WOOD2WOOD

A Wood-to-Wood Cascade Upcycling Valorisation Approach

» Deliverable 13.3

Supply chain optimisation tool

WP No.	13
GA Number	101138789
Project Full Title	Wood2Wood Project
Project Acronym	W2W
Funding Scheme	Horizon RIA
Start Date	01/01/2024
Duration	48 months
Coordinator	Dr. Angelos Amditis, Research and Development Director, ICCS
Project Website	https://www.wood2wood-project.eu/

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or HADEA. Neither the European Union nor the granting authority can be held responsible for them.

PROJECT INFORMATION

Version	Date	Author	Description of changes
0.1	22/03/2024	Abhimanyu Chakravorty	Template released
0.2	30/05/2024	UPV	Version for internal review
0.3	09/06/2024	EBOS, DRAXIS	Version after internal review
0.4	13/06/2024	ICCS	Version after plagiarism check
1.0	13/06/2024	UPV	Final submitted version
2.0			Revised submitted version (if required)

REVISION AND HISTORY CHART

Dissemination Level		
PU	Public	X
RE	Restricted to a group specified by the consortium	
CO	Confidential, only for members of the consortium (including the European Commission Services)	

Legal Disclaimer

TABLE OF CONTENTS

GLOSSARY OF ACRONYMS

Acronym	Extended Definition
API	Application Programming Interface
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CBC	Coin-or Branch and Cut
CSV	Comma Separated Values
CSS	Cascading Style Sheets
GA	Genetic Algorithm
JSON	JavaScript Object Notation
MAPE	Mean Absolute Percentage Error
SCO	Supply Chain Optimization
SCSS	Sassy CSS
W2W	Wood2Wood

List of Tables

List of Figures

EXECUTIVE SUMMARY

The Wood2Wood project has developed the first version of a SCO (Supply Chain Optimisation Toolkit) to help companies in the wood industry create more sustainable and efficient supply chains. This web-based platform addresses the urgent need for circular wood practices in the construction and furniture industries across Europe.

The toolkit provides five specialised tools that solve common supply chain problems. The Facility Location Problem tool finds the best locations for warehouses and distribution centers. The Inventory Optimisation tool calculates optimal stock levels and reorder points for products. The Vehicle Routing Problem tool plans the most efficient delivery routes for trucks and vehicles. The Network Flow Optimisation tool determines the best paths for materials to move from factories to customers. The Supply Demand Matching tool forecasts future demand based on historical sales data.

Users interact with the platform through a simple process: they upload their data in CSV files, select their preferred solving method, and download the results. The system runs on modern web technology with a React front end and Python backend services. Each optimisation problem operates as an independent service that can run in Docker containers.

Testing shows that the algorithms work correctly and produce reasonable results. The facility location tool performed well compared to commercial software, especially for larger problems. The inventory optimisation provides clear cost analysis, while the network flow tool uses proven mathematical methods. The forecasting tool achieved good accuracy when tested against historical data.

However, Version 1 has significant limitations that restrict its readiness for widespread use. The platform lacks data validation, meaning it cannot check if uploaded files contain correct information. There are no template files available for users to download as formatting guides. The system needs better error handling to provide helpful messages when problems occur. Some tools offer only one solving method instead of multiple options. Most importantly, the toolkit has not been tested with real companies or use cases, so its practical effectiveness remains unknown.

The development team plans to address these limitations in Version 2 by collecting feedback from actual users, adding data validation features, providing downloadable templates, implementing more algorithms, and improving the user interface. The project will continue through repeated cycles of testing and improvement until it meets the specific needs of wood industry companies seeking to build more sustainable supply chains.

1. INTRODUCTION

1.1. PURPOSE OF THE TOOL

The Supply Chain Optimization Toolkit is a digital tool designed to help companies manage the recycling of wood materials more effectively throughout their supply chains. The tool creates virtual copies of company supply networks that include secondary material providers, recyclers, transport companies, processors, and end users.

Companies can use this system to design, test, and improve their supply chain networks for recycled materials. The toolkit uses computer algorithms and data analysis to run different scenarios, helping businesses find the most efficient and cost-effective ways to organise their supply chain operations. This covers the entire process from collecting waste materials to processing and distributing the final recycled products.

The tool helps companies decide the best inventory levels of recycled materials to keep at different points in their supply chain while meeting customer needs and keeping costs low. It also helps with transport planning by choosing the right shipping methods and carriers and managing delivery routes and schedules.

This tool helps them spot areas where they can make improvements and continue to make their recycling processes work better. The tool connects to company information systems to track and monitor the supply chain, giving updates on where recycled materials are and what condition they are in at every step of the process.

This toolkit is part of a larger project focused on creating better ways to recycle construction and demolition wood waste, turning it into useful new products while reducing environmental impact and waste sent to landfills.

1.2. SCOPE OF VERSION 1

The first version of the Supply Chain Optimization Toolkit is a web-based platform that helps companies solve five main types of supply chain problems. The toolkit works through a simple process where users upload their data in CSV format, choose their preferred solving method, and get results they can download.

However, Version 1 has several important limitations. The platform does not check if the data users upload is correct or complete, which means it might produce wrong results if given bad data. There are no template files available for download to help users format their data properly. The system also lacks proper error handling, so users might not get helpful messages when something goes wrong.

Some optimisation problems only offer one solving method instead of multiple options. The vehicle routing feature, for example, only uses genetic algorithms, while other similar tools might offer several different approaches. The toolkit has not yet been tested with real companies or use cases, so the developers do not know how well it works in practice or what changes might be needed.

The system does not include a Transport Mode Selection module, even though this feature was planned at the beginning. During the requirements collection process, the use cases showed that users were not interested in this function. The project team also could not create a clear definition of how this module should work or what information it would need as inputs and what results it would produce as outputs for the situations where users normally operate.

The platform is also missing some basic features that would make it easier to use. Users cannot download example files to see how their data should be formatted, and there is no built-in help or guidance for people who are new to optimisation problems.

The tool is a basic starting system that can solve the problems that use cases found important during the requirements collection process. It has a simple interface and basic coverage that is expected to get better in future versions.

1.3. Relation to Project and W13 Objectives

The Supply Chain Optimisation Toolkit connects directly to the broader project goals of improving how companies handle construction and demolition wood waste. The project works to create better ways to recycle this type of waste and turn it into useful new products while reducing environmental impact and the amount of waste sent to landfills.

This toolkit supports these project goals by helping companies in the wood recycling value chain work more efficiently. When recycling companies, manufacturers, and waste management organisations can plan their operations better, they can process more wood waste and create more recycled products. This directly helps the project's mission to increase the use of secondary wood materials instead of sending waste to landfills.

The tool addresses specific problems that project partners identified during the requirements collection process. Companies involved in wood waste recycling told the project team what challenges they face in managing their supply chains. The toolkit was built to solve these real problems rather than creating features that might not be useful.

As part of Work Package 13 objectives, this toolkit represents the first step in developing digital tools that support circular economy practices in the wood industry. The work package focuses on creating digital solutions that help companies track materials, plan operations, and make better decisions about recycling processes.

The current version provides basic functionality that covers the most important problems identified by industry partners. Future versions will add more features and improve the user experience based on feedback from real-world testing. This gradual development approach ensures that the final tool will meet the actual needs of companies working in wood waste recycling.

2. PLATFORM OVERVIEW

2.1. Core Concepts and Goals

The supply chain optimisation toolkit is a digital tool designed to help companies manage wood recycling processes more effectively. The main idea behind this toolkit is to create virtual copies of real supply chains so companies can test different approaches before making changes in the real world.

The toolkit works by collecting data from different parts of the supply chain and using computer algorithms to find better ways to move materials from one place to another. Companies can use this tool to see how changes might affect their costs, delivery times, and overall performance before they make those changes.

One of the key goals is to help companies figure out the best places to store recycled wood materials and how much to keep at each location. This helps balance having enough materials available when needed while not spending too much money on storage. The toolkit also helps plan the best routes for trucks and other vehicles to move materials between different locations. The toolkit also creates reports that show how well the supply chain is working and where improvements can be made.

Another important goal is to reduce waste and make better use of recycled materials. By finding more efficient ways to collect, process, and distribute recycled wood, companies can save money while also helping the environment.

2.2. Target Users and Use Cases

The main users of this toolkit are companies involved in collecting and processing recycled wood materials. Recycling companies that pick-up wood waste from construction sites or old furniture can use the tool to plan better routes for their trucks and decide where to build processing centers.

Manufacturing companies that make new products from recycled wood will also find this toolkit useful. These companies need to know they can get enough recycled materials when they need them. The toolkit helps them plan and work with suppliers to make sure materials arrive on time.

Supply chain managers at large companies are another key group of users. These people are responsible for making sure materials flow smoothly from suppliers to factories to customers. The toolkit gives them a way to test new ideas and see potential problems before they happen.

Public organisations that manage waste collection in cities and towns can also benefit from this tool. They can use it to plan more efficient collection routes and decide where to locate recycling facilities to best serve their communities.

The toolkit is particularly useful when companies want to expand their operations or when they face unexpected changes like new environmental rules or shifts in demand for recycled materials. Instead of making expensive mistakes, they can use the digital tool to explore different options safely.

Small and medium-sized companies will find the toolkit especially valuable because they often do not have large teams to analyse complex supply chain problems. The tool gives them access to the same type of analysis that big companies use, helping them compete more effectively in the recycling market.

The interface workflow in this first version works as shown in Figure 1. First, users pick an algorithm to solve their problem. They provide data in CSV format. Then the data goes to the API backend. The algorithms run there. The system returns the solutions. Users can get the results in both JSON and CSV formats.

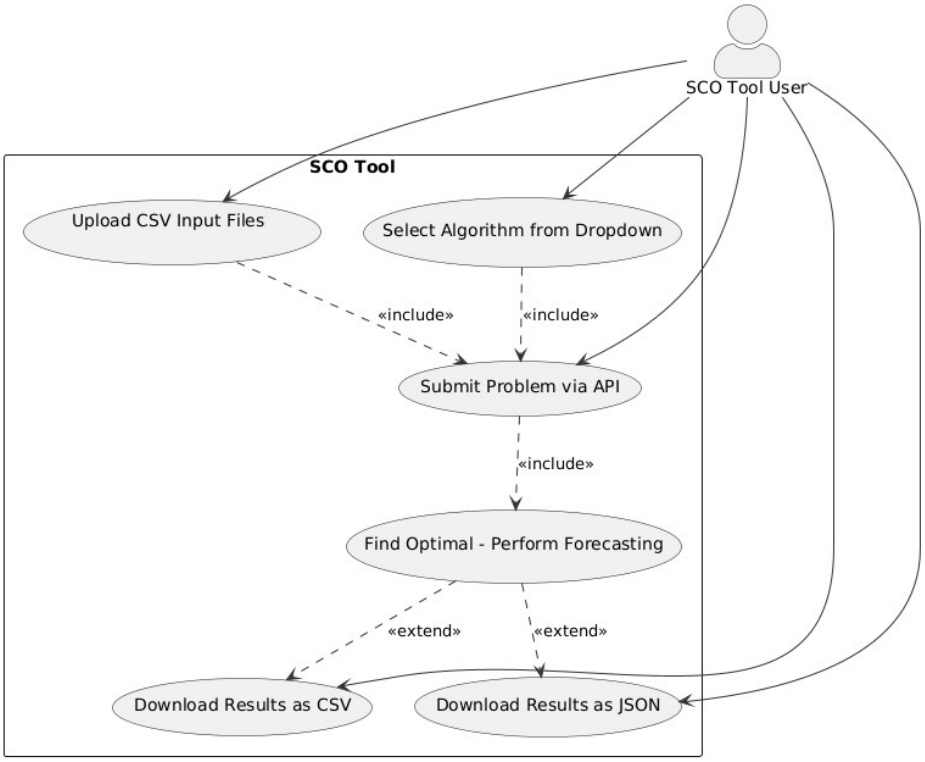

Figure 1 Use Cases Diagram

2.3. KEY FEATURES (V1)

The toolkit handles facility location problems by finding the best places to build warehouses while keeping costs low. It uses two different solving methods, a mathematical approach called CBC and a mixed approach that combines genetic **algorithms** with CBC. This helps companies decide where to put their distribution centers to serve customers most efficiently.

For inventory management, the platform helps businesses figure out how much stock to keep and when to reorder products. It looks at historical demand data, lead times, and costs to calculate the best inventory levels. The system uses genetic algorithms, and a method called BFGS to find solutions that balance holding costs with the risk of running out of stock.

The vehicle routing part of the toolkit plans the best routes for delivery trucks. It considers where customers are located, how much they need, and when deliveries must happen. Right now, this feature only uses genetic algorithms to solve routing problems, which means it has fewer options than some other parts of the toolkit.

Network flow optimisation helps companies plan how materials should move through their supply chain from factories to warehouses to customers. This feature uses exact mathematical methods to find the best flow paths and keeps transportation costs as low as possible.

The demand forecasting tool looks at past sales data to predict future demand. For now, it offers two forecasting methods, the Theta method and Holt Winters method, so users can pick the approach that works best for their type of data.

The toolkit runs on modern web technology with a React-based front end and Python backend services. Each optimisation problem has its own dedicated service, and the whole system can run in Docker containers. Users interact with the platform through a clean interface built with PrimeReact components.

3. FUNCTIONALITIES AND REQUIREMENTS

3.1. REQUIREMENTS

3.1.1. Facility Location Problem: Product Backlog

Table 1 Facility Location Problem Backlog

ID	Priority	Title	Description	Acceptance Criteria
0	High	Core Optimisation Engine Development	Minimise the combined costs of transportation, facility operation, and inventory, ensuring the most efficient flow of wood secondary materials from source to their consumers.	The system can read input data with plants, warehouses, regions, products, costs, and limits. The system can find solutions that minimise all costs (building, operating, handling, and transport). The system can handle both small and large problems. The system can output results in a clear format.

1	High	Simulation	Possibility of simulation to evaluate different facility locations and their impact on supply chain efficiency.	The system can run "what-if" scenarios with different warehouse locations. The system shows cost or references for comparison between different location scenarios. Results show which regions are served by which warehouses in each scenario.
2	Medium	Add two different ways to solve the problem - pure mathematical (CBC) and hybrid (GA+CBC) methods.	Add two different ways to solve the problem - pure mathematical (CBC) and hybrid (GA+CBC) methods for scalability.	CBC solver works correctly for finding optimal solutions. GA+CBC hybrid solver works for larger problems. User can choose which method to use. Both methods produce the same output format.
3	Medium	API Backend Encapsulation	Create a system to handle different input formats for the problem data.	API validates all input data and returns clear error messages for invalid data. API provides consistent output format regardless of the solver used (CBC or GA+CBC). Documentation includes examples of all API calls and data formats.
4	Medium	Front End Development	Create a Front End.	User Interface that loads the input tables from csv format files and outputs the results in an appropriately formatted table, with download buttons that work properly.

5	Low	Documentation and Training Materials	Create help guides and training for users.	Has simple instructions for basic use. Has detailed technical documentation. Includes examples of common use cases. Has training materials for new users.

3.1.3. Inventory Optimisation: Product Backlog

Table 2 Inventory Optimization Problem Backlog

ID	Priority	Title	Description	Acceptance Criteria
0	High	Core Optimization Engine Development	Simulate the demand data and supply times from provided historical data. Then use profit data, warehouse capacity and expiration data to obtain the holding, stockout, ordering and expiration costs along with the reorder points and optimal order quantities per product and warehouse combination and the corresponding service levels.	System can read input data with demand, supply times, profit data, warehouse capacity and expiration data. System can find solutions that minimize all costs. System can output results in a clear format.
1	High	Simulation	Possibility of simulation to evaluate different or changing profit data, facility capacities, expiration dates, demands, or supply times.	System can run "what-if" scenarios with different warehouse data. System shows cost and operative variables conditions for the optimal solution.
2	Medium	API Backend Encapsulation	Create a system to handle different input formats for the problem data.	API validates all input data and returns clear error messages for invalid data. API provides consistent output format.

				Documentation includes examples of all API calls and data formats.
3	Medium	Front End Development	Create a Front End.	User Interface that loads the input tables from csv format files and outputs the results in an appropriately formatted table, with download buttons.
4	Low	Documentation and Training Materials	Create help guides and training for users.	Has simple instructions for basic use. Has detailed technical documentation. Includes examples of common use cases. Has training materials for new users.

3.1.3. Vehicle Routing Problem: Product Backlog

Table 3 Vehicle Routing Problem Backlog

ID	Priority	Title	Description	Acceptance Criteria
0	High	Core Optimisation Engine Development	The input data are the nodes with their demand, capacities, as well as the different types of costs. The algorithm gets the best possible flows and quantities at each node.	The system can read input data with demand, times, capacity and cost information. The system can find solutions that minimise all costs. The system can output results in a clear format.
1	High	Simulation	Possibility of simulation to evaluate different or changing demand nodes, information, or fleet information. The tool finds the best way to make products at factories, ship them through warehouses,	The system can run "what-if" scenarios with different node data. The system shows the cost and operational variables conditions for the optimal solution.

			meet customer demand, keep total costs low and meet the capacity constraints.	
2	Medium	API Backend Encapsulation	Create a system to handle different input formats for the problem data.	API validates all input data and returns clear error messages for invalid data. API provides consistent output format. Documentation includes examples of all API calls and data formats.
3	Medium	Front End Development	Create a Front End.	User Interface that loads the input tables from csv format files and outputs the results in an appropriately formatted table, with download buttons.
4	Low	Documentation and Training Materials	Create help guides and training for users.	Has simple instructions for basic use. Has detailed technical documentation. Includes examples of common use cases, Has training materials for new users.

3.1.4. Network Flow Optimisation: Product Backlog

Table 4 Network Flow Optimisation Problem Backlog

ID	Priority	Title	Description	Acceptance Criteria
0	High	Core Optimization Engine Development	The network flow method needs specific data about locations, transport links, demand, and production.	The system can read input data with demand, times, location, and fleet information.

				The system can find solutions that minimise all costs. The system can output results in a clear format.
1	High	Simulation	Possibility of simulation to evaluate different or changing demand nodes information or costs information.	The system can run "what-if" scenarios with different node data. System shows cost and operative variables conditions for the optimal solution.
2	Medium	API Backend Encapsulation	Create a system to handle different input formats for the problem data.	API validates all input data and returns clear error messages for invalid data. API provides consistent output format. Documentation includes examples of all API calls and data formats.
3	Medium	Front End Development	Create a Front End.	User Interface that loads the input tables from csv format files and outputs the results in an appropriately formatted table, with download buttons.
4	Low	Documentation and Training Materials	Create help guides and training for users.	Has simple instructions for basic use. Has detailed technical documentation. Includes examples of common use cases. Has training materials for new users.

3.1.5. Supply Demand Matching: Product Backlog

Table 5 Supply Demand Matching Problem Backlog

ID	Priority	Title	Description	Acceptance Criteria
0	High	Core Engine Development	It is a forecasting tool so it needs historical demand data with dates and a forecasting horizon.	The system can read input data with demand and times. The system can find forecasted values for a given time horizon. The system can output results in a clear format
1	High	Simulation	Possibility of simulation to evaluate different or changing demand information.	The system can run "what-if" scenarios with different demand historical data.
2	Medium	Add several different ways to solve the problem - theta method and Holt Winters method.	Add different ways to solve the problem.	Theta method is the least sophisticated method that could work well when data characteristics make it especially difficult to forecast, and Holt Winters could be a more sophisticated alternative for other kinds of data. Users can choose which method to use.
3	Medium	API Backend Encapsulation	Create a system to handle different input formats for the problem data.	API validates all input data and returns clear error messages for invalid data. API provides consistent output format. Documentation includes examples of all API calls and data formats.
4	Medium	Front End Development	Create a Front End.	User Interface that loads the input tables from csv format files and outputs the results in an appropriately

				formatted table, with download buttons.
5	Low	Documentation and Training Materials	Create help guides and training for users.	Has simple instructions for basic use. Has detailed technical documentation. Includes examples of common use cases. Has training materials for new users.

4. DEVELOPMENT AND DESIGN

4.1 User Interface and Workflows

4.1.1. Facility Location Problem: Flow Diagram, Inputs, Outputs

Inputs:

- Optimiser selection (“CBC” or "GA+CBC")
- Demand data for regions and products
- Parameters settings maximum warehouses and maximum investment
- Plant production capacities
- Basic information about plants, products, and regions
- Transportation costs between plants and warehouses
- Transportation costs between warehouses and regions
- Handling costs for products at each warehouse
- Warehouse information including building costs and yearly operation costs

Outputs:

- Best warehouses to be built, and the regions they are supposed to serve.
- Flows from plants to warehouses, and from warehouses to customers.

In Figure 2, a flow diagram of the tool can be found for greater detail.

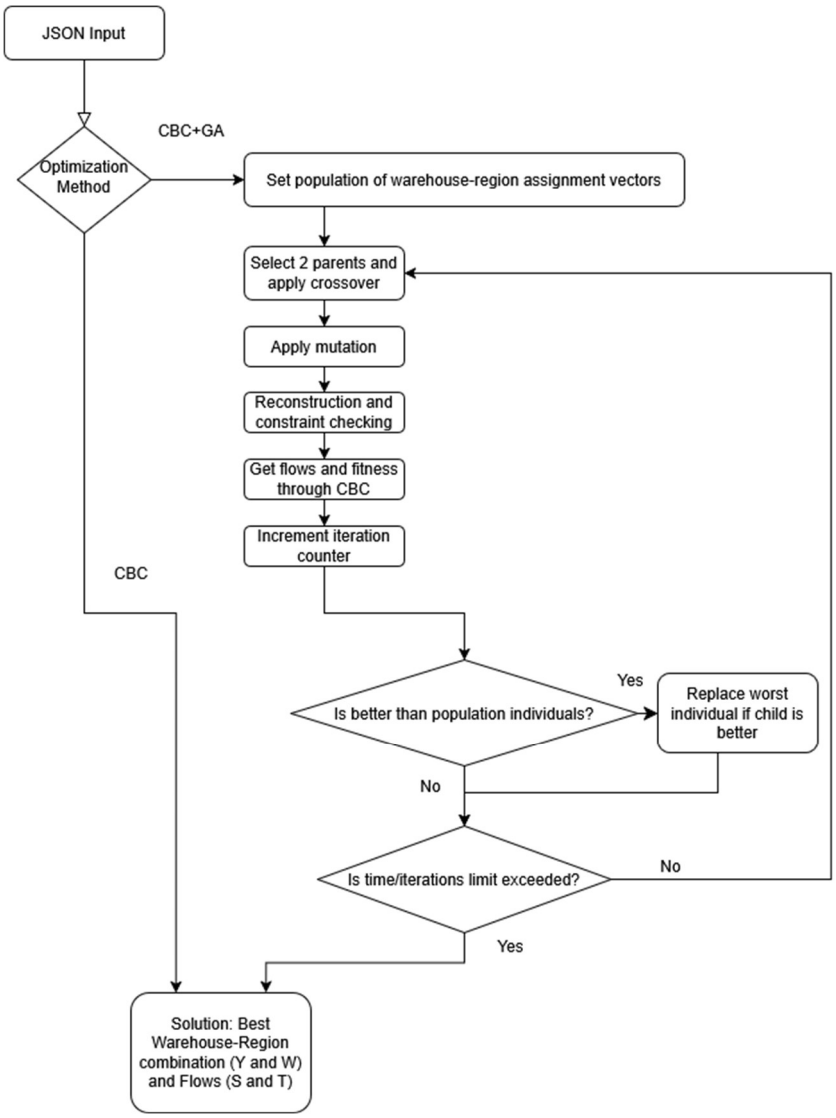

Figure 2 Facility Location Backend Flow Diagram

4.1.2. Inventory Optimisation: Flow Diagram, Inputs, Outputs

Inputs:

- Demand historical data for each warehouse and product.
- Lead time historical data for supply operations.
- Data on the expiration dates of each product.
- Data of each product and warehouse selling price, order cost, unit cost, holding cost, stockout cost and target service level.
- Capacity of warehouses.

Outputs:

- Best paths for the materials to follow, from factories to warehouses and from warehouses to customers, including the quantities of materials and the cost of transportation.
- Fraction of demand met.
- Fraction of link capacity used.

In Figure 3, a flow diagram of the tool can be found for greater detail.

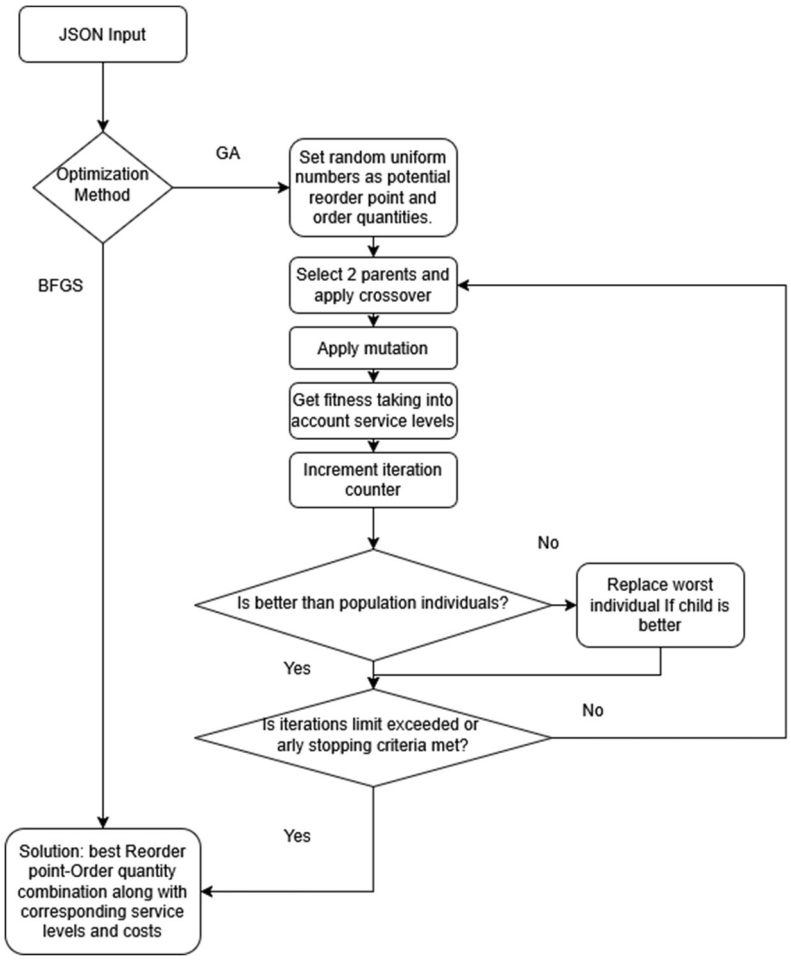

Figure 3 Inventory Optimisation Backend Flow Diagram

4.1.3. Vehicle Routing Problem: Flow Diagram, Inputs, Outputs

Inputs:

- Coordinates, demand, ready and due times and service times for each of the nodes of the network.
- Number of maximum vehicles and their capacity.

Outputs:

- Best routes for the vehicles to follow, distance and demand/fraction of capacity covered.
- Fraction of demand met, and fraction of vehicles used.

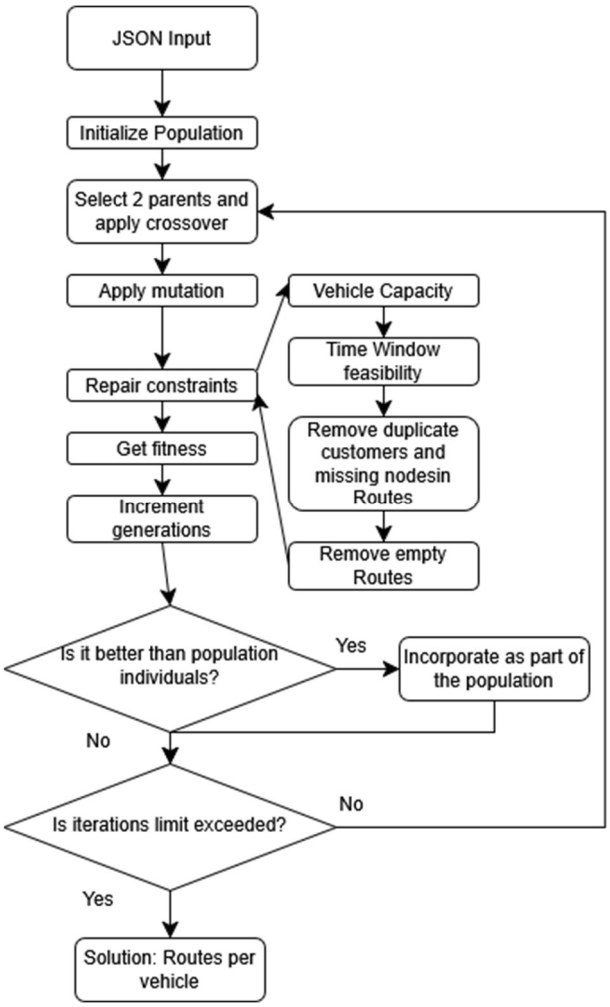

Figure 4 Vehicle Routing Backend Flow Diagram

4.1.4. Network Flow Optimization: Flow Diagram, Inputs, Outputs

Inputs:

- Demand data for regions and products
- Demand coordinates.
- Maximum production and production cost.
- Plant production capacities
- Capacity and cost of transportation.
- Transportation costs between plants and warehouses
- Warehouse capacity and operating cost.

Outputs:

- Best paths for the materials to follow, from factories to warehouses and from warehouses to customers, including the quantities of materials and the cost of transportation.
- Fraction of demand met.
- Fraction of link capacity used.

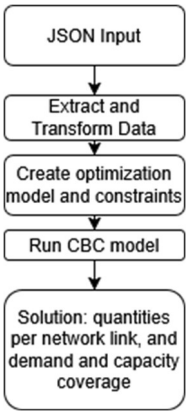

Figure 5 Network Flow Backend Flow Diagram

4.1.5. Supply Demand Matching: Flow Diagram, Inputs, Outputs

Inputs:

- Forecasting method selection (“Theta” or "Holt Winters") and forecasting horizon.
- Historical data to forecast, just before the requested forecasting period.

Outputs:

- Values and dates of the forecast.

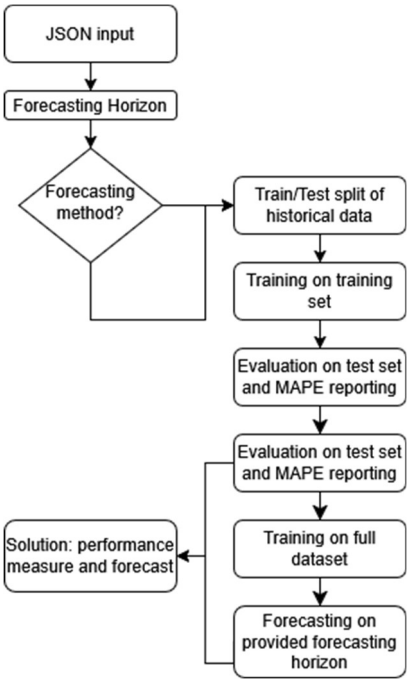

Figure 6 Supply Demand Matching Flow Diagram

5. DATA MANAGEMENT

5.1. User Data Collection

The system works only with data that users provide directly. Users upload their own files or input their data into the tool. No data is collected from other sources or stored on external servers.

5.2. Results

All results are created by running code on the user's data in real-time. The tool processes the information and shows results immediately. These results exist only during the current session and are not saved anywhere.

5.3. Confidentiality

The tool does not store any user data permanently. All processing happens in temporary memory while the code runs. When the session ends, all data is removed from memory. This means the tool's abilities are limited by available memory, but it ensures complete data privacy. Users keep full control of their information at all times.

6. SYSTEM ARCHITECTURE

6.1. Technical Stack

6.1.1. Backend

The backend of this project is built with **Python** and **FastAPI**. Python was chosen because it is easy to use, works well with numbers, and connects with many scientific libraries. FastAPI was picked because it runs fast, has clear documentation through the OpenAPI standard, and makes it simple to build modern **REST APIs**.

During development, the team follows good coding practices. They stick to **PEP 8 standard**, which helps keep code clean, readable, and easy to work with. API routes are set up using the right HTTP methods. GET is used for getting information, and POST is used for creating new resources. The API uses clear version numbers to make sure older versions still work. The system also sends back the right HTTP status codes. It sends 200 when things work correctly and 404 when something cannot be found.

For specific optimisation tasks and numerical calculations, the project uses well-known libraries like **NumPy, SciPy, Pyomo and Pandas**. When speed is very important, the team uses profiling tools to find slow parts of the code and make them faster. The project also uses advanced Python features like FastAPI's async functions to make everything run smoothly.

We take inspiration from trusted documents and standards like:

- **PEP 8** – Python's official style guide
- **FastAPI documentation** – https://fastapi.tiangolo.com
- **OpenAPI standard** – used by FastAPI to create API docs

6.1.2. Frontend

The frontend is developed using modern technologies such as **React** and **Next.js**, along with the **PrimeReact** component library. Next.js is chosen because it can do server-side rendering (SSR), It uses file-based routing, and makes the whole application run better.

For development, the team follows current React best practices. They use functional components with hooks like useState and useEffect. This makes it easier to manage component state and lifecycle in a clear way. The project keeps a clean and clear structure, which makes it easier to maintain and grow over time. Visual elements and interactive parts use PrimeReact, which gives clean, professional and consistent interfaces. The project also uses PrimeFlex for flexible layouts and PrimeIcons for consistent icons. SCSS is used for specific visual changes, following CSS best practices to keep the design clean and modular.

The system handles data well, especially when loading and processing CSV files and talking to backend APIs. It uses standard browser APIs like FileReader and Fetch for this work. The interface is also set up to avoid unnecessary re-rendering by using React hooks to manage component updates efficiently. Finally, the web accessibility and user experience (UX) follow WCAG (Web Content Accessibility Guidelines) standards. This makes sure the interface works well for all users and is easy to use.

We take inspiration from trusted documents and standards like:

- **React documentation** – https://react.dev
- **Next.js documentation** – https://nextjs.org/docs
- **PrimeReact documentation** – https://primereact.org
- **Web Accessibility Guidelines (WCAG)** – for making the app usable by everyone

6.2. Main Components

This optimization platform has a modern web frontend built with **Next.js and React** which is linked to five backend microservices. The frontend application uses **PrimeReact** components for the user interface, allowing users to upload CSV files, set optimization parameters, and see results through interactive tables and charts. The styling uses **SCSS** files, while the services layer handles all API communications with the backend systems.

The backend architecture follows a microservices pattern. Each optimization problem domain has its own **FastAPI** service:

- **Facility Location Problem** service performs warehouse placement optimization.
- The **Inventory Optimization** manages stock level calculations.
- The **Network Flow Optimization** solves supply chain flow problems.
- The **Supply Demand Management** provides forecasting capabilities.
- The **Vehicle Routing Problem** optimizes delivery routes.

Each service contains its own algorithm modules, sample data for testing, and runs independently in **Docker** containers.

The entire system is organized using **Docker Compose** for local development and **Helm charts** for Kubernetes deployment in production environments. Authentication is managed by a separate **Keycloak** service, which makes sure access is secure across all components. The communication flow starts when users upload data through the React frontend. The frontend then sends requests to the right backend service, where specialized algorithms process the optimization problems and return structured results. These results are displayed back to the user through formatted tables and downloadable reports.

7. DEPLOYMENT AND ACCESS

7.1. HOSTING ENVIRONMENT

The platform is deployed on a server running Ubuntu Server 24.04, with 8 GB of RAM and approximately 25 GB of disk space. Docker (version 27.5.1) and Docker Compose are used to manage microservices. All of this makes sure the resulting application is isolated, scalable and stable.

Each system algorithm is packaged in a separate Docker image that exposes its own API. This allows for independent updates and horizontal scaling when required. The frontend is also deployed in its own Docker container. This provides a robust and user-friendly interface for the end user.

7.2. ACCESS INSTRUCTIONS

Keycloak provides secure authentication in the solution. The steps to access the platform are the following:

1. Open a web browser (Chrome, Firefox, Edge).

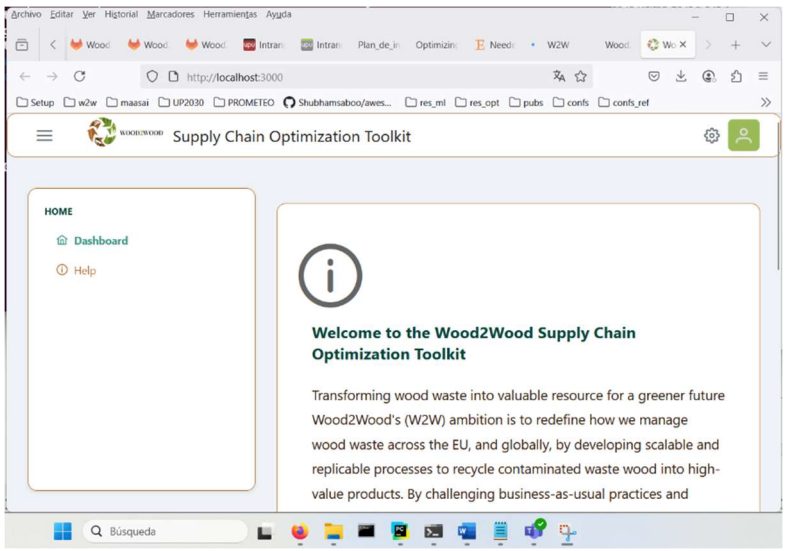

Figure 7 Use a modern web browser (Chrome, Firefox, Edge)

2. Enter the URL.

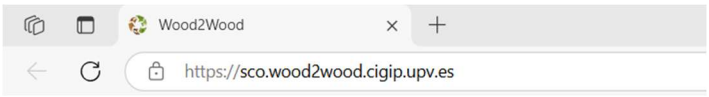

Figure 8 Enter the URL provided to access the system frontend

3. Enter valid credentials (username and password) in the Keycloak authentication interface.

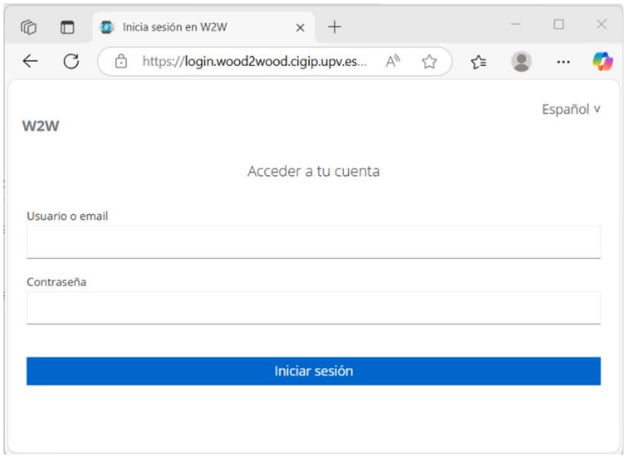

Figure 9 Enter valid credentials

4. Once authenticated, users will have full access according to their permissions, allowing them to upload data, configure optimisations, and download results.

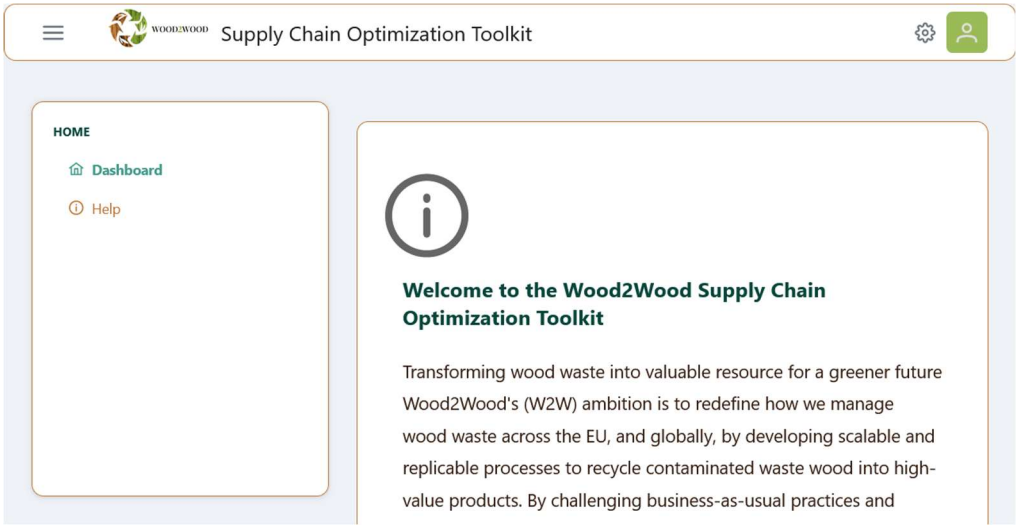

Figure 10 Once authenticated, users have full access

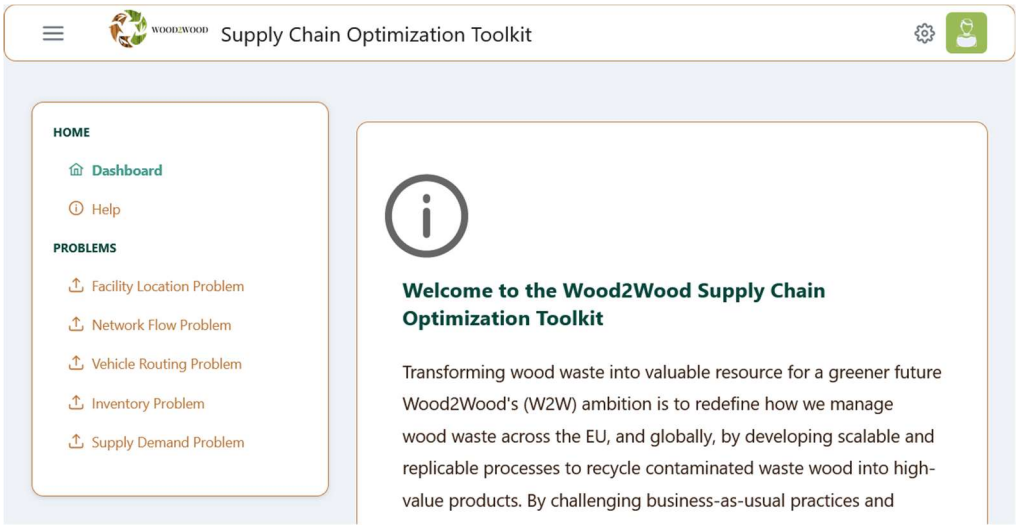

Figure 11 Once authenticated, users have full access

8. TESTING

Tests were performed at different levels of the development to check the proper performance of the different parts. Two levels of tests have been performed, the backend tests regarding how the algorithms worked, and the tests on the full tools, including the front-end.

8.1. ALGORITHM TESTING

Synthetic data was generated randomly for each of the algorithms, considering the information that should be available to solve each of the problems. The algorithms were developed in a first contextualisation phase and the outputs were visually processed to see If they were of any sense. Here, a sample of those visualisations is shown along with a brief explanation of the results.

8.1.1. Facility Location Problem

Gurobi works the best and fastest of all the options which were tested. It gives the best results for all problem sizes. But our new method that mixes a genetic algorithm with CBC is also good, especially for big problems with many warehouses. CBC alone is still better than this combination for small instances. The best part is that our method is free and anyone can use it. When problems get bigger and harder, the difference between Gurobi and our method gets smaller. This makes our approach useful for solving complex warehouse problems without paying for expensive licenses.

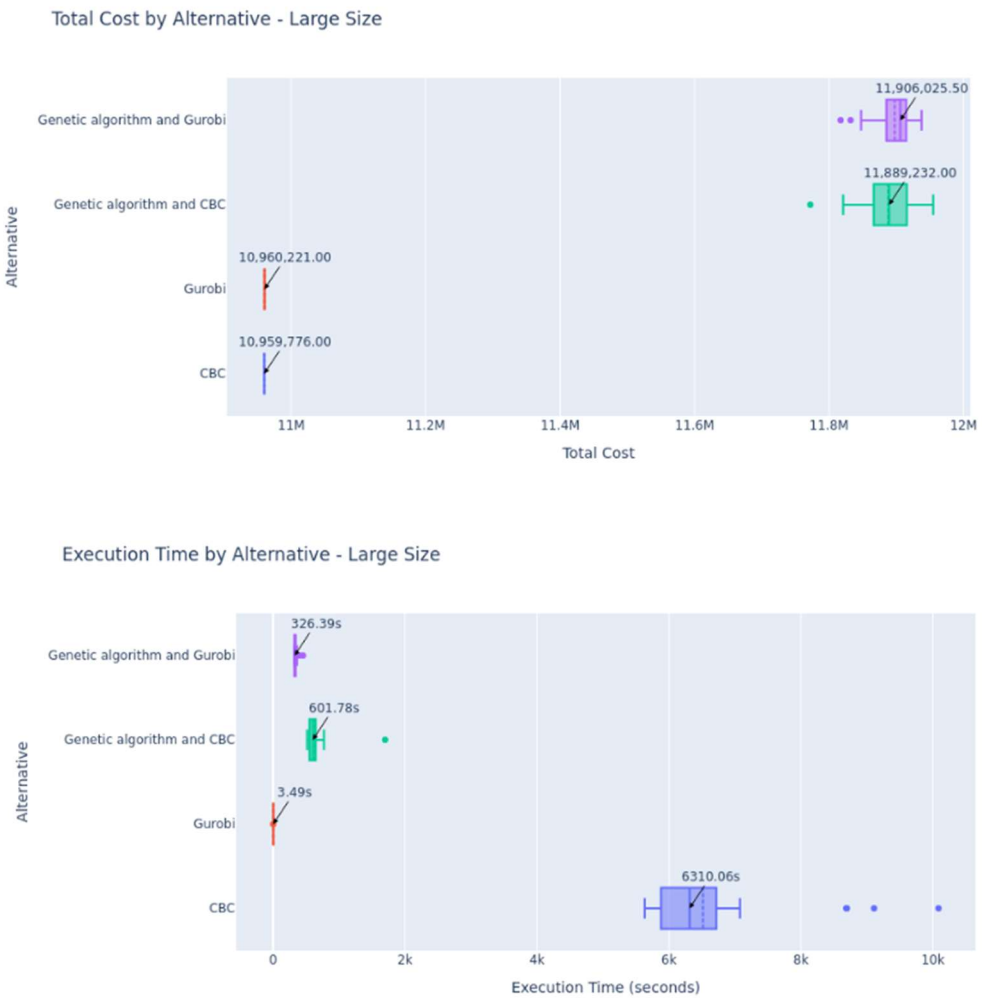

Figure 12 Execution time and optimisation function value for various algorithms tested regarding facility location problem instances

8.1.2. Inventory Optimisation

For inventory optimisation, there are two choices, so users can try different methods depending on what problem they have. The first choice is a genetic algorithm and the second is BFGS, which is a different type of heuristic method. Proper data was generated so that clear results for Figure 8 are shown. Figure 8 shows the different costs that come from the order quantity and reorder point. These numbers come from the process of finding the best answer for one example of this problem. The figure helps us see how different choices affect the total costs in the system.

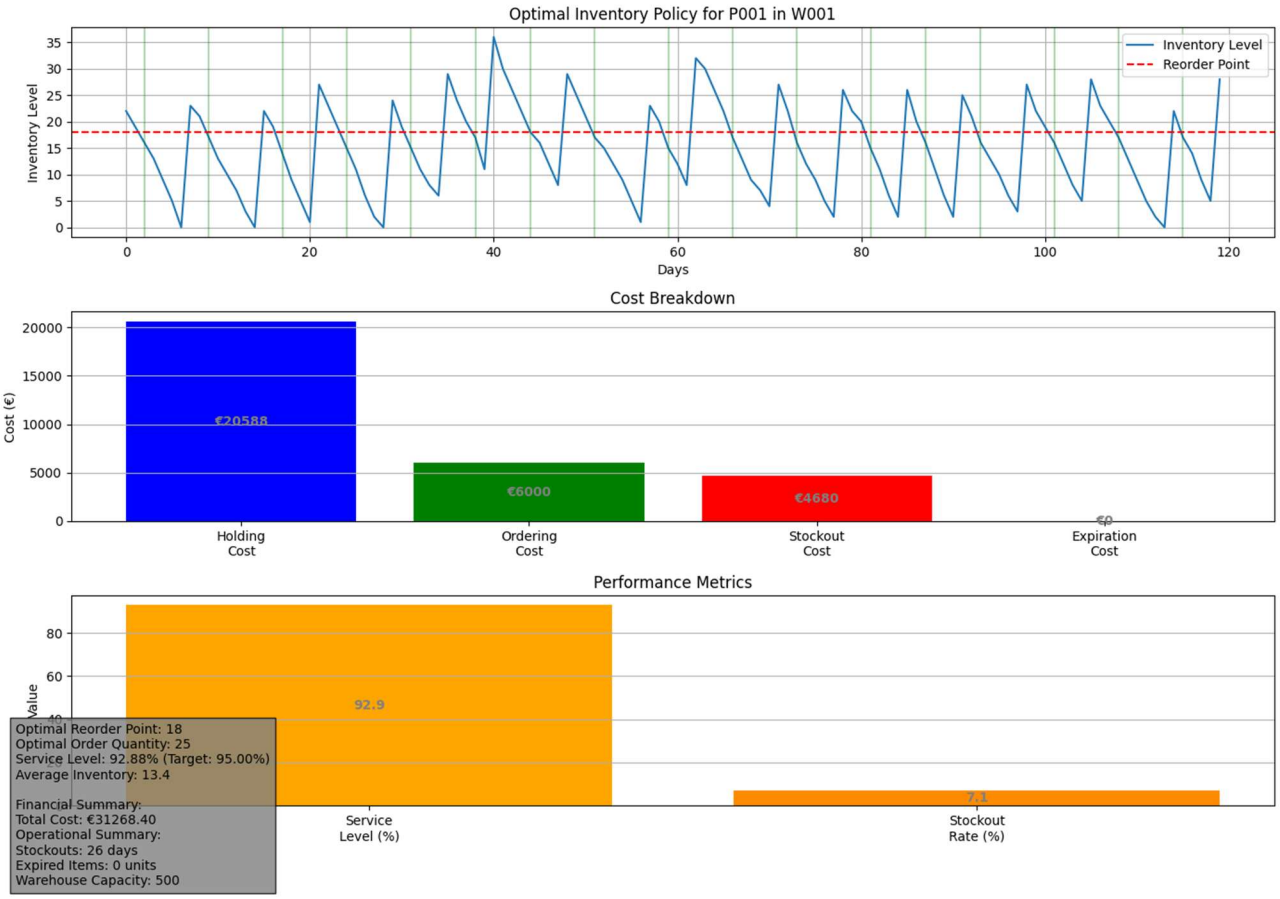

Figure 13 Visual output of the conceptualisation phase of the inventory optimisation problem trials

8.1.3. Vehicle Routing Problem

The vehicle routing problem was built using a standard test case called the Solomon random instance. This test case has 50 vehicles in it. Only one algorithm was tried, a genetic algorithm.

The next phase of the project focuses on two main areas of development. The first area involves improving the quality of results that the current solutions produce. The second area requires testing different types of algorithms beyond the genetic algorithm approach currently in use. Testing various algorithm types may lead to better solutions and more accurate results. Figure 9 displays the current appearance of one of these solutions in operation.

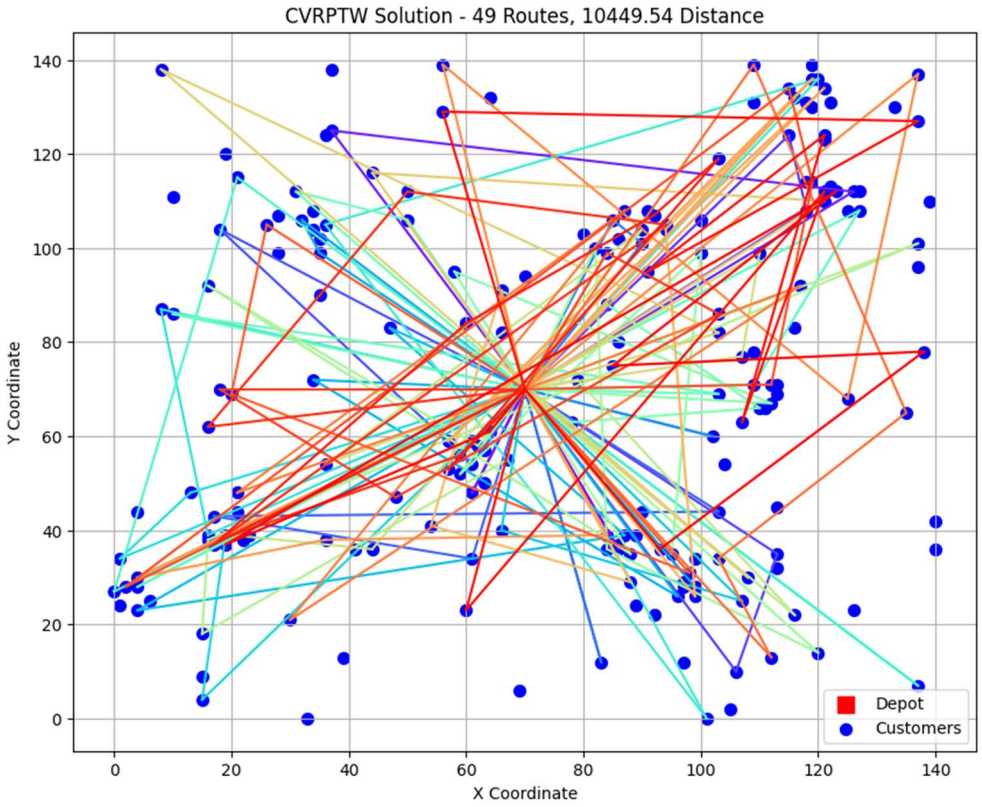

Figure 14 Map of solutions for the vehicle routing problem

8.1.4. Network Flow Optimisation

Exact methods remain the most effective approach for solving network flow problems. These methods produce reliable results while maintaining reasonable computation times. This method was selected for the project because it provides correct solutions within acceptable time limits.

The figure displays the movement of materials through the system. Materials originate at factories, are transported to warehouses, and then distributed to customers, as illustrated in Figure 10. This flow pattern reveals how the network operates and identifies areas where improvements are needed. Exact methods help determine the most efficient way to move these materials, which reduces both time and costs.

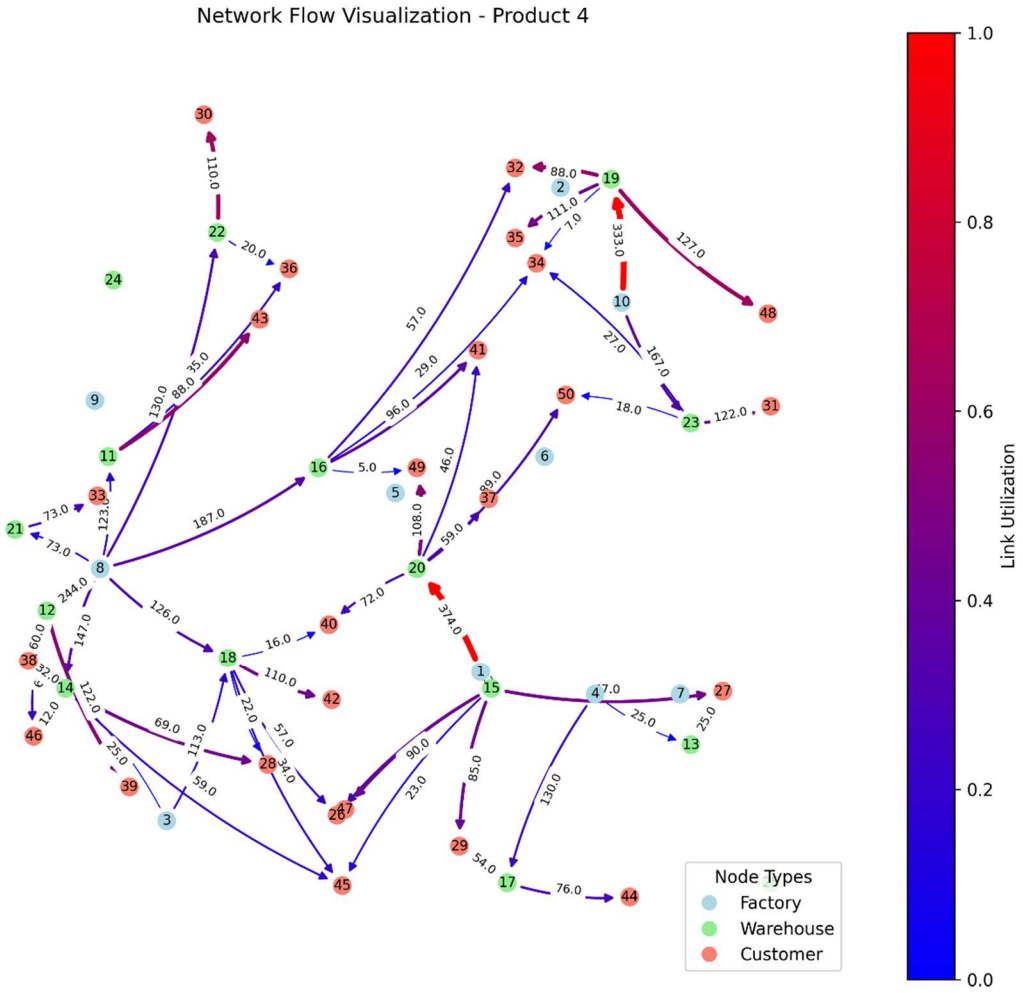

Figure 15 Visual map of solutions of network flow optimisation process for one of the products in a problem instance

8.1.5. Supply Demand Matching

Two algorithms were selected to match supply and demand. These algorithms work by taking historical data and dividing it into two parts. 80% of the data trains the algorithm, while the remaining 20% tests how well it performs. The algorithm's accuracy is measured by comparing its predictions to the actual results in the test set using a metric called MAPE (Mean Absolute Percentage Error).

Figure 11 presents the results from this work. It shows what the algorithm predicted for the entire period of available data. It also displays the actual numbers from that same period, which allows for comparison between predictions and real outcomes. The figure includes the algorithm's predictions for future periods based on patterns learned from the historical data.

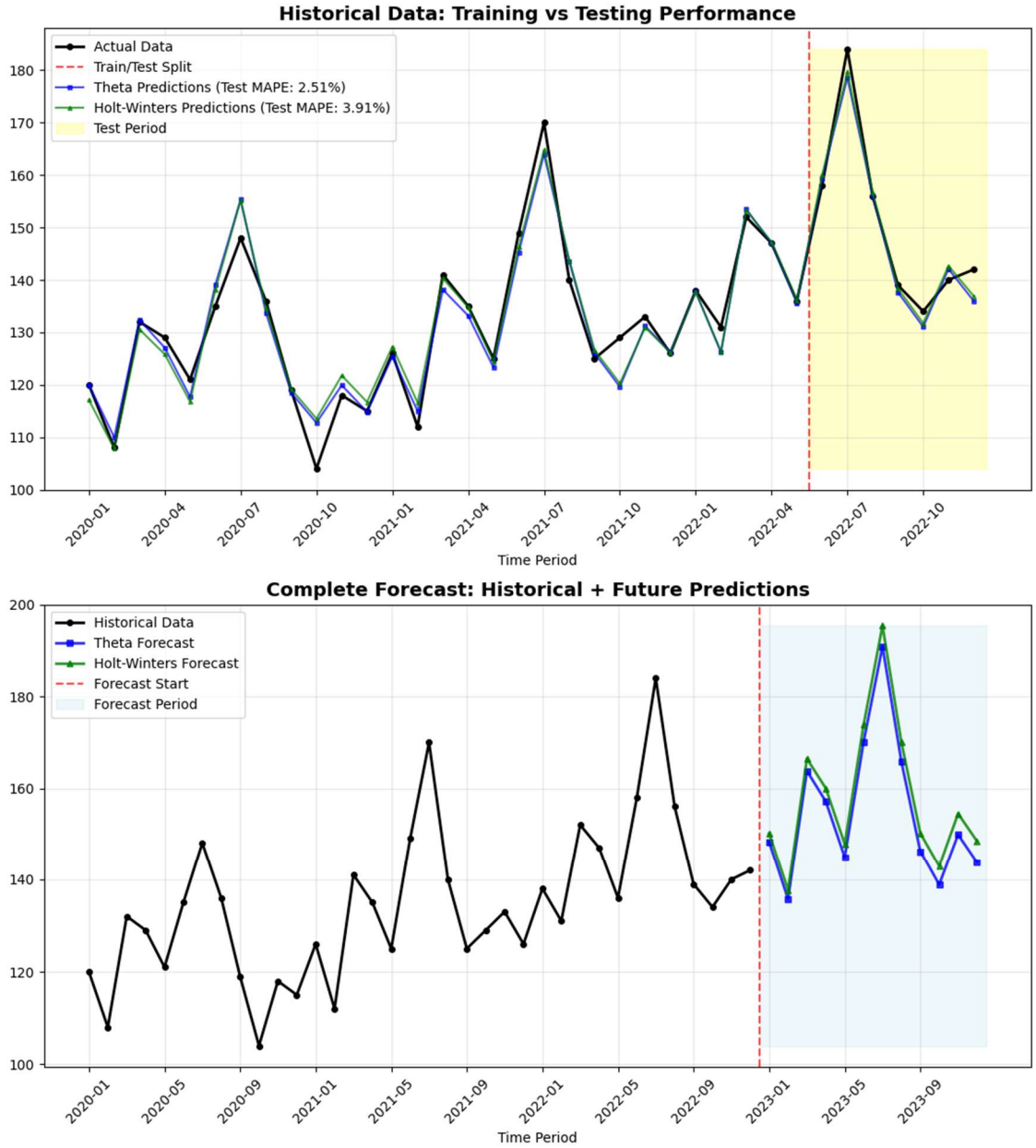

Figure 16 Representation of the historical data time series and the results of 2 forecasting models for the same period with a specific train test split (above) and the same historical data time series along with a future forecast (below)

8.2. TOOLS TESTING

In this section, the user interface for each of the problems is shown, the kinds of inputs it receives and the outputs it produces. The workflow is the same for all cases as it can be seen from captures, it starts with the selection of the algorithm in case it applies, then it follows with the csv input upload and finally the output is returned and it is possible to download that output in csv or JSON format. All cases work properly, so the integration between the front end and the backend is working smoothly and correctly. Some downloadable examples from buttons in different pages are considered as a valuable improvement for future deliverables

8.2.1. Facility Location Problem

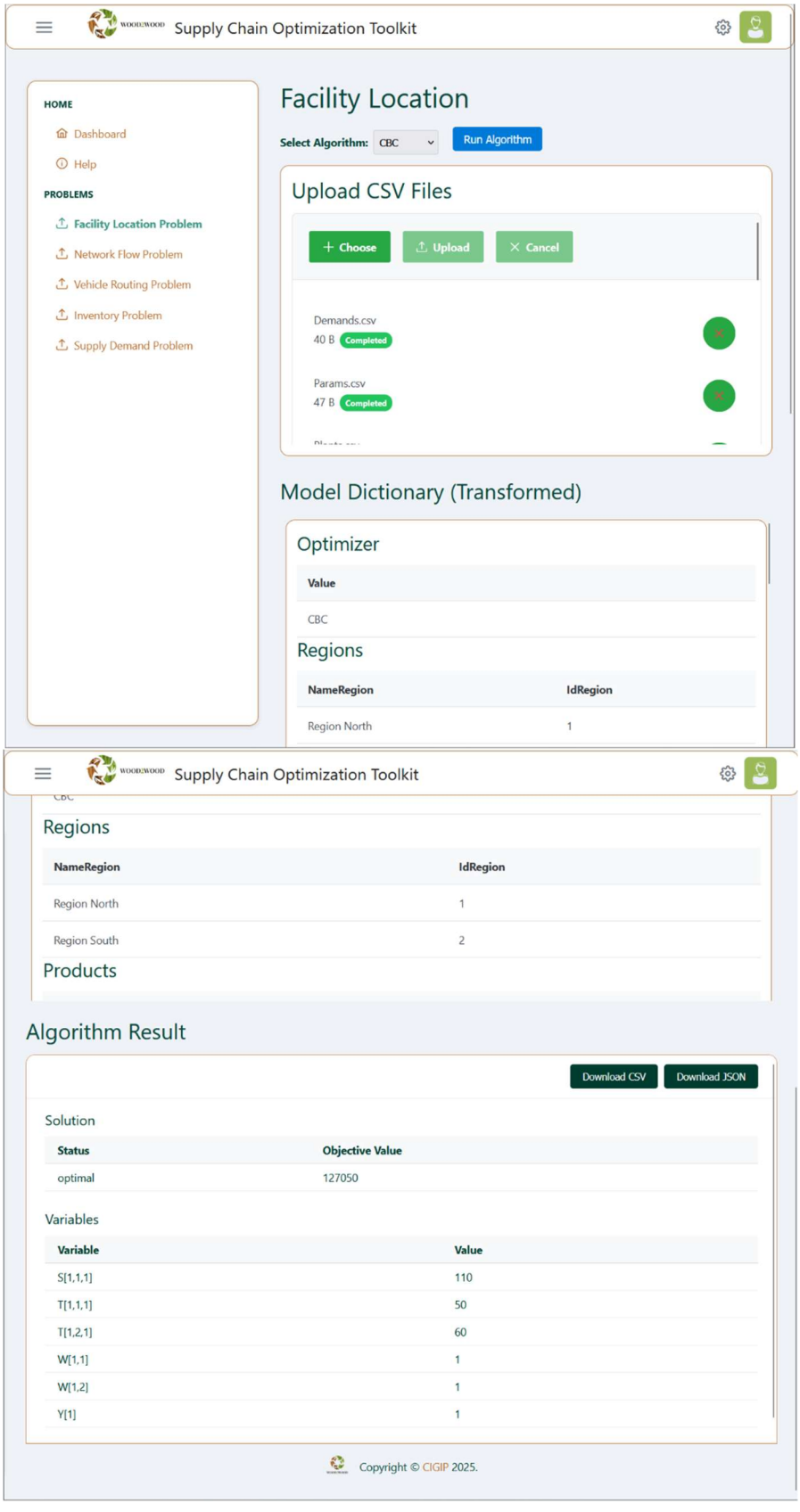

Figure 17 Testing process picture for one of the algorithms used for the Facility Location Problem, from the front end

8.2.2. Inventory Optimisation

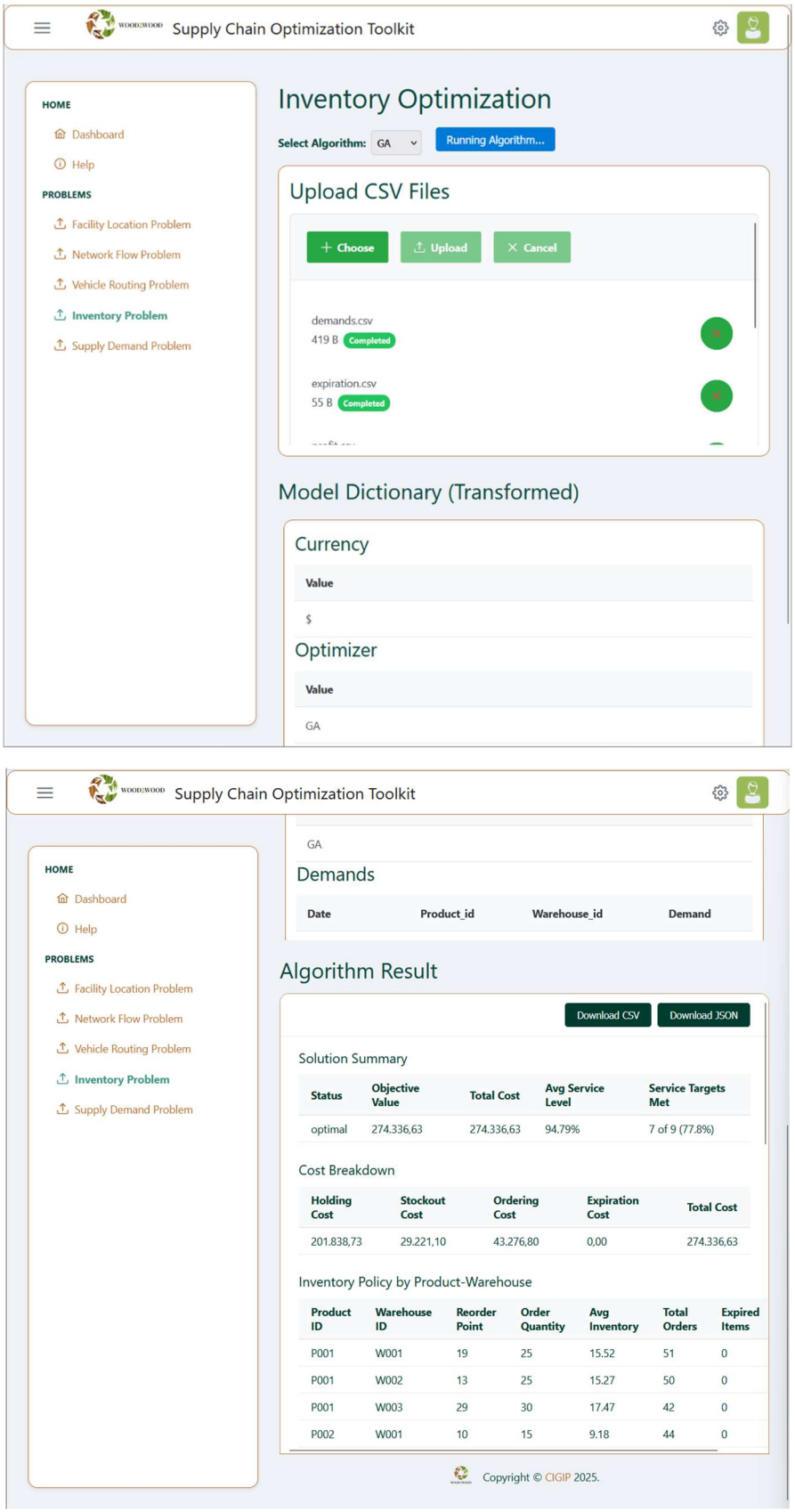

Figure 18 Testing process picture for one of the algorithms used for the Inventory Optimisation Problem, from the front end

8.2.3. Vehicle Routing Problem

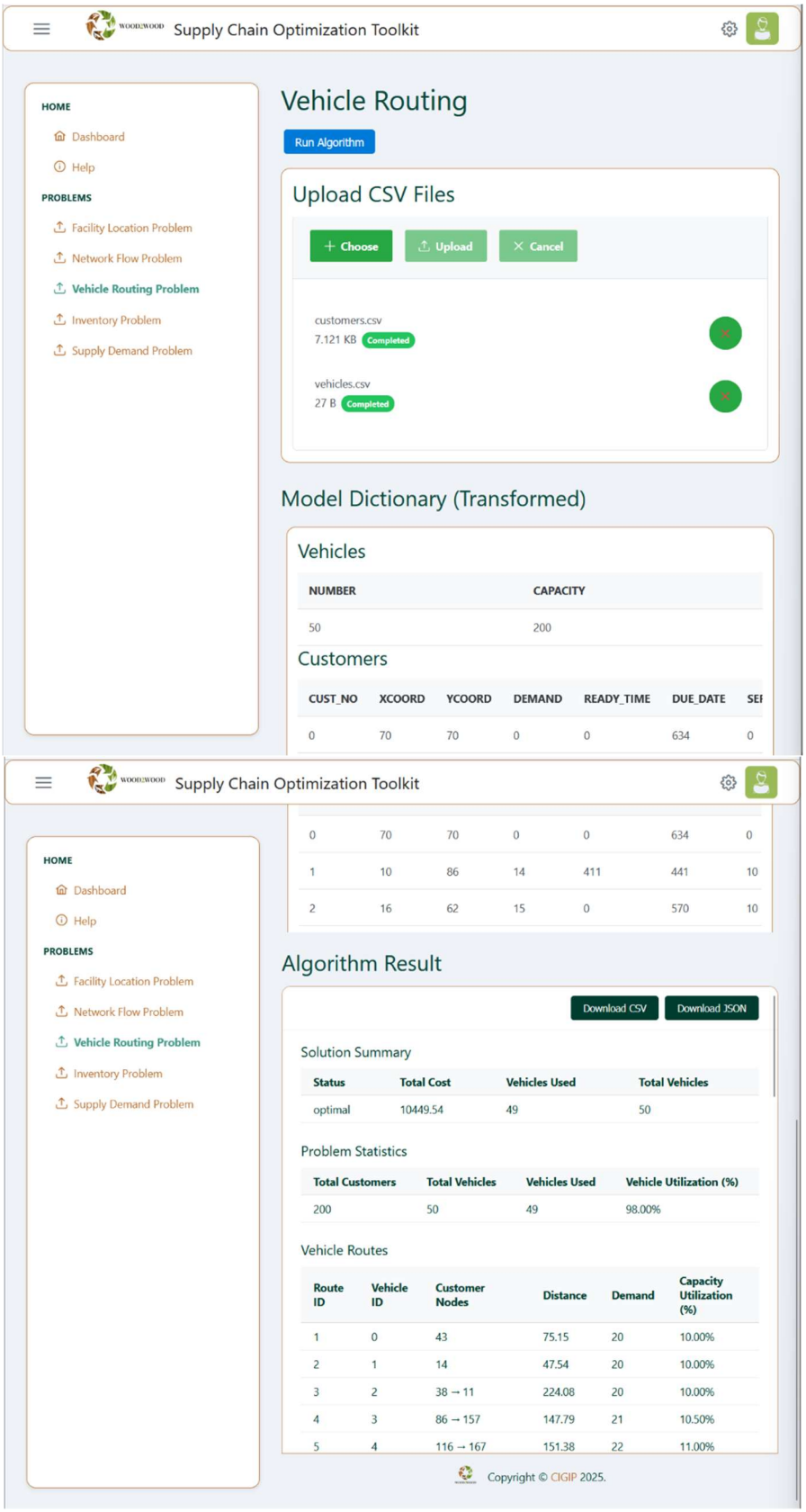

Figure 19 Testing process picture for the algorithm used for the Vehicle Routing Problem, from the front end

8.2.4. Network Flow Optimisation

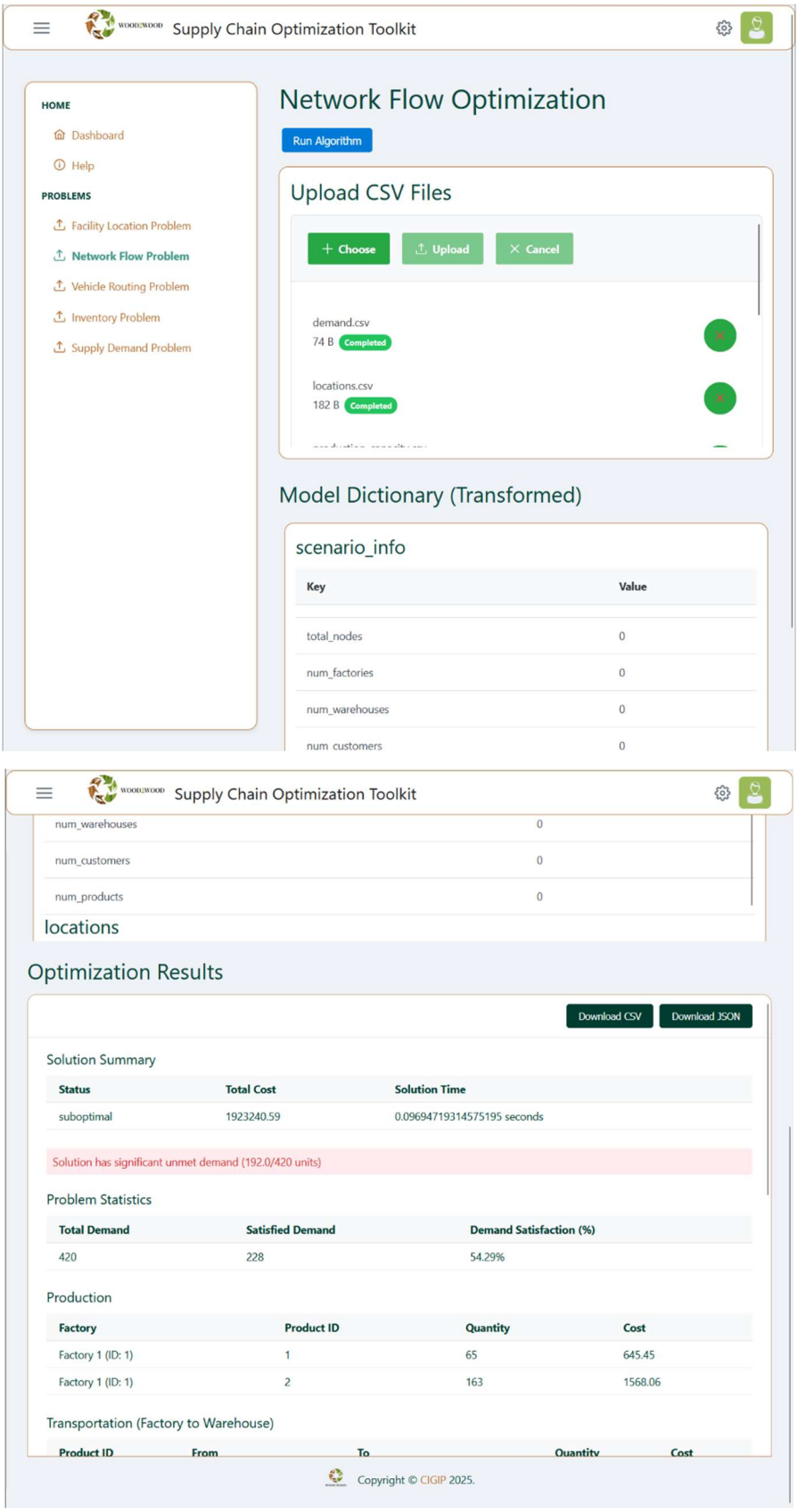

Figure 20 Testing process capture of the algorithm used for the Network Flow Optimisation Problem, from the front end

8.2.5. Supply Demand Matching

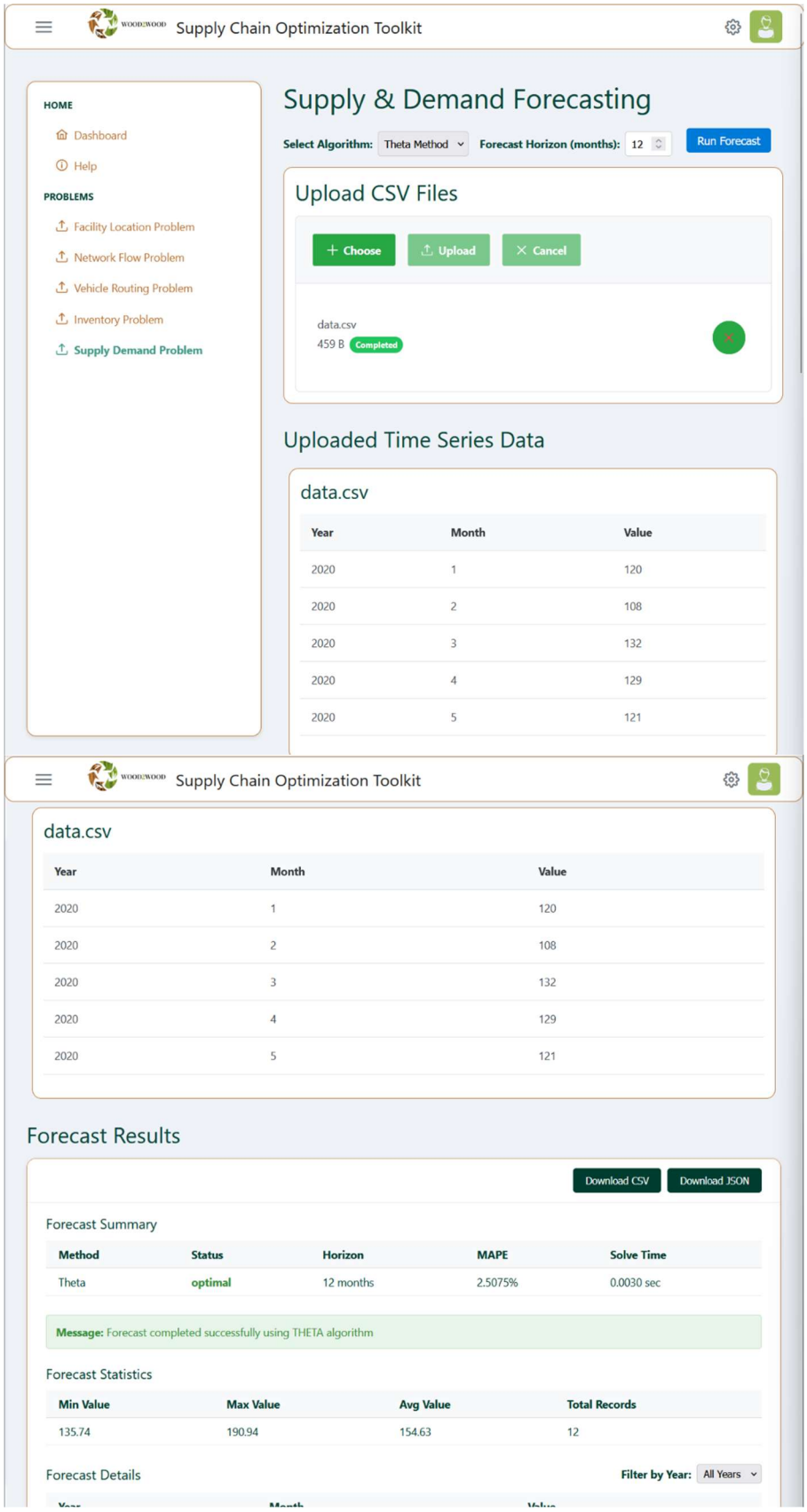

Figure 21 Testing process capture of one of the algorithms used for the Supply Demand Matching Problem, from the front end

9. NEXT STEPS

9.1. PLANNED FEATURES FOR V2

There are some basic features which have been planned for the second version of the toolkit, and which are somehow transversal to any of the specific problems:

- Data validator for inputs.
- Button to download templates for each of the problems for the user to check the proper input format, after the algorithm is selected.
- More algorithms or approaches to solve some of the problems, considering different possible requirements when solving a specific problem.
- Problems and interface adjustments according to the feedback and new requirements collected from the Use Case agents.

9.2. FEEDBACK COLLECTION STRATEGY

The project will gather requirements again through repeated cycles when necessary. This process will follow the same approach used during the initial requirement collection. The focus will be on specific use cases that have the most importance.

Meetings will be scheduled to present the tool to the use case teams. Following the presentation, teams will receive time to test the tool independently. Teams will then provide their thoughts and suggestions about the tool. This feedback will help improve the tool's interface and make it easier to use. The feedback will also help determine which algorithms deserve additional development time.

This cycle of presenting, testing, and collecting feedback will repeat until the correct solution is found. Each round will make the tool more useful for the end user's needs.

9.3. TIMELINE

The development team will continue improving the algorithms already present in the tool while collecting new requirements. This process will continue throughout the entire project duration. Interface changes are expected to require significant time, and work will continue on these changes until end users are satisfied with the tool's operation. This process will begin when the requirement collection starts. After interface changes are completed, the development team will shift focus toward building new algorithms. The selection of which algorithms to develop will be based on specific needs identified through use case analysis. The final phase will address solution details, including the creation of template download buttons, improvements to error handling, and resolution of any integration issues that come up.

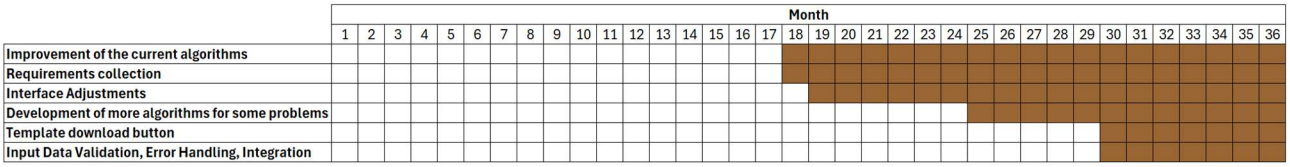

Figure 22 Roadmap for the next development phase

10. CONCLUSION

The Wood2Wood project has successfully delivered the first version of a Supply Chain Optimization Toolkit designed to help companies in the wood recycling industry operations. This web-based platform addresses five main supply chain problems: warehouse locations, inventory levels, delivery routes, material flows, and demand forecasting.

The toolkit is an important step toward circular economy practices in the wood industry. The platform contributes to reducing environmental impact and waste sent to landfills through better logistics and materials management. The system allows to test different scenarios virtually before making costly changes to their real operations.

Testing results show that the algorithms work correctly and produce reasonable solutions. Regarding the facility location tool, the genetic algorithm combined with CBC solver provides a free alternative to expensive licensed software and it provides good performance. The inventory optimization tool offers clear cost analysis through two different solving methods. The network flow optimization uses exact methods to get reliable results within acceptable time limits. The forecasting tool achieved good accuracy when tested against historical data using both Theta and Holt Winters methods.

However, Version 1 has significant limitations. The platform lacks data validation capabilities; it can't verify if uploaded files contain correct information. There are no template files available to guide users in formatting their data properly. The system needs better error handling to provide helpful messages. Some optimization tools offer only one solving method instead of multiple options. Most importantly, the toolkit has not been tested with the real use cases, so its practical effectiveness remains unknown.

The development team has identified clear priorities for Version 2. The next phase will focus on collecting feedback from actual industry users, adding data validation features, providing downloadable template files, implementing additional algorithms, and improving user interface. The project will continue through repeated cycles of testing and improvement based on input from use case partners.

The toolkit provides a solid foundation for future development. The microservices architecture using Docker containers makes the system scalable and maintainable. The separation of frontend and backend services allows for independent updates and improvements. The choice of modern web technologies like React, Next.js, and FastAPI makes it reliable in the long term.

The success of this toolkit will depend on how well it meets the real requirements of companies. The planned feedback collection strategy and iterative development approach is a step forward towards this goal. The project shows digital tools can play an important role in improving circular economy practices.

WOOD2WOOD