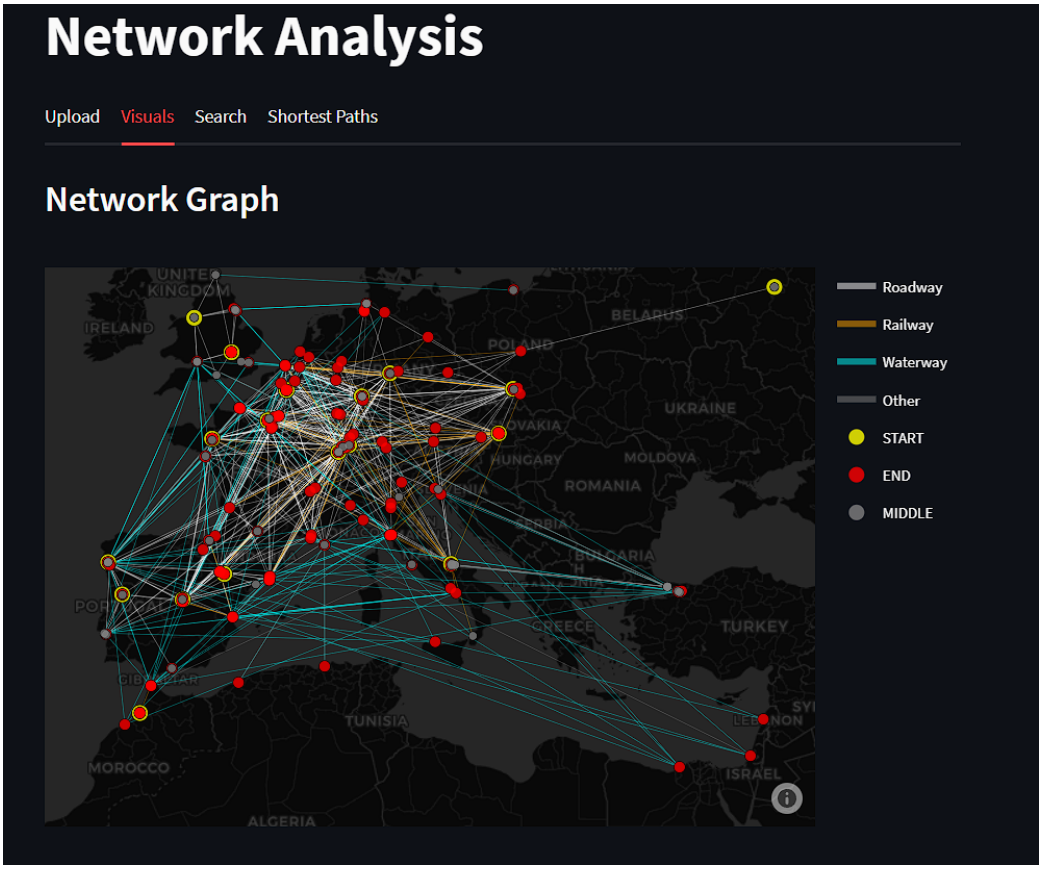
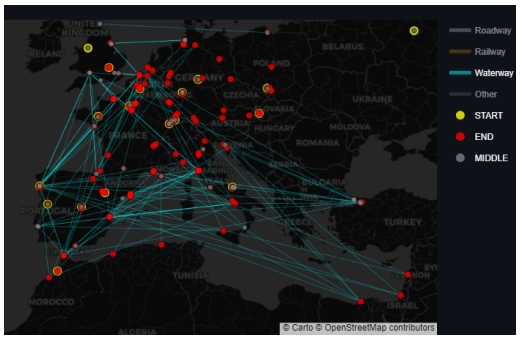
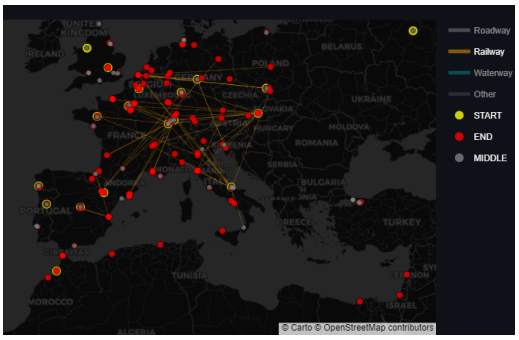


Figure 4.12: Screenshot of the network visualization.

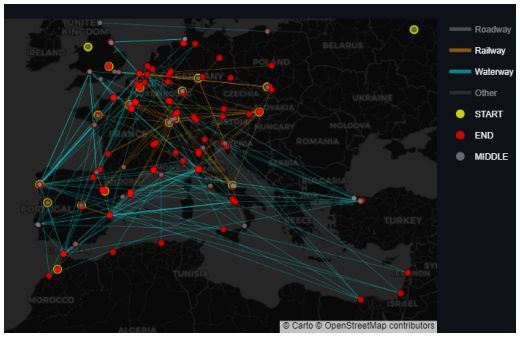
One of the notable features is the ability to selectively focus on specific aspects of the network under examination, as shown in the provided examples in Figure 4.13



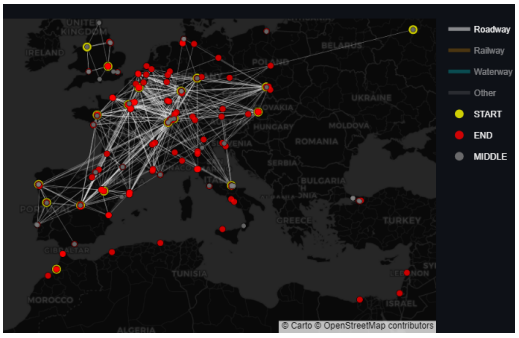
(a) Waterway network



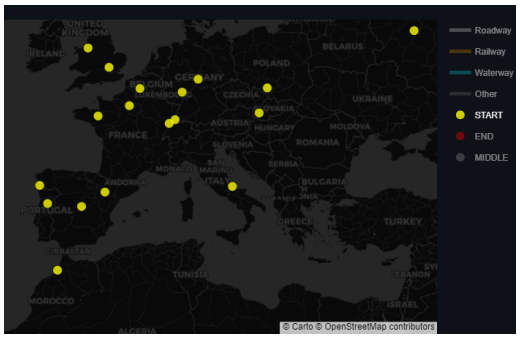
(b) Railway network



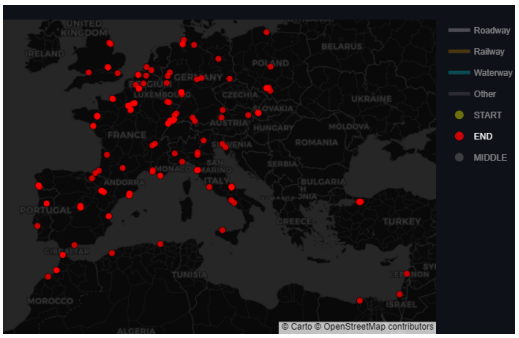
(c) Waterway & Railway network



(d) Road network



(e) Start nodes in the network



(f) End nodes in the network

Figure 4.13: Examples of different aspects of the transport network that can be visualized using the interactive map.

In addition, it is possible to unravel the clustered portions of the network using the zoom, drag, and mouse hover functions, Figure 4.14.

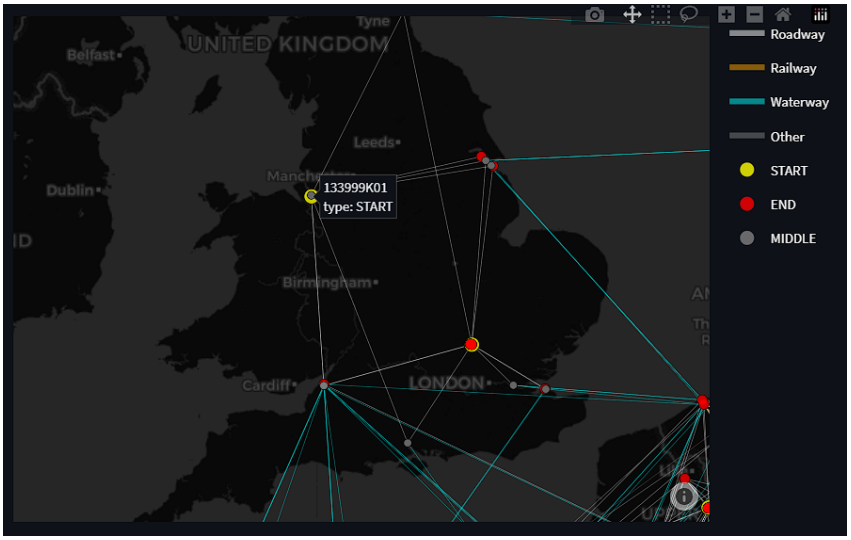


Figure 4.14: Example of a zoomed-in section of the network with mouse-hover details.

On the same page, newly loaded data are presented in the form of interactive tables, accompanied by some details, Figure 4.15. Furthermore, it is possible to visualize edge attributes using bar charts, Figure 4.16.

Compounds

	COMPOUND	TYPE
2	575352H01	MIDDLE
3	259999H01	START
4	575800W10	END
5	029484Y13	END
6	575639W02	MIDDLE
7	575616W01	END
8	029484Y06	END
9	789999W01	START

Total 261
Start 18
Middle 53
End 190

(a) Compounds data frame

Transports

	START	END	TYPE	COST	CAPACITY	TIME	CO2	other data
5	575046F29	013279C01	Other	533	3,819	2	431	479
6	575800W23	575684Q01	Roadway	443	2,012	1	169	3,677
7	575641T01	575900A30	Roadway	133	4,453	4	284	4,944
8	421999W01	575643N07	Roadway	389	4,817	4	145	2,276
9	575657B01	575679X03	Waterway	406	4,108	3	57	4,317
10	575046F66	575046F05	Roadway	318	2,940	2	238	780
11	575800W14	575382G05	Roadway	1,850	5,503	4	307	1,526

Total 801
Roadway 529
Railway 54
Waterway 106
Others 107

(b) Transports edges data frame

Figure 4.15: Screenshots of the network data uploaded.

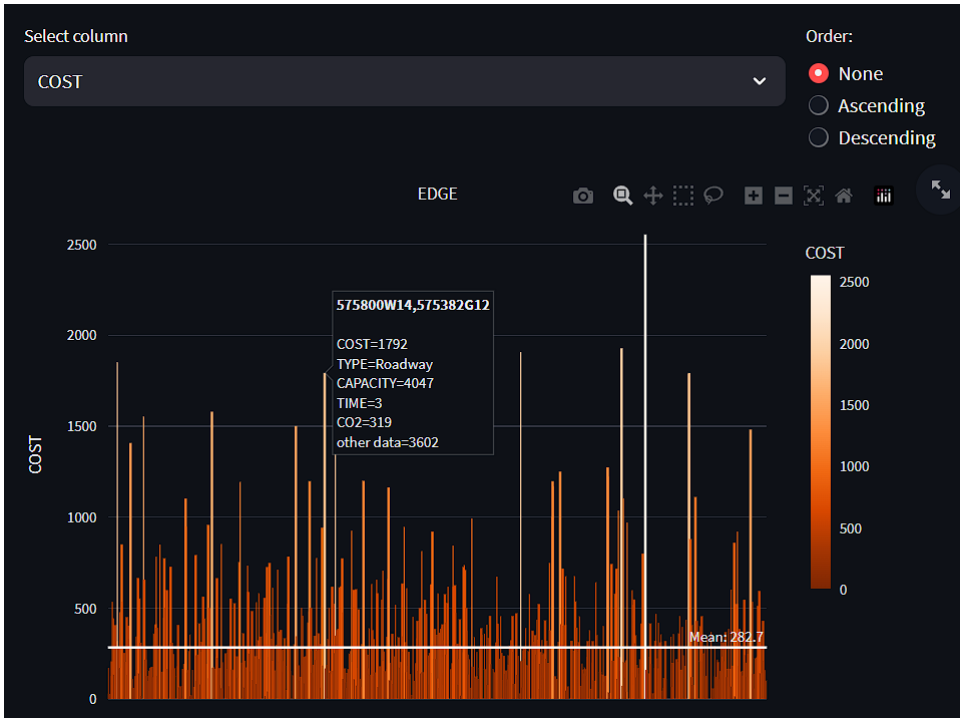


Figure 4.16: Screenshot of cost bar chart, with mouse-hover details.

On the search page, it is possible to search for specific nodes or edges in the transportation network, as shown in Figure 4.17.

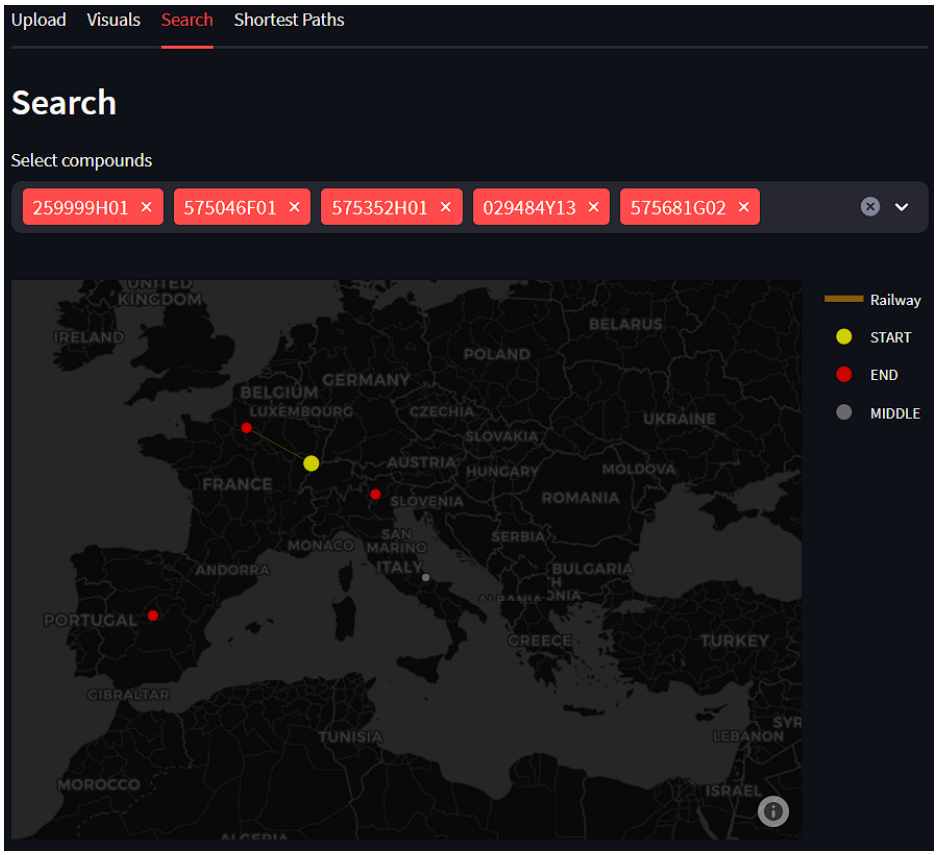
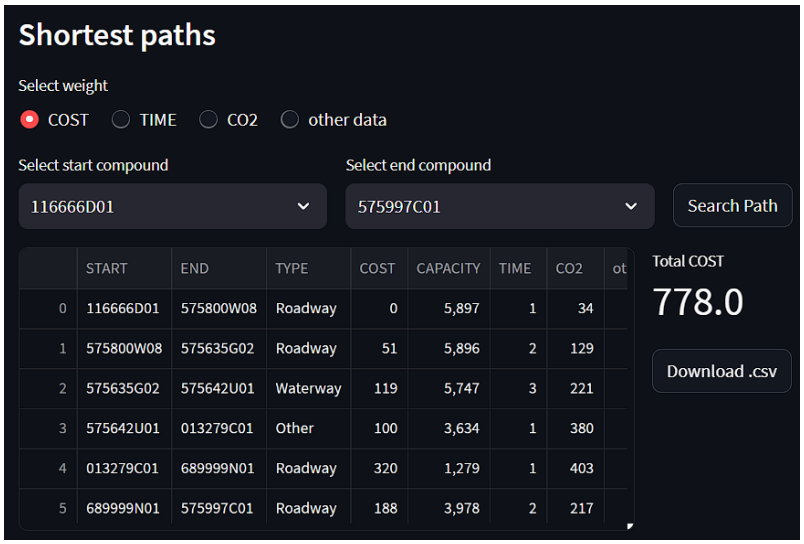


Figure 4.17: Example of search feature.

Lastly, on the shortest path page, it is possible to find the shortest path between a start node and an end node by selecting the edge weight. If a path exists, the results are displayed visually on a map (Figure 4.19) and in a downloadable table (Figure 4.18) containing all the steps required to perform the route.



	START	END	TYPE	COST	CAPACITY	TIME	CO2	ot
0	116666D01	575800W08	Roadway	0	5,897	1	34	
1	575800W08	575635G02	Roadway	51	5,896	2	129	
2	575635G02	575642U01	Waterway	119	5,747	3	221	
3	575642U01	013279C01	Other	100	3,634	1	380	
4	013279C01	689999N01	Roadway	320	1,279	1	403	
5	689999N01	575997C01	Roadway	188	3,978	2	217	

Figure 4.18: Example of shortest path result between two compounds.

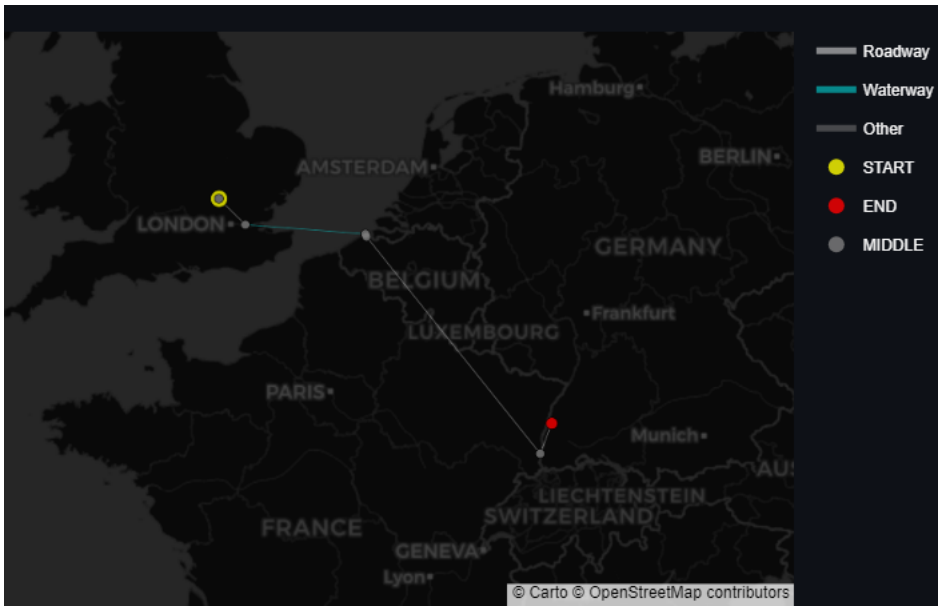


Figure 4.19: Example of shortest path visualisation between two compounds.

4.5.2 Route Scenario

On the first page of the Route Scenario section, it is possible to load a CSV file containing the demands of compounds. This file serves as the input data for solving the minimum cost flow problem, choosing an edge weight. If the solution is feasible, the results are displayed on a map (Figure 4.20), provided in a downloadable table (Figure 4.21), and presented in a usage bar chart(Figure 4.22) to easily identify overused links.

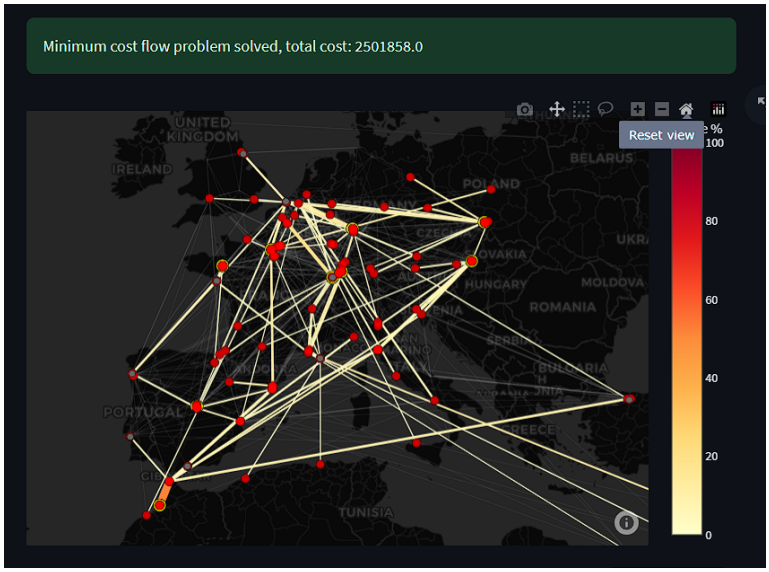
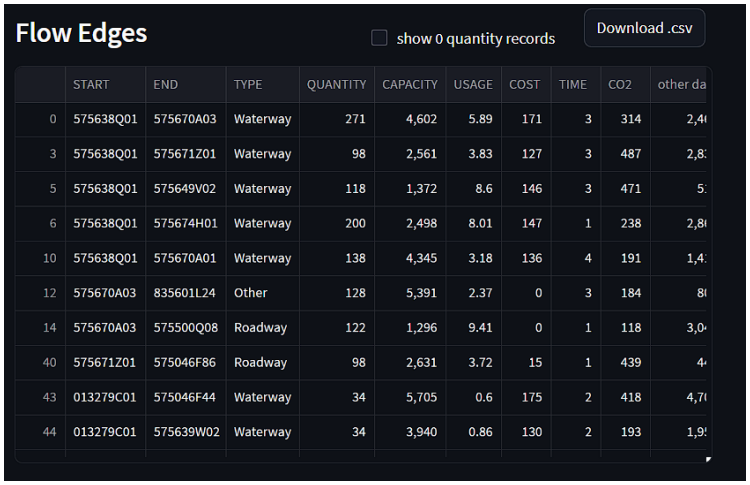


Figure 4.20: Example of min-cost flow visualization.



	START	END	TYPE	QUANTITY	CAPACITY	USAGE	COST	TIME	CO2	other da
0	575638Q01	575670A03	Waterway	271	4,602	5.89	171	3	314	2,4
3	575638Q01	575671Z01	Waterway	98	2,561	3.83	127	3	487	2,8
5	575638Q01	575649V02	Waterway	118	1,372	8.6	146	3	471	5
6	575638Q01	575674H01	Waterway	200	2,498	8.01	147	1	238	2,8
10	575638Q01	575670A01	Waterway	138	4,345	3.18	136	4	191	1,4
12	575670A03	835601L24	Other	128	5,391	2.37	0	3	184	8
14	575670A03	575500Q08	Roadway	122	1,296	9.41	0	1	118	3,0
40	575671Z01	575046F86	Roadway	98	2,631	3.72	15	1	439	4
43	013279C01	575046F44	Waterway	34	5,705	0.6	175	2	418	4,7
44	013279C01	575639W02	Waterway	34	3,940	0.86	130	2	193	1,9

Figure 4.21: Example of min-cost flow results.

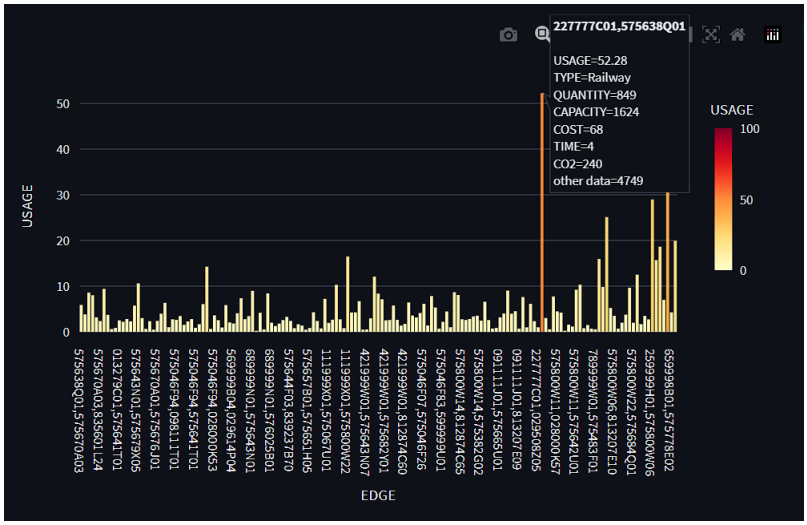


Figure 4.22: Example of min-cost flow usage bar chart, with mouse-hover details.

On the second page, it is possible to load a CSV file containing the demands of different commodities orders to solve the multi-commodity flow problem, choosing an edge weight. The results are shown similarly to the min-cost problem, using a graphical representation (Figure 4.23), and displayed in a bar chart (Figure 4.24)

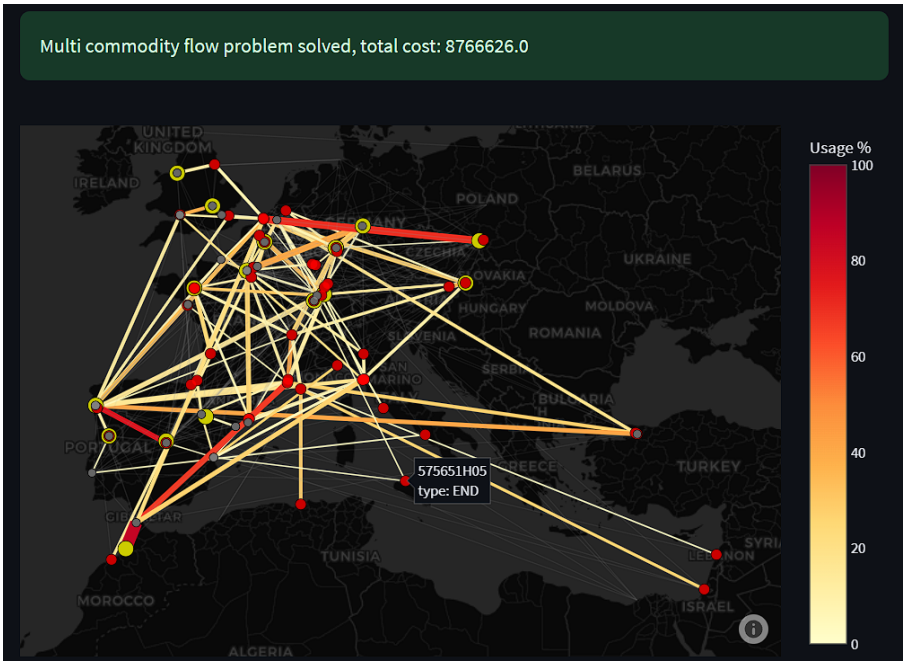


Figure 4.23: Example of multi-commodity flow visualization.

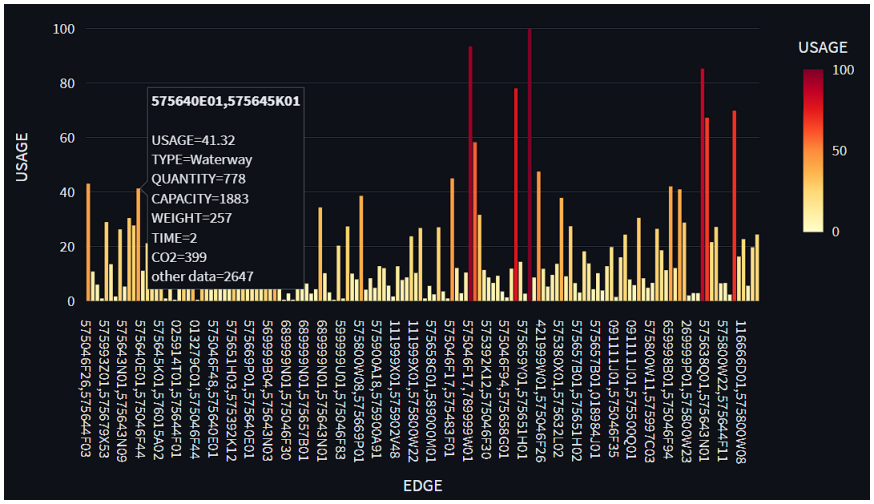
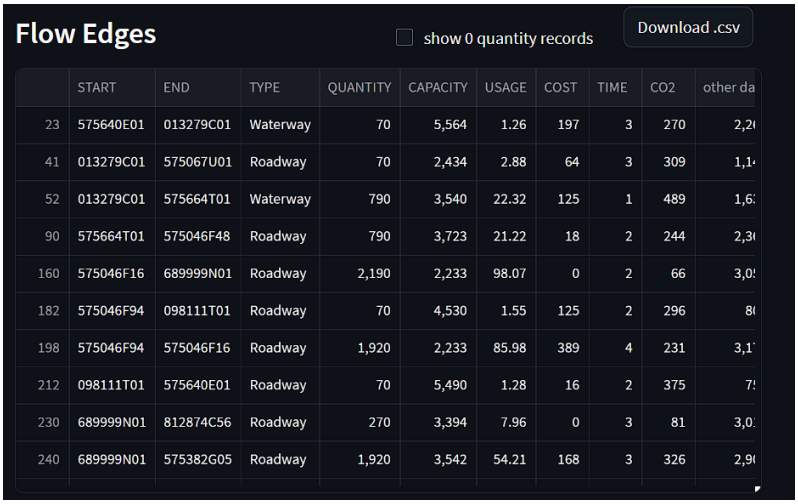


Figure 4.24: Example of multi-commodity flow usage bar chart, with mouse-hover details.

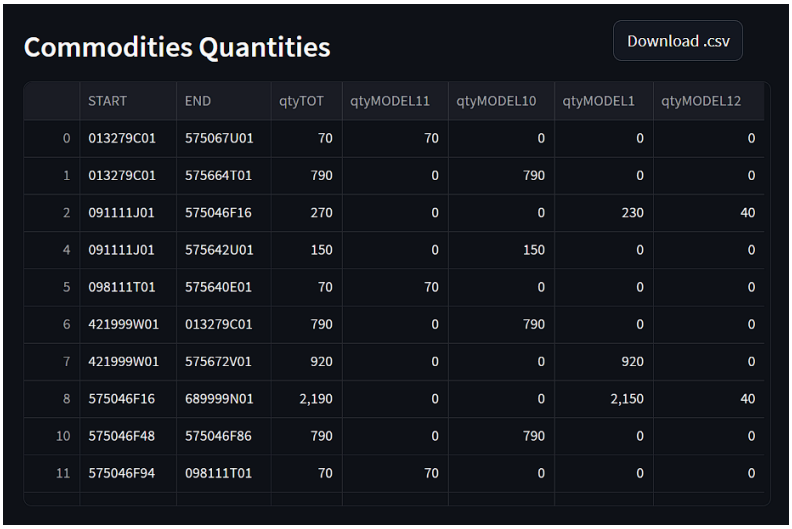
Additionally, three downloadable tables are available: one displaying the total quantities transported through each edge (Figure 4.25), another showing the quantities of each commodity for each edge (Figure 4.26), and a third detailing all the routes (Figure 4.27), composed of the individual steps required to transport a commodity from its source to the destination.



Flow Edges show 0 quantity records Download .csv

	START	END	TYPE	QUANTITY	CAPACITY	USAGE	COST	TIME	CO2	other da
23	575640E01	013279C01	Waterway	70	5,564	1.26	197	3	270	2,2(
41	013279C01	575067U01	Roadway	70	2,434	2.88	64	3	309	1,1
52	013279C01	575664T01	Waterway	790	3,540	22.32	125	1	489	1,6
90	575664T01	575046F48	Roadway	790	3,723	21.22	18	2	244	2,3(
160	575046F16	689999N01	Roadway	2,190	2,233	98.07	0	2	66	3,0
182	575046F94	098111T01	Roadway	70	4,530	1.55	125	2	296	8(
198	575046F94	575046F16	Roadway	1,920	2,233	85.98	389	4	231	3,1
212	098111T01	575640E01	Roadway	70	5,490	1.28	16	2	375	7
230	689999N01	812874C56	Roadway	270	3,394	7.96	0	3	81	3,0
240	689999N01	575382G05	Roadway	1,920	3,542	54.21	168	3	326	2,9(

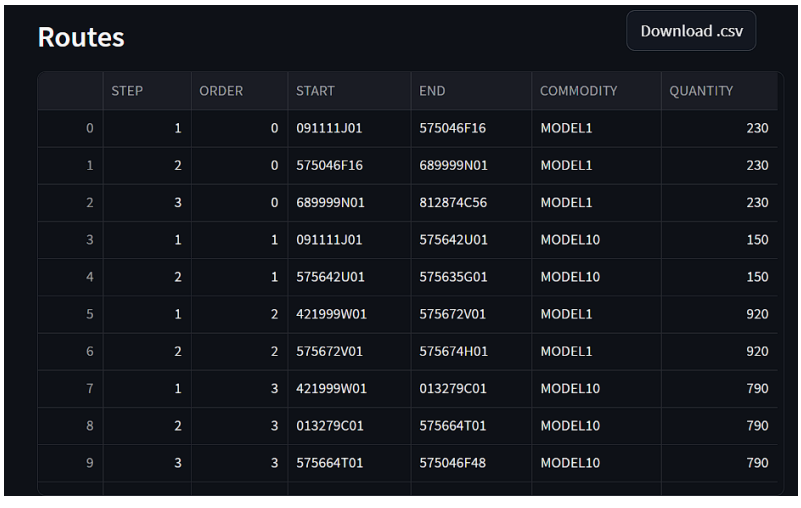
Figure 4.25: Table displaying the total quantities per edge.



Commodities Quantities Download .csv

	START	END	qtyTOT	qtyMODEL11	qtyMODEL10	qtyMODEL1	qtyMODEL12
0	013279C01	575067U01	70	70	0	0	0
1	013279C01	575664T01	790	0	790	0	0
2	091111J01	575046F16	270	0	0	230	40
4	091111J01	575642U01	150	0	150	0	0
5	098111T01	575640E01	70	70	0	0	0
6	421999W01	013279C01	790	0	790	0	0
7	421999W01	575672V01	920	0	0	920	0
8	575046F16	689999N01	2,190	0	0	2,150	40
10	575046F48	575046F86	790	0	790	0	0
11	575046F94	098111T01	70	70	0	0	0

Figure 4.26: Table displaying each commodity quantity per edge,



Routes Download .csv

	STEP	ORDER	START	END	COMMODITY	QUANTITY
0	1	0	091111J01	575046F16	MODEL1	230
1	2	0	575046F16	689999N01	MODEL1	230
2	3	0	689999N01	812874C56	MODEL1	230
3	1	1	091111J01	575642U01	MODEL10	150
4	2	1	575642U01	575635G01	MODEL10	150
5	1	2	421999W01	575672V01	MODEL1	920
6	2	2	575672V01	575674H01	MODEL1	920
7	1	3	421999W01	013279C01	MODEL10	790
8	2	3	013279C01	575664T01	MODEL10	790
9	3	3	575664T01	575046F48	MODEL10	790

Figure 4.27: Table displaying routes for each commodity.

4.6 Conclusions and further improvements

In conclusion, we have explored the significance of logistics, with a specific focus on transportation networks. We have delved into the realm of operations research, a scientific discipline dedicated to problem-solving and decision-making. Furthermore, we have acquired a fundamental understanding of linear programming in the context of formalizing problems, particularly related to flow networks, such as minimum-cost flow and multi-commodity flow problems.

This knowledge has enabled the realization of a tool whose purpose is not to be functionally adequate for real use, but rather to begin an exploratory examination of a potential linear programming approach to the problem.

I have not initiated a complete and usable project, but I have brought a business interest to the topic which could lead to a more structured and serious development of a transport optimisation software using this approach.

In particular, it is important to conduct a more in-depth analysis of the complexity and computational time required. In my case, utilizing a sub-optimal solver and a standard laptop, I always managed to get the results in an acceptable time. with larger-scale problems, the time required for the solving process may become prohibitively long. Hence, it is crucial to consider whether the adoption of superior solvers and higher-performance hardware can sufficiently mitigate this issue.

Moreover, there is potential for further improvement in enhancing the model's accuracy by carefully considering specific problem details. For instance, the cost function is currently modeled in a rather standard manner, but a deeper understanding of contracts and relationships with various carriers could result in a more accurate model. This enhanced model may include fixed costs and costs proportional to the volume of products transported, among other potential factors to be taken into consideration.

Lastly, it is crucial to be aware of the limitations of this approach. The numerical results obtained should not be used blindly, as they are not meant to be entirely accurate, but rather as indications and support for human decision-making that must consider a multitude of other factors. In fact, Logistics is a field that should not be viewed as static, and the results obtained cannot remain unchanged indefinitely. The optimization of logistics is a dynamic process that continuously adapts to address future challenges. This type of optimization may be necessary on an annual, weekly, or even real-time basis, requiring different approaches.

The results must always be interpreted with awareness of the general limitations of solving problem models. The results may lack precision due to potential errors in the data, approximations, or the omission of unforeseen factors in the modeling process. As the philosopher Alfred Korzybski remarked, *"the map is not the territory"*. No matter how accurate the representation of a problem is, it is still a representation and not the problem itself.

Bibliography

[1] MD Sarder. *Logistics Transportation Systems*. Elsevier, 2020.

[2] Council of Supply Chain Management Professionals. Scm definitions and glossary of terms, 2019.

[3] Gianpaolo Ghiani, Gilbert Laporte, and Roberto Musmanno. *Introduction to logistics systems planning and control*. Wiley, 2004.

[4] Robert J Vanderbei. *Linear programming : foundations and extensions*. Springer, 2008.

[5] Saul I. Gass and Arjang A. Assad. *An Annotated Timeline of Operations Research: An Informal History*. Editorial: Dordrecht: Kluwer Academic Publishers, 2005.

[6] Alex Elkjær Vasegaard. Why operations research is awesome an introduction, 08 2021.

[7] Frederick S Hillier and Gerald J Lieberman. *Introduction to operations research*. Mcgraw-Hill, 2010.

[8] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows : theory, algorithms, and applications*. Prentice-Hall, 1993.

[9] pandas documentation. Available at `https://pandas.pydata.org/docs/`.

[10] NumPy documentation. Available at `https://numpy.org/doc/`.

[11] NetworkX documentation. Available at `https://networkx.org/`.

[12] Matplotlib documentation. Available at `https://matplotlib.org/stable/index.html`.

[13] Plotly documentation. Available at `https://plotly.com/python/`.

[14] PuLP documentation. Available at `https://coin-or.github.io/pulp/`.

[15] Streamlit documentation. Available at `https://docs.streamlit.io/`.

Dedicato a mia madre