ReMuNet

Pioneering resilient and adaptive multimodal transport networks

D4.3

# Optimization algorithm tested and parameterized

Fraunhofer Austria Research GmbH

19/12/2025

Funded by the European Union

## PROJECT INFORMATION

| | |
|---|---|
| **PROGRAMME** | Horizon Europe |
| **TOPIC** | HORIZON-CL5-2022-D6-02-07 |
| **TYPE OF ACTION** | HORIZON Research and Innovation Actions |
| **PROJECT NUMBER** | 101104072 |
| **START DAY** | 1 July 2023 |
| **DURATION** | 36 months |

## DOCUMENT INFORMATION

| | |
|---|---|
| **TITLE** | Optimization algorithm tested and parametrized |
| **WORK PACKAGE** | 4 |
| **TASK** | T 4.3 Develop agent-based reinforcement learning optimization |
| **AUTHORS (Organisation)** | Catherine Laflamme (FhA), Thomas Kroiss (FhA), Sandra Stein (FhA) |
| **REVIEWERS** | All Partners |
| **DATE** | 1.12.2025 |

## DISSEMINATION LEVEL

| | | |
|---|---|---|
| **PU** | Public, fully open | x |
| **SEN** | Sensitive, limited under the conditions of the Grant Agreement | |
| **Classified R-UE/EU-R** | EU RESTRICTED under the Commission Decision No2015/444 | |
| **Classified C-UE/EU-C** | EU CONFIDENTIAL under the Commission Decision No2015/444 | |
| **Classified S-UE/EU-S** | EU SECRET under the Commission Decision No2015/444 | |

## DOCUMENT HISTORY

| VERSION | DATE | CHANGES | RESPONSIBLE PARTNER |
|---|---|---|---|
| 1.0 | 01/11/2025 | Preliminary version for review by partners | Catherine Laflamme (FhA), Thomas Kroiss (FhA), Sandra Stein (FhA) |
| 1.1 | 04/12/2025 | Additions to chapter 2-3 | Thomas Kroiss (FhA) |
| 1.2 | 12/12/2025 | Review of all chapters | Thomas Kroiss (FhA), Felix Kamhuber (FhA) |
| 1.3 | 17/12/2025 | Additions to chapter 3, implemented scenarios | Leonhard Czarnetzki (FhA), Thomas Kroiss (FhA) |
| 1.4 | 19/12/2025 | Final version and external submission | Sandra Stein (FhA) |

Pioneering resilient and adaptive multimodal transport networks

# TABLE OF CONTENTS

# LIST OF FIGURES



# ABBREVIATIONS

| | |
|---|---|
| **RL** | Reinforcement Learning |
| **PPO** | Proximal Policy Optimization |

# Executive Summary

# 1. Introduction

Synchromodal transport is emerging as a cornerstone of next-generation freight logistics, aiming to boost efficiency and cut emissions by shifting loads from road to more sustainable modes such as rail and inland waterways, while enabling real-time re-planning and routing across a connected network. Within Europe, this vision aligns with the long-term drive toward an integrated, multimodal, and sustainable logistics ecosystem, where dynamic coordination across actors and modes allows for both environmental and economical gains.

However, translating this potential into widespread practice remains challenging. Operational plans are often rigid, collaboration among stakeholders is limited, and the underlying systems are complex and highly dynamic, making real-time coordination challenging. Sustainable modes like rail and waterways are also more susceptible to disruptions, which can render plans infeasible and trigger cascading effects across the network - raising the bar for resilience, responsiveness, and robustness in day-to-day operations.

These realities motivate a planning paradigm that both anticipates and adapts. Effective synchromodal deployment requires decision support that can: (i) react swiftly to disturbances in real time, and (ii) design robust plans that hedge against uncertainty before it materializes. Reinforcement Learning (RL), a subfield of AI, is well-suited to this dual mandate. By learning policies through interaction in a state–action–reward loop, RL agents can navigate high-dimensional, stochastic decision spaces and optimize long-horizon outcomes, capturing not only immediate operational effects but also delayed consequences of today's choices. However, practical RL adoption poses its own challenges, including solution quality and stability, training and deployment complexity, transferability across contexts, and transparency for stakeholders. In our previous report (D4.2) we discussed the different possibilities of how RL can be used to improve the re-planning problem of synchromodal transport systems. Instead of generating routes from scratch, our agent acts as a high-level decision maker that selects from a robust set of path-finding heuristics - specifically cheapest, earliest arrival, lowest CO2, and minimal road distance - depending on the current network state. This design ensures valid solutions while engaging the RL policies to balance conflicting objectives dynamically. In this report (D4.3) we will go into detail about how our chosen method was technically implemented, including the input data required and the definition of scenarios.

# 2. Description of implemented algorithm

## 2.1 Overview Hybrid-RL Method

Our chosen method for implementation is hybrid-RL, where RL is combined with standard heuristics to optimize dynamic rescheduling in the presence of disruptions. The reasoning behind this decision is given in more detail in D4.2. Here we will concentrate on the technical implementation of this method.

We focus on a hybrid RL approach, where the main advantage is the idea of integrating prior knowledge about the problem into the RL agent. This can be done by, e.g., pre-training the policy networks using expert behavior, by using explicit forecasting methods to add insights to the agent's observation space, or by reducing the problem's complexity by using known methods for subproblems/tasks. This restricts the scope of the RL Agent to the aspects of the problem where other methods perform poorly, such as dealing with disruptions.

Our method is based on [1], where an RL agent operates on a discrete action space of heuristics and aims at selecting the right heuristic to solve a given subproblem of an optimization problem. Here, the subproblem is re-planning the transport of a single request under disruption, and we assign the RL Agent the task of deciding if re-planning is necessary and, if so, to choose the appropriate heuristic to do so.

Every time an order reaches a terminal the agent receives information about the current network state, including all disruptions, future vehicles and their capacities. After deciding to perform re-planning the agent chooses from the defined set of different heuristics (see D4.2 for a more in-depth discussion on what set of heuristics was used) and aims to select the one that contributes to maximizing the overall reward. Once a heuristic is selected, it is applied to the underlying network and re-planning is performed accordingly. This is shown schematically in Fig. 1.

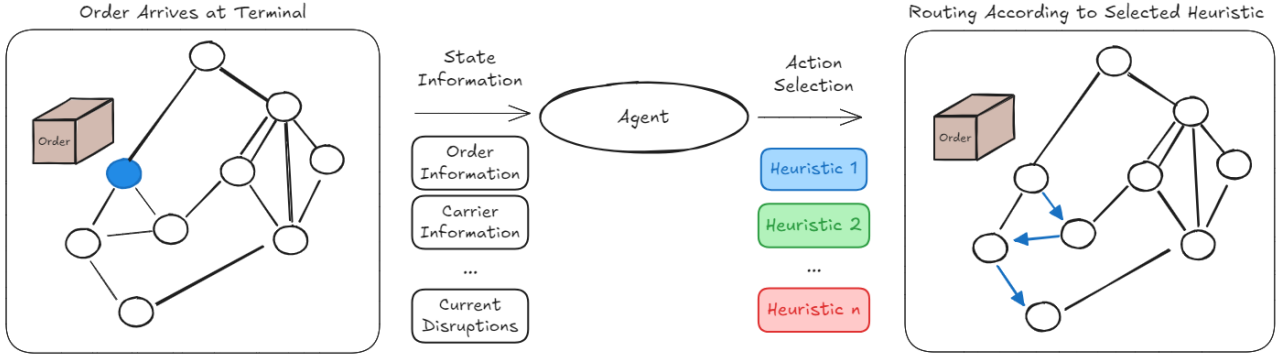



**Figure 1: The chosen implementation of hybrid RL for dynamic re-planning of synchromodal transport syste**



[1] - Kai Hu et al., A review of research on reinforcement learning algorithms for multi-agents, Neurocomputing 599 (2024): 128068, doi:10.1016/j.neucom.2024.128068.

The agent is trained to make the optimal decision about which heuristic to take via RL, based on the final reward (calculated as a combination of costs and CO2 impact). Once the agent is trained, it can be run online in real time. The technical implementation is described in the following sections.

## 2.2 Technical Implementation

The technical implementation of our approach includes two distinct phases:

1. **Training Phase:** In this phase, the Reinforcement Learning (RL) agent is trained to learn an optimal policy for decision-making under uncertainty. The implementation utilizes the Stable Baselines 3 library, specifically the Proximal Policy Optimization (PPO) algorithm. The first phase consists of:
   - **Environment**: A custom OpenAI Gym environment (*RemunetGymFactory*) wraps the core simulation. It exposes the state of the transport network (carrier locations, delays, order status) as observations to the agent.
   - **Loop:** The training process follows a standard RL loop: the agent observes the state, selects an action (choosing an implemented routing heuristic, see Del 4.2), and receives a reward based on the outcome (e.g., on-time delivery, CO2 emissions).
   - **Optimization:** Over many episodes (simulation runs), the PPO algorithm updates the agent's neural network weights to maximize the cumulative reward, effectively "learning" which heuristics work best in different disruption scenarios. The entry point for this phase is *remunet_model_training*.

2. **Online Phase:** Once trained, the agent (or the trained model) is deployed to make real-time decisions during a simulation run or potentially in a live operation.
   - **Execution:** The simulation is executed using the Simulation class (in *remunet_sim/sim/sim*). It runs in a "step-by-step" mode using the *run_until_decision_point* method.
   - **Interaction:** When a carrier arrives at a hub (a "decision point"), the simulation pauses and yields a *DecisionData* object containing the current context (affected orders, available paths).
   - **Decision:** The trained model receives this data, computes the best action (heuristic), and returns a new set of order assignments.

- **Continuation:** These assignments are fed back into the simulation via *sim.set_order_assignments*, and the simulation resumes (*DecisionPointResume*) until the next decision point is reached. This allows for dynamic re-planning in response to unfolding events like delays.

The backbone of the implementation is the simulation environment which is built on the following technical aspects:

**Input Data**

1. Define the transportation network: Allow for predetermined or stochastic disruptive events
2. Define the orders: Define the orders that will be transported through the network including deadlines and priorities.

**Reset/Initialise the Simulation**

3. Build an initial solution: Build an initial solution for the orders to be transported through the network to the goal hub.

**Run the Simulation**

4. Simulate the transportation network: Simulate the transportation network and the orders being transported through it, subject to stochastic disruptive events.
5. Stop at each decision point and re-plan: Every time a carrier enters a hub there is the possibility to re-plan the route. In order to restrict the optimisation space, a fixed set of heuristics are used as the possibilities for re-planning. These heuristics are newly calculated based on the current status of the network. This re-planning can either be done by a benchmark policy or by a trained RL agent.

**Evaluate the Results**

6. Evaluate the results: Evaluate the results of the simulation and compare the results of the benchmark policy with the results of the RL agent.

**Visualise the Results**

7. Visualise the flow through the network and the decisions made along the way

An overview of this approach is given in Fig. 2.

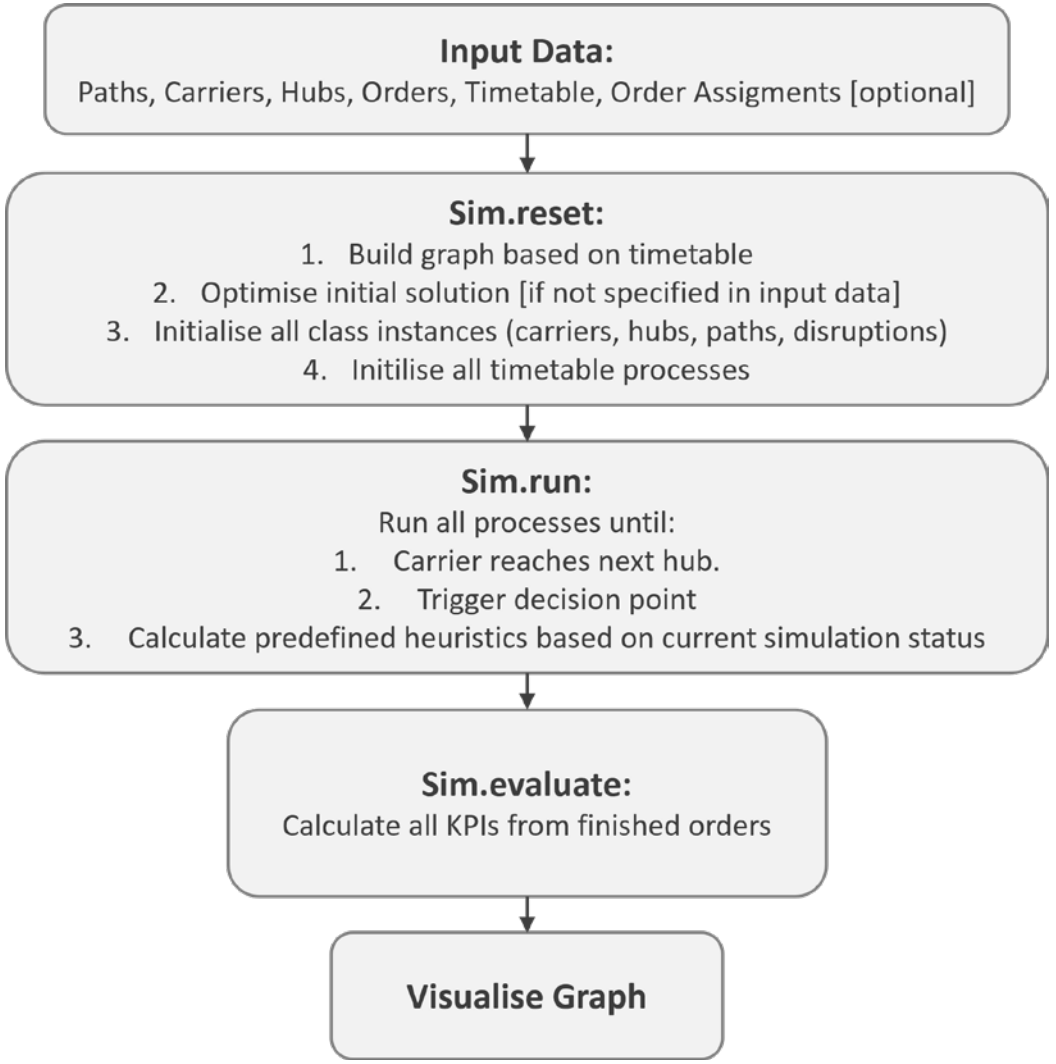



**Figure 2: The overarching workflow of the simulation**

## 2.3 Input Data

The input data is designed as the interface to implement the transport network of interest, including the list of transport orders to be transported through the network.

An overview of the input data is given in Fig.3.

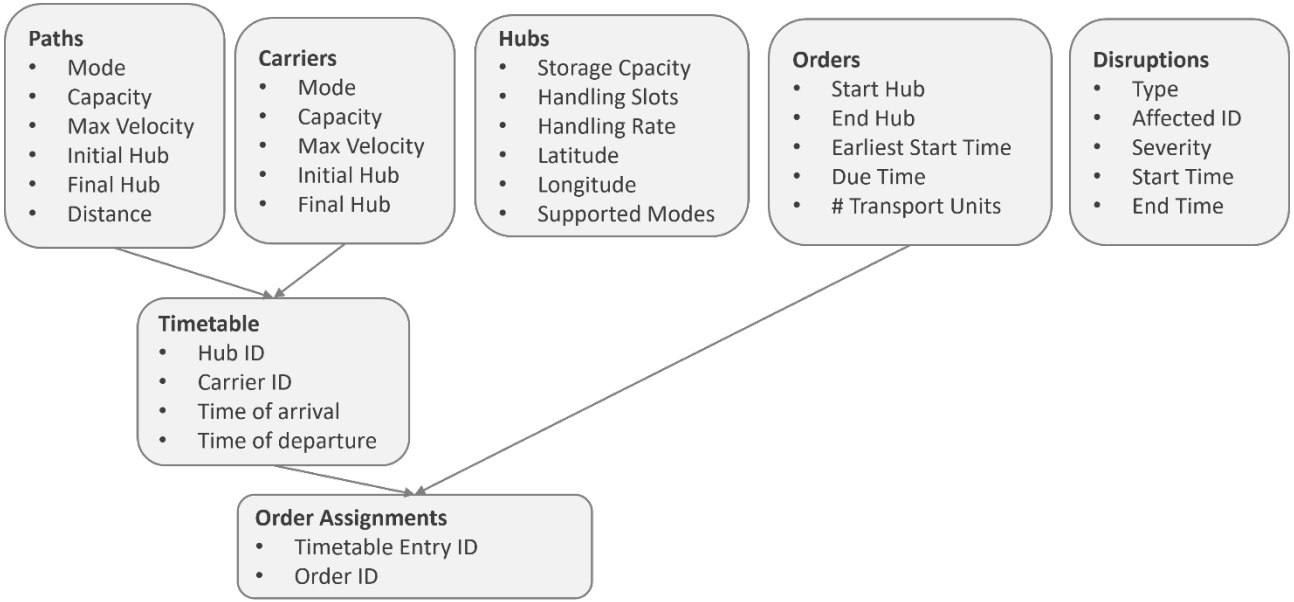
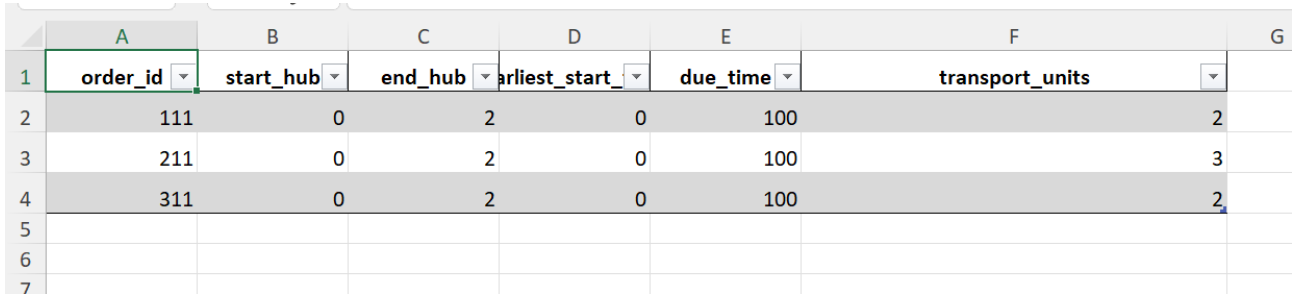



**Figure 3: An overview of required input data**

This data is all given in an associated excel file input_data.xlsx and an example of how this is formatted in given below in Figure 4.



| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | order_id | start_hub | end_hub | arliest_start_ | due_time | transport_units | |
| 2 | 111 | 0 | 2 | 0 | 100 | 2 | |
| 3 | 211 | 0 | 2 | 0 | 100 | 3 | |
| 4 | 311 | 0 | 2 | 0 | 100 | 2 | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |

**Figure 4: A screenshot of the input_data.xlsx which defined the parameters of the relevant transport network.**

The complete data from this input_data.xlsx is directly read into the simulation, is tested to ensure the formatting is correct and is then used to initiate the simulation (see Section 2.4).

## 2.4 Background Graph

The first step in initialising the simulation is to build the graph which represents both the geographical representation of the transport network (latitude and longitude of hubs and carriers) but also the time representation (when do carriers leave and arrive at hubs). The first step in initializing the simulation is to build the graph that represents both the geographical layout of the transport network (latitude and longitude of hubs and carriers) and the temporal structure (when carriers depart from and arrive at hubs). This graph is created using the *SimulationGraph* class (located *in remunet_sim/sim/graph*).

The graph is constructed with the *rustworkx* library for efficient graph operations. It contains three main types of nodes:

- **Hub nodes (*HubNodeData*)**: Represent physical hub locations in the network.
- **Carrier departure nodes (*CarrierDepartureNodeData*)**: Represent a specific carrier departing from a hub at a specific time.
- **Carrier arrival nodes (*CarrierArrivalNodeData*)**: Represent a specific carrier arriving at a hub at a specific time.

Edges in the graph represent:

- **Transport legs**: Movement of a carrier between two hubs (from a departure node to an arrival node). These edges contain attributes such as distance, travel_time, CO2, cost, and capacity.
- **Waiting/transfer**: Remaining at a hub or transferring between carriers (connecting arrival nodes to departure nodes at the same hub or connecting hub nodes to carrier nodes).

This time-expanded graph structure allows the simulation to solve routing problems that inherently depend on schedules and the time-dependent availability of transport resources.

The plot (Figure 5) demonstrates the time-expanded graph structure of the physical transport network, which forms the basis for the simulation. This graph contains three main node types – hub nodes (physical locations), carrier departure nodes (departure of a means of transport) and carrier arrival nodes (arrival of a means of transport). The edges represent either transport legs between hubs, including attributes such as distance, CO2 and costs, or waiting/transfer processes, which can be used to solve routing problems that depend on timetables and time-dependent resource availability.

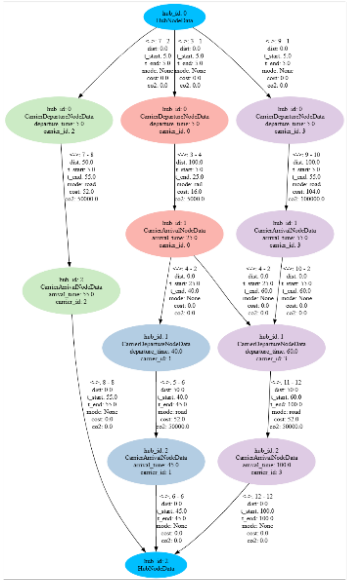


Figure 5: Background graph

## 2.5 Original Solution

We now have our transport network implemented digitally in our simulation and we can begin to solve our problem of optimising the re-planning in the case of disruptions. The first step is to find the original (near) optimal solution – which we assume would remain so if no disruptions would occur.

The process to generate this solution is as follows:

- **Candidate route generation**: Using the Background Graph (*SimulationGraph*), possible routes between start and end points are identified.
- **Evaluation**: These routes are evaluated using cost functions and sorted according to their arrival times and costs. Then, the top 25 Routes are chosen for each Order.
- **Global optimization**: A Mixed Integer Linear Programming (MILP) approach is used to select the best combination of routes for all orders, respecting capacity constraints. This results in an approximately globally optimal solution for the disruption-free case.

Disruptions are then introduced on top of this optimized plan during the simulation. These disruptions (e.g., delays) cause connections to be missed, necessitating replanning. The heuristics described in Section 2.6 are used dynamically during the online phase to handle these disruptions. While a manual approach or a simple baseline might rigidly stick to one heuristic (potentially leading to high costs or lateness), the goal of the RL agent is to learn the best heuristic to apply in each specific disruption scenario.

**Weaknesses and challenges:** The primary weakness of the original solution lies in its assumption of an ideal, disruption-free world. Because it is optimized for this ideal case, the original solution is fragile. It leaves very little room for delays or unexpected events, so even small disruptions force the system to replan immediately. This makes the online phase challenging, which then has to “repair” a schedule that was not designed to handle problems caused by disruptions.

The heuristics used for replanning also have limitations. They only look at the affected order and try to find the best local fix, without considering the wider network or the impact on other orders (such as increased congestion). Since each order is replanned individually, the result is always less optimal than a full global re-optimization would be.

The RL agent tries to balance these trade-offs while keeping orders on time, but it is still limited by the fact that it can only choose among these locally focused heuristics.

## 2.6 RL Algorithm

Using the solution of Section 2.5 we can now run the simulation. Based on the original solution the orders are assigned to carriers and starting the simulation starts the clock and we can track the transport of orders on carriers through the network. As each carrier comes into a hub, the simulation is stopped, the SimKPIs are calculated and the agent is asked the question: "Given the current global state (locations, time, known delays), should the order be re-routed and if so, which heuristic should be used?" For more information see Deliverable 4.2 (chapter 4-5: KPIs & heuristics)

This decision making process is not a simple static rule, but a dynamic interaction loop:

- **Pause and observe:** When a carrier arrives, the simulation pauses. The system compiles a snapshot of the current situation (the Observation), including the order's deadline status and the current network bottleneck.
- **Decide:** The RL agent's trained policy network processes this observation and outputs a probability distribution over available strategies, selecting the most appropriate heuristic (the action).
- **Act and re-plan:** The selected heuristic (e.g., " HeuristicCo2") is executed on the current state of the background graph (see Figure 6) This calculates a new, feasible path for the order from its current location to the destination.
- **Resume**: The new path assignments are fed back into the simulation, and the clock resumes until the next decision point.

At the beginning of training the agent makes random decisions with a random policy. Following the RL algorithm, this decision making policy is updated to improve the total reward. The RL algorithm implemented is Proximal Policy Optimization (PPO) from the *Stable Baselines 3* library, chosen for its stability and efficiency.

The plot below (Figure 5) demonstrates the learned decision-making strategy (policy) of the hybrid reinforcement learning (RL) agent for a simplified problem involving the choice between the 'fastest route' and the 'low cost route'. This strategy shows how the agent makes its decision based on the current location (Current Location, h1 or h2) and the time remaining until the delivery deadline (Time to Deadline), choosing the fast option (e.g. lorry) only if it is necessary to meet the deadline, otherwise preferring the more cost-effective and environmentally friendly option (e.g. train).

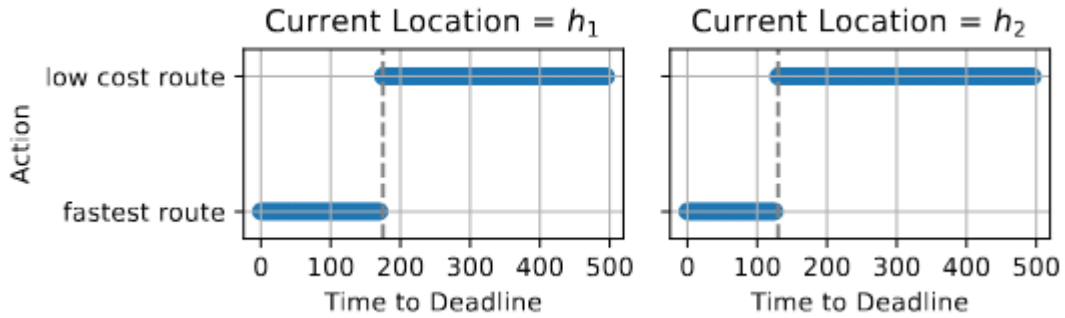



**Figure 6: Learned decision policy of the RL agent**

The decision making process relies on a defined interface between the agent and the environment:

- **Observation:** The agent "sees" the world through a simplified feature vector. This includes the temporal status (time remaining vs. deadline), spatial status (current, start, and end hubs), and a feasibility flag for the current plan using the original heuristic.

- **Action:** Based on this observation, the agent selects a discrete action corresponding to one of the available routing heuristics (e.g., *HeuristicCheapest*, *HeuristicEarliestArrival*, *HeuristicCo2*). Crucially, this is not just a label selection but triggers a real-time pathfinding calculation (A*) using the chosen cost function, allowing specific responsiveness to the disruption (e.g., pivoting to a "Fast" heuristic to minimize delay penalties).
- **Reward:** The learning signal maps logistics goals to scalar values. It incentivizes on-time delivery (+10 bonus) while heavily penalizing lateness (-10). A continuous reward structure also encourages modal shift by penalizing road transport and rewarding sustainable modes like Rail and Water when schedule slack permits.

To ensure better performance, we move beyond manual parameter selection and utilize Optuna for automated Hyperparameter Tuning. Using a Tree-structured Parzen Estimator (TPE) sampler, the system efficiently searches the hyperparameter space—sweeping over critical values like *learning_rate*, *batch_size*, gamma (discount factor), and *ent_coef* (entropy coefficient). The tuning process runs multiple trials, training candidate agents for a fixed number of episodes and selecting the configuration that maximizes average cumulative reward, ensuring the learning dynamics are specifically tailored to the simulation's characteristics.

Operating as an autonomous dispatcher, the agent aims to learn to optimize transport flows by continuously observing order states - such as remaining time and hub location - and selecting the most appropriate high-level routing heuristics for dynamic conditions. Through iterative interaction with the environment, the policy converges on a strategy that maximizes a multi-objective reward function, effectively learning to balance on-time delivery performance with the minimization of transport costs and CO2 emissions.

Examples of a training run is shown below.

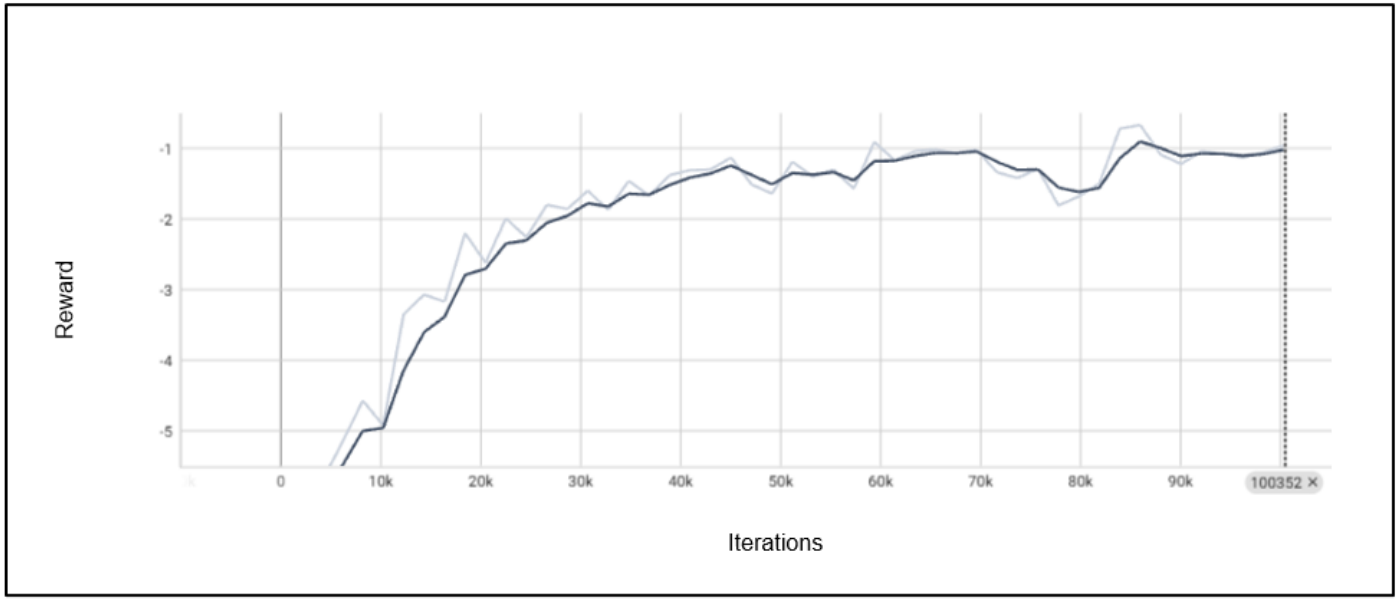



Figure 7: Evolution of Mean Reward during PPO Agent Training

# 3. Input Data and Parameterisation of Scenarios

## 3.1 Review of required input data

The input data defines the entire context for a simulation run. It is structured into several key components, as defined in the InputData class (remunet_sim/data/input_data.py) and loaded from an Excel file.

The data includes:

- Hubs: Physical locations with properties like storage capacity, handling rate, and coordinates (Latitude/Longitude).
- Carriers: Transport vehicles (Trucks, Trains, Ships) with attributes like mode, capacity, CO2 emissions per km, and maximum velocity.

- Paths: Connections between hubs for specific transport modes, defining distances and average speeds.
- Timetable: A schedule defining when specific carriers depart from and arrive at hubs, forming the backbone of the time-expanded graph.
- Orders: Transport requests with a start hub, end hub, quantity (transport units), earliest start time, and a due time (deadline).
- Order Assignments: Initial mappings of orders to specific timetable entries (the "original solution").
- Disruptions: Pre-defined or stochastically generated events that reduce capacity or cause delays (e.g., SimulationPathDisruption, SimulationHubDisruption).

## 3.2 Description of implemented scenarios

The scenarios serving as a testbed for the implemented algorithms are based on the European logistics network in the Rhine-Danube corridor. The Rhine–Danube corridor is a major east–west multimodal freight axis linking North-Sea ports and Western Europe to Central/Eastern Europe. It combines inland waterways (Rhine, Main, Main-Danube Canal, Danube), long-distance rail, major motorways and a network of ports, intermodal terminals and logistics hubs across several European countries such as the Netherlands, Germany, Austria, Hungary and others. The implemented testing scenarios model the part of the network that extends from the Netherlands to Hungary by distinct timetabled connections (see section 2.3.) covering multimodal movements using barge, rail and road carriers. Each connection lists ordered hub stops with arrival and departure times. The schedule allows for two main viable connections between West and East with multiple redundancies and alternatives. For demonstration purposes, some of these connections are selected such, that they become infeasible if carriers are delayed too much, making it necessary to reschedule. Furthermore, the scenarios consider orders that start at the western end of the network and have to be shipped to the eastern end. During the simulation randomly sampled disruptions can occur, for example due to extreme weather events or congestion. Based on this, different scenarios are created by varying the intensity of occurring disruptions.

## 3.3 Explanation & Outlook – Potential integration into the platform

While the developed Reinforcement Learning (RL) framework successfully validates the concept of autonomous synchromodal re-planning, it is currently architected as a research prototype and is not yet suitable for direct integration into the future ReMuNet platform. The system is designed to prioritize training flexibility and experimental control, relying on monolithic, in-memory simulations that are not built to scale

with the high-concurrency demands of a live logistics network. Transitioning this solution to a production environment would require significant architectural refactoring - specifically, decoupling the system state processing from the simulation loop to create a lightweight, scalable microservice, and replacing the current static data processing with real-time stream ingestion capabilities. As such, the current implementation serves as a functional proof-of-concept rather than a deployment-ready module.

To sum up, our framework for autonomous synchromodal re-planning meets the required TRL 4.

# The project

ReMuNet identifies and signals disruptive events and assesses their impact on multimodal transport corridors. It reacts quickly and seamlessly upon disruptive events in real-time. It supports TMS providers to improve route planning resilience. ReMuNet communicates alternative, pre-defined, multimodal transport routes to logistics operators and subsequently to truck drivers, locomotive drivers and barge captains. Through this, it enables a faster and adaptive multimodal network response. ReMuNet orchestrates route utilization, suggests transshipment points and optimizes capacity allocation, minimizing damage and shortening the recovery time. What is ReMuNet's core objective? As trailblazer for the Physical Internet, ReMuNet pursues the vision to enable and incentivize synchro-modal relay transport on European rail, road, and inland waterways to increase the holistic network resilience. It significantly reduces emissions and boosts freight transport corridor efficiency in case of disruptive events. Collaboration with stakeholders is essential to ensure Europe-wide practicability and acceptance.

Coordinator: FORSCHUNGSINSTITUT FUER RATIONALISIERUNG (FIR)

| PARTNER | | SHORT NAME |
|---|---|---|
| fir RWTH Aachen University | FORSCHUNGSINSTITUT FUER RATIONALISIERUNG | FIR |
| HANKEN | SVENSKA HANDELSHOGSKOLAN | HANKEN |
| PTV GROUP | PTV PLANUNG TRANSPORT VERKEHR GmbH | PTV |
| 4PL Intermodal | 4PL INTERMODAL GMBH | INT |
| MANSIO truck meets truck | MANSIO GMBH | MAN |
| Fraunhofer AUSTRIA | FRAUNHOFER AUSTRIA RESEARCH GMBH | FHA |
| HAFEN WIEN ein unternehmen der wienholding | HAFEN WIEN GMBH | HWI |
| WHITE | WHITE RESEARCH SRL | WRE |
| UIRR UNION INTERNATIONALE POUR LE TRANSPORT COMBINE RAIL-ROUTE | UNION INTERNATIONALE DES SOCIETES DE TRANSPORT COMBINE RAIL-ROUTE SCRL | UIR |
| CONTARGO® trimodal network | CONTARGO GMBH & CO KG | CON |
| vedia | VEDIAFI OY | VED |

| | | |
|---|---|---|
| RØDE KORS | DANSK RODE KORS (DANISH RED CROSS) | DRC |
| FMI | ILMATIETEEN LAITOS | FMI |
| alice | ALLIANCE FOR LOGISTICS INNOVATION THROUGH COLLABORATION IN EUROPE | ETP-ALICE |
| SCHACHINGER LOGISTIK | SCHACHINGER IMMOBILIEN UND DIENSTLEISTUNGS GMBH & CO OG | SCH |