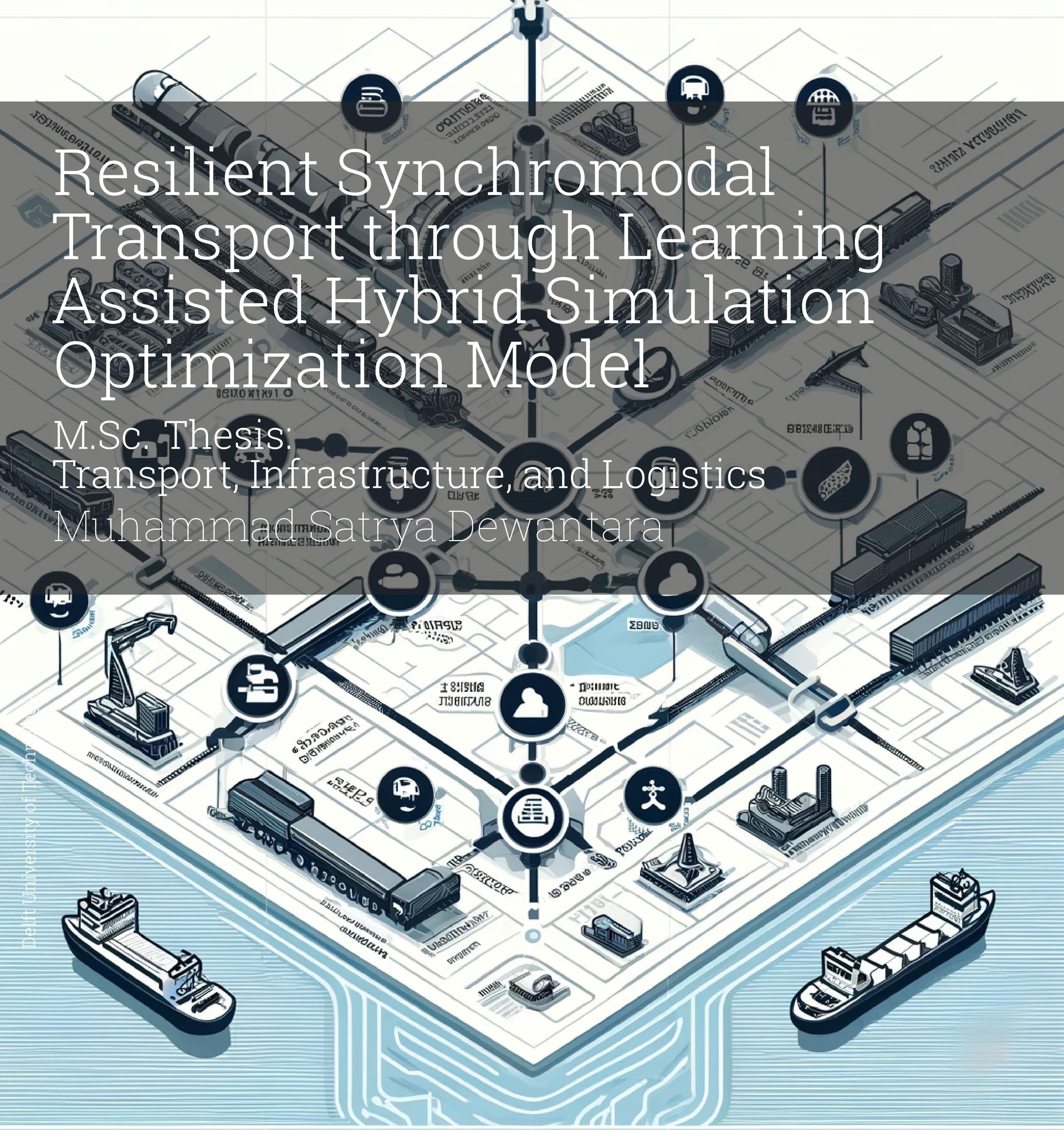


Resilient Synchromodal Transport through Learning Assisted Hybrid Simulation Optimization Model

M.Sc. Thesis:
Transport, Infrastructure, and Logistics

Muhammad Satrya Dewantara

Resilient Synchromodal Transport through Learning Assisted Hybrid Simulation Optimization Model

by

Muhammad Satrya Dewantara

Student Name	Student Number
Muhammad Satrya Dewantara	4861159

Master of Science in Transport, Infrastructure and Logistics
to be defended on August 28th 2024

Supervisors:	Dr. Bilge Atasoy
	Dr. Ir. M. Saeednia
Faculty:	Faculty of Civil Engineering and Geoscience
Report Number:	2024.TIL.8970

Preface

My primary motivation for pursuing a master's degree was to learn coding and to gain a deep technical understanding of freight and logistics—objectives that TU Delft has fully met. I never imagined that I would be creating a simulation of a complex system, let alone integrating a machine learning agent into it. These achievements have been possible thanks to the unwavering support of many individuals throughout my time at TU Delft, particularly over the past five months while working on this thesis project.

I would like to express my heartfelt gratitude to those who made the completion of this thesis possible. My sincere thanks go to my daily supervisor, Dr. Ir. Mahnam Saeednia, who introduced me to this topic and encouraged me to push beyond my comfort zone. Together with my chair, Dr. Bilge Atasoy, they provided exceptional guidance and valuable insights throughout the process. I am also deeply appreciative of Dr. Saiedeh Razavi and Siyavash Filom for their collaboration on the integration work, as well as Dr. Neil Yorke-Smith for the discussions that significantly strengthened my understanding of reinforcement learning. To my family and friends, thank you for your intangible support, and to Puitry Neurita, thank you for always being there when I needed it most.

Lastly, it is meaningful to me that this thesis will be defended on August 28th, which would have been my late father's birthday. I hope this work serves as a fitting birthday present for him. I believe I have put forth my best effort into this thesis, and I hope it offers even a small glimpse of inspiration to those who read it, offering something valuable to learn. As David Silver said, ”Life is one big training set.”

Muhammad Satrya Dewantara
Delft, August 2024

Summary

Synchromodality is a concept developed to address the growing freight trade and its dynamics. Its objective is to thrive in the highly competitive transportation market and meet growing customer demands by enhancing flexibility and offering more customized services. This flexibility attribute enables real-time mode shifts to respond and adapt to unexpected circumstances in the uncertain and competitive market. To fully leverage this flexibility, certain requirements must be met. It is suggested that Logistics Service Providers (LSPs) can make the most of this feature when shippers agree to mode-free or a-modal requests. In this scenario, LSPs have the freedom to choose the mode of transport that best suits the cargo's delivery, provided it meets the customer's requirements. This flexibility is a significant departure from traditional transportation approaches, offering a responsive and adaptable solution in the face of dynamic market conditions. The flexibility allows the services to adapt and react to disruptions which will be the focus of this thesis.

The main objective of this research is to propose a learning-assisted model under synchromodal framework in creating a resilient multi-modal transport, addressing the unknown duration of disruption in port-inland freight transportation. The model is designed to capture disruption scenarios and provide reaction strategies to maintain its performance. On top of that, a learning approach is incorporated to enhance the model performance in providing the reaction strategies. While it has the potential to improve the performance of synchromodal framework, a thorough literature review shows that only a few studies propose the integration of a learning approach in the synchromodal framework.

The model comprises three main modules: Simulation Module, Optimization Module, and Learning Agent. The scope of the simulation only covers the inbound shipment from the main port to the inland terminal neglecting the return trip back to its origin. The main objective of this simulation is to capture the dynamic nature of disruptions in the hinterland freight network representing real-world operations to create an environment for implementing a decision support system. 5 Profiles of service disruptions and 3 profiles of demand disruptions are developed and applied to the network.

Upon receiving a new request or detecting a disrupted shipment, the optimization module is triggered to initiate the planning or re-planning process. The matching decision involves assigning a service line to the shipment. Additionally, the optimization module proposed in this research enables various analytical methods to plug-in to serve as a central planner.

A reinforcement learning technique is integrated into the model, allowing it to decide whether to wait or reassign according to the solutions suggested by the optimization module. This learning approach enables the model to make more informed decisions, balancing the benefits of reassignment with the potential advantages of waiting. The integration of simulation-based synchromodal framework with a reinforcement learning agent is the novel idea proposed in this research.

The Learning Assisted Model is implemented in a case study. A service network is adapted from the European Gateway Services (EGS) network, now known as Hutchinson Ports Europe Intermodal (HPEI). This network connects the Hutchinson Ports ECT in the Port of Rotterdam to inland terminals in the Netherlands, Belgium, and Germany through the Rhine-Alpine corridor. Demand is generated randomly based on the service capacity proportion of each origin-destination pair within the network.

a matching problem optimization algorithm from an existing study is integrated as a plug-in to the optimization module and acts as the centralized planner. To address a run-time issue, a solution pool is provided during the planning, therefore reducing the number of triggering the optimization model.

The complete model is trained through 20,000 episodes before the implementation. In the implementation, the learning-assisted model uses a *greedy policy* by choosing the action with the highest value function. The *greedy policy* is evaluated by comparing it to the *always wait* as the benchmark policy. *Always wait* means, the affected shipments always stay with the given itinerary and wait until the disruption ends

The results show that the greedy policy by the RL agent outperforms the benchmark policy in 16 out of 20 simulations of different disruption occurrences. the cost savings by the learning-assisted model range from 0.08% to 15.96%. If the calculation only considers the affected requests, the highest savings from greedy policy over benchmark policy reaches 35.7%. The amount of savings depends on the severity of delays caused by the disruptions in the network. Furthermore, the experiment of different disruption profile sets shows that the RL agent still outperforms the benchmark policy on different sets of occurrence rates ranging to 30% difference from the one used in the training.

There are important limitations in each module that affect the quality of the final model including limited disruption data, limitations in the algorithm used for matching algorithms, and value function updating for the RL agent. The implications of those limitations are the run-time issues and the required episodic training. Ultimately, this affects the quality of actions taken during implementation.

The main finding of this research is that integrating Reinforcement Learning (RL) into a synchromodal framework could enhance the resilience of port-inland freight transportation. This is evidenced by the superior performance of the RL-integrated model compared to the benchmark policy (*Always Wait*). Additionally, the model demonstrates consistent resilience across various disruption scenarios, as indicated by its stable performance under a certain extent of different occurrence probabilities from those used in training.

While there is room for improvement, the results suggest that this model could serve as a foundation for future research. Potential research directions include implementing communication schemes among RL agents for different shipments, exploring deep reinforcement learning techniques, and refining disruption profiles. These enhancements could further improve the model's effectiveness and adaptability.

Contents

List of Figures

List of Tables

1 Introduction

1.1. Background and Problem Definition

Hinterland freight transport plays an essential role in logistic activities. Despite the fact that container shipments spend most of their time on deep sea transport in the global supply chain, hinterland transport takes a cost proportion ranging from 40-80% from the total shipping costs of container (Notteboom and Rodrigue, 2005) showing its significance in the whole chain. Most ports around the world have recognized the hinterland connection as one of the most important issues (Merk and Notteboom, 2015) To connect the main port to its hinterland, containers could be transported by road (truck), railway (train), waterway (barge), or a combination of them.

Road transportation dominates inland freight movement, accounting for 77% of the EU's freight in 2020 Eurostat (2020). Logistic Service Providers (LSPs) often prefer unimodal transport primarily due to its inherent reliability Tavasszy et al. (2015). However, road transport introduces several externalities, including accidents, road damage, environmental harm, and congestion (Santos et al., 2010). The Intermodal concept was introduced to shift the freight flow to other transport modes, such as barge and train. It can offer a cheaper option for inland freight transport due to the economics of scale Tavasszy et al. (2015). Despite its cost-effectiveness, the share of intermodal transport remains low due to its lack of flexibility. The concept of synchromodality aims to increase the attractiveness of intermodal transport. By having a flexibility feature within the intermodal transport, synchromodal transport provides a higher number of combined routes, thus creating added values in the trade-off between price and time to the shippers Tavasszy et al. (2015).

Another reason why conventional intermodal is less preferred compared to trucks is the disruptions and variations in the network due to uncertainties (Delbart et al., 2021). Disruptions and uncertainties negatively contribute to the efficiency of conventional intermodal transport (Delbart et al., 2021) and could cause a severe economic loss. Disruption with low occurrence probability but high impact such as COVID-19 increased the logistic cost by 12% globally (Rodríguez-Clare et al., 2023). From another spectrum where disruption occurs frequently, a study by (Schlake et al., 2011) estimates $15.2 Million loss in a year due to train delays. The estimation made in 2011 is likely to have a significantly higher value today.

The flexibility of trucks in terms of route and schedule allows them to better handle uncertainties in the road network, while it is not the case in intermodal transport where the schedule of barge and train are generally fixed. In addition to the added values in the price and time trade-off, the flexibility feature in synchromodality concept equips the multi-modal transport system to react better to disruptions if compared to the conventional intermodal. For instance, theoretically, the containers that are supposed to be carried by barge could be shifted to train if there is a disruption in the barge network.

The type of disruptions in the network vary and could appear both on the supply and demand side. On the supply side, examples of unexpected events are labor strikes, accidents, or broken infrastructure

(Delbart et al., 2021). Meanwhile, the example of disruption in the demand side is, in the hinterland transport, the change of release time of containers in the port due to customs issues. Each disruption type may have a different frequency and severity to the logistic chain (Ambra et al., 2019). In dealing with disruptions, different reactions could be taken. As mentioned earlier, containers could be reallocated to other mode service types (Guo et al., 2020) or wait for the next departure in the same mode service (Hrušovský et al., 2021).

This research investigates how to create a resilient intermodal freight transport system that leverages the flexibility offered by synchromodality. The resilience of a freight transport network is defined by its ability to recover from disruptions and is measured by the effort required to restore normal operations (Chen and Miller-Hooks, 2012). It provides a learning-based modular framework capable of capturing the disruptions and generating a reaction plan. Developing the comprehensive model as mentioned above with practical solutions creates a challenge for scholars. Not only filling the gap of knowledge, the model should be practical to be implemented at the industry level, thus, shifting the freight transport paradigm away from road dependency towards more sustainable and flexible options. The modular framework utilizes optimization, simulation, and machine learning techniques allowing plug-and-play possibility for connecting with different existing/under development models and is extensible, making it applicable to different eco-systems of port-inland connections, expected to improve the solution space in generating decisions to react to the disruptions.

1.2. Research Gap

Synchromodal transport in inland freight transport has been studied for the last couple of years. Several static models have been well investigated under the synchromodal framework. However, static models have their limitations when it comes to addressing the real-time planning and uncertainties in synchromodal systems (Qu et al., 2019).

Dynamic models for synchromodal transport have also been proposed in several studies. These researchers have different approaches to capturing the uncertainties or disruptions and propose a real-time solution by utilizing the flexibility feature in synchromodal framework. Existing studies of synchromodal by Ambra et al. (2019), Di Febbraro et al. (2016), Hrušovský et al. (2021), and Layeb et al. (2018) propose simulation-based models that are capable of capturing the dynamic behavior and disruptions in the network. However, while these simulation-based models offer valuable insights, none have thoroughly explored reaction strategies of container reallocation or rerouting as proposed by Guo et al. (2020). Given the difficulty in altering the service network in reaction to disruptions, the combination of a simulation-based model and container reallocation strategy is an interesting gap to address in this thesis.

Furthermore, recent works by Guo et al. (2022), Larsen et al. (2023b), and Zhang et al. (2023) show evidence that learning approach can improve the performance of the synchromodal framework. Integrating the learning approach into the decision support system could provide a powerful tool for dealing with disruption. In reality, the duration of disruption is usually unknown, thus creating another challenge in the decision making of reaction strategies. Therefore, this thesis considers to incorporate the learning approach on top of the simulation-based model to see how it improves the resilience of the synchromodal transport. The comparison of these related studies with the model proposed in this thesis is presented in Table 1.1.

From those studies, the closest modeling approach to this thesis are the ones proposed by Guo et al. (2022) and Zhang et al. (2023). Nevertheless, there are essential differences either from conceptual or methodological perspectives between this thesis and those two studies.

First, none of those studies use a simulation-based approach to capture the dynamic behavior of the disruption in the multimodal transportation network. The similarity of this thesis with those two studies is the integration of reinforcement learning (RL) approach, which leads to the second argument. This thesis employs an RL agent to assist the replanning module in response to disruptions, whereas the model proposed by Guo et al. (2022) serves as a centralized platform responsible for the entire shipment planning process. Third, the action space of the RL agent in this thesis is to select between 'wait'

Table 1.1: Overview of Studies Proposing Related Models

Source	Service Disruptions	Request Disruptions	Simulation -based	Learning Agent	Cost-based Reward system
(Ambra et al., 2019)	v		v		
(Di Febbraro et al., 2016)	v		v		
(Hrušovský et al., 2021)	v		v		
(Guo et al., 2020)		v			
(Guo et al., 2021)	v	v			
(Qu et al., 2019)	v	v			
(Larsen et al., 2021)	v				
(Larsen et al., 2023a)	v				
(Larsen et al., 2023b)	v			v	
(Guo et al., 2022)	v	v		v	v
(Zhang et al., 2023)	v			v	
This thesis	v	v	v	v	v

or 'reassign', with the optimization model narrowing down the solution space. In contrast, Guo et al. (2022) covers all possible matches between shipments and services in the solution space.

The above arguments bring this thesis closer to the study by Zhang et al. (2023). However, the reward system in this thesis is different from the model proposed in that study. Zhang et al. (2023) proposes the RL agent is granted a discrete reward either 1 or 0 based on whether or not the delay occurs due to the disruptions with respect to the taken actions. For example, if the action taken is to wait, and the disruption does not cause delay, the RL agent receives a reward of 1, and 0 if it removes the shipment from the assigned service. Consequently, this approach does not differentiate rewards based on shipment volume, nor does it account for the amount of delay time. Conversely, this thesis proposes a negative cost value as the reward. By having this, the same delay with higher shipment volume is penalized more, enabling the RL agent to prioritize shipments with higher volume.

1.3. Research Approach

To address the problem in the research gap, the research objective along with the research questions are determined to provide clearer direction for this thesis.

1.3.1. Research Objective and Contribution

The main objective of this research is to propose a learning-assisted model under synchromodal framework in creating a resilient multi-modal transport, addressing the unknown duration of disruption in port-inland freight transportation. The model is designed to capture disruption scenarios and provide reaction strategies to maintain its performance. On top of that, a learning approach is incorporated to enhance the model performance in providing the reaction strategies. While it has the potential to improve the performance of synchromodal framework, a thorough literature review shows that only a few studies propose the integration of a learning approach in the synchromodal framework.

1.3.2. Research Question

In accordance with the objective, research questions are formulated as follows:

"To what extent does a learning approach improve the resilience of a synchromodal framework in coping with disruptions?"

This main research question is accompanied by a series of sub-questions designed to provide a comprehensive understanding of the subject matter:

1. What are the types of disruptions in the synchromodal transport and what strategies are applied to react to them?
2. How can a synchromodal framework under disruption be adequately modeled?

3. How can reaction strategies under synchromodal framework be modeled in response to the disruptions?
4. How can a learning approach be incorporated into the synchromodal framework to provide a better solution space?
5. How does the learning-based synchromodal framework perform under the disruptions?

This thesis will construct a simulation-based model supported by a learning approach under synchromodal framework focusing on hinterland transportation. The model provides a modular structure that enables the integration with optimization models with different objectives for hinterland transport planning to generate reaction strategies in response to disruptions. The contribution of this study will be a novelty of a model, a combination of hybrid simulation-optimization assisted by a learning approach under synchromodal framework.

1.3.3. Methodology

The main research question and its sub-questions will be answered by applying a series of methodologies as elaborated in the table 1.2

Table 1.2: Methodology

SRQ	Method	Remarks
1	Literature review	The literature review will generate type and classification of disruptions to be incorporated in the model
2	Simulation model	Simulation model using discrete-event system will be employed to mimic the synchromodal operation
3	Hybrid simulation-optimization model	Hybrid model to connect the simulation with an optimization model
4	Learning approach	Develop algorithm of learning approach to enhance the reaction strategy
5	Case study	Test the model under different scenarios to measure the performance

SRQ: Sub research question

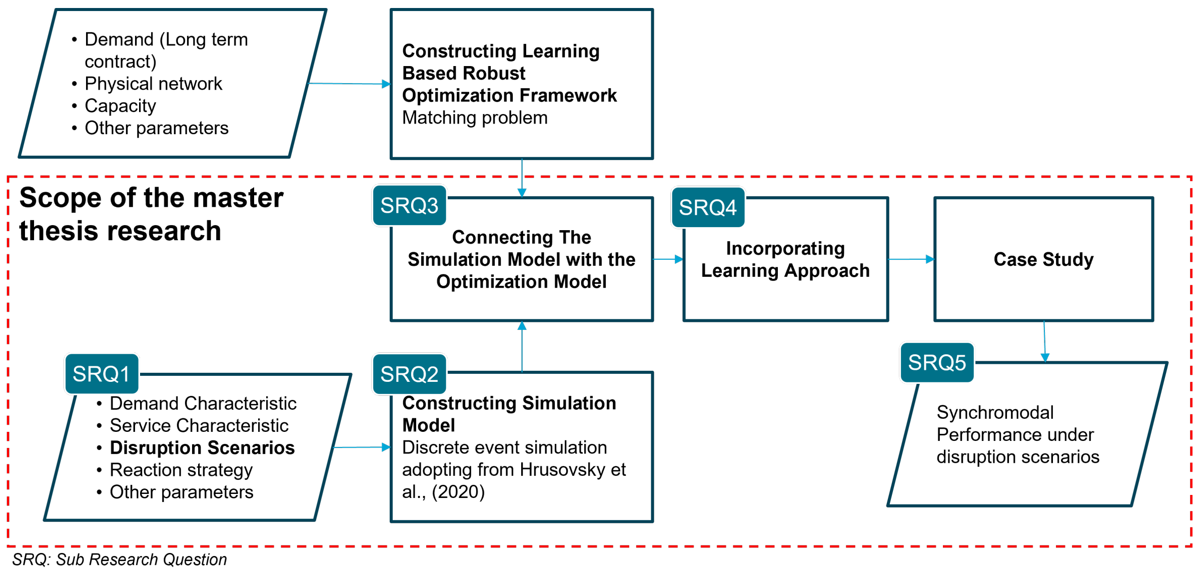
In summary, there are five methods performed for all the sub-questions including literature review, simulation model, hybrid model, learning approach, and the case study. It is important to note that the optimization model formulation is not part of this research. Instead, the simulation model is constructed to be capable of integration with an available optimization model.

Literature Review

In the initial phase before constructing the model, a literature review is undertaken to gain an understanding of disruptions in synchromodal network. The identified disruptions will be categorized and used in the later model. The categorization could encompass aspects such as frequency, duration, and distribution of occurrence. Moreover, the literature review serves to gain insight of how synchromodal framework is modeled

Simulation Model

The first step in constructing the dynamic model of synchromodal framework is a simulation model. A discrete-event simulation will be employed to mimic the process in the synchromodal network and also capture the disruptions both in supply and demand. Supply network of port-hinterland will be constructed including the location of the inland terminal and service schedule of barge, truck, and train. The demand will be a shipment request from the terminal. The simulation model is formulated to be a modular structure, thus, enabling the plug-in of an optimization model.



• Demand (Long term contract)
• Physical network
• Capacity
• Other parameters
Constructing Learning Based Robust Optimization Framework
Matching problem
Scope of the master thesis research
SRQ3
Connecting The Simulation Model with the Optimization Model
SRQ4
Incorporating Learning Approach
Case Study
SRQ1
• Demand Characteristic
• Service Characteristic
• Disruption Scenarios
• Reaction strategy
• Other parameters
SRQ2
Constructing Simulation Model
Discrete event simulation adopting from Hrusovsky et al., (2020)
SRQ5
Synchromodal Performance under disruption scenarios
SRQ: Sub Research Question

Figure 1.1: Methodology

Hybrid Simulation-Optimization

Once the simulation is done, it will be connected to an available optimization model. This approach is adapted from Hrušovský et al. (2021) which combines a simulation and an optimization model. The optimization model will solve a matching problem between the shipment request and the service network, thus providing an itinerary for each request. The request will be used as an input in the simulation model. The optimization model will run in every time interval or if there is a disruption in the network. The disruption will be an input to the optimization model generated by the simulation model.

Learning Approach

This learning approach will also be embedded in the Hybrid Simulation-Optimization model and use input generated from the simulation. The learning approach will focus on the demand management side and will support the optimization model in creating the decision, especially in response to the disruptions. One possible method for the learning approach algorithm is reinforcement learning. Adopted from Zhang et al. (2023), it is possible to combine a learning approach with an optimization model.

Case Study

After the model is completed and verified, the case study is conducted. The case study will use real network data for the supply side to represent the real-world operation. The demand side will depend on the data availability and could use a synthetic dataset. The model is tested under various scenarios to see the performance measured in key performance indicators such as cost, time, or emissions.

Each methodology employed for this research will be completed in steps and illustrated in Figure 1.1

1.4. Report Outline

This thesis report consists of 8 chapters starting with an introduction to define the problem statement and elaborate the research approach. **Chapter 2** elaborates the fundamental theory behind the resilient synchromodal framework and investigates the models in existing studies. **Chapter 4** explains how the synchromodal framework is modeled in this thesis including the hybrid simulation-optimization and the embedded learning approach. **Chapter 5** Verifies the complete model with synthetic data to evaluate if the model work according to the conceptual logic. **Chapter 6** implements the complete model from **Chapter 4** using real-world data and analyze the output with respect to the sub-research question 5. Finally, the discussion about the limitation and result interpretation is presented in **Chapter 7** followed by the conclusion of the whole thesis and recommendation for future research in **Chapter 8**.

2 Literature Review

A literature review is performed to gain insight and understanding about how to construct the model under synchromodal framework. The section starts with an explanation of the concept of synchromodality and the disruptions in the freight network to get the reader in the same context before delving deeper into how it is modeled in various existing studies.

2.1. Concept of Synchromodality

Synchromodality is a concept developed to address the growing and dynamic freight trade. Its objective is to thrive in the highly competitive transportation market and meet growing customer demands by enhancing flexibility and offering more customized services (Tavasszy et al., 2015). In recent years, various studies have aimed to define synchromodality. A study concludes that synchromodality is a planning system of multi-modal transport that integrates supply chain stakeholders to flexibly adapt transport modes based on real-time insights (Acero et al., 2022). Another study describes Synchromodality as an efficient, reliable, flexible, and sustainable service enabled by stakeholders' cooperation supported by real-time information sharing (Giusti et al., 2019). De Juncker et al. (2017) describes Synchromodality as real-time transport planning using real-time information from different parties to increase flexibility, reliability, efficiency, and sustainability.

Notice that synchromodality has been defined in several studies with slight differences. Nonetheless, all the studies always include flexibility and real-time planning with shared information aspects. Acero et al. (2022) identifies four distinct dimensions of synchromodality including visibility, integration, multi-modal transport, and flexibility. In synchromodal environments, visibility is crucial and involves the exchange of real-time information and data among multiple stakeholders to optimize operations (Acero et al., 2022). This data includes demand forecasts and distribution network status from customers, transport network conditions, and timely updates from logistics partners and subcontractors (Acero et al., 2022). This real-time information is integrated among various actors to support mode choice decisions. Unlike the integration seen in conventional intermodal transport, synchromodal transportation relies on horizontal integration, which extends beyond a single logistics chain to encompass different intermodal chains (Tavasszy et al., 2015).

The horizontal integration opens up more options in the network and creates added values to the multimodal transport through a higher number of service line combinations. (Tavasszy et al., 2015) illustrates those added values of synchromodal transport in a two-dimensional chart as shown in Figure 2.1. On the left hand side, the chart shows a classical cost-time trade-off among different modes where truck is the fastest mode with a high price and barge (waterways) is the opposite. The combination of the different modes through intermodal transport creates more choices as represented by T1 to T9 dots in the right chart. With the horizontal integration under the synchromodal framework, the number of mode service combinations could increase dramatically, providing more non-dominated solutions (Tavasszy et al., 2015).

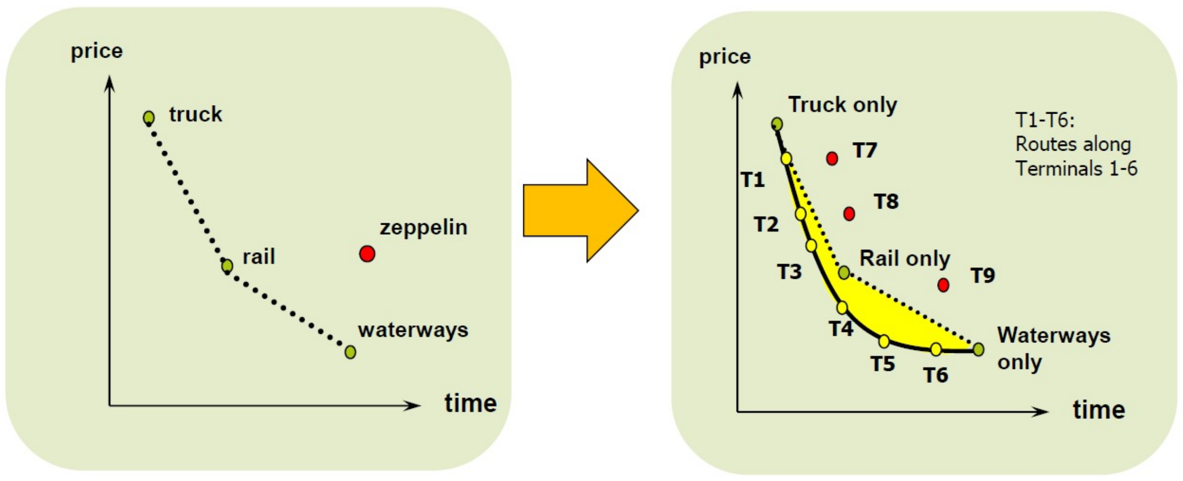


Figure 2.1: Synchromodal Added Values (Tavasszy et al., 2015)

This broader integration fosters flexibility in mode choice, enabling real-time adaptation to changing circumstances. Flexibility allows for real-time mode shifts, helping to navigate unexpected challenges in a volatile and competitive market (Acero et al., 2022). To fully capitalize on this flexibility, certain conditions must be met. It is suggested that Logistics Service Providers (LSPs) can maximize this feature when shippers agree to mode-free or a-modal requests (Acero et al., 2022). In this scenario, LSPs have the freedom to select the mode of transport that best suits the cargo's delivery requirements. This flexibility marks a significant departure from traditional transportation approaches, offering a more responsive and adaptable solution in dynamic market conditions. Beyond the added value from increased mode combinations, flexibility is a key benefit of synchromodal transport, allowing services to adapt and react to disruptions (Giusti et al., 2019), which will be the focus of this research and will be elaborated on further in the following sections.

There are many types of disruptions in the freight transport network, ranging from a natural disaster with less frequency to the operational disruption that can occur more frequently (Wide et al., 2022). In the case of a more frequent disruption, a real-time reactions are necessary. This is how synchromodal transport theoretically performs better under a disrupted network compared to conventional intermodal transport. For example, if there is a disruption in a certain service line, a container can be moved to other available service lines (Hrušovský et al., 2021) as well as transported to a different transshipment terminal with the same end destination (Ambra et al., 2019). This is possible to be done under synchromodal framework due to the real-time visible information and the flexibility feature.

From the implementation perspective, several factors enable the effective implementation of synchromodal transport. Pfoser et al. (2016) identify that sophisticated planning and ICT/ITS technology are key enablers, recognized for their significant importance and high feasibility. Additionally, A-modal booking or modal free booking, as one of the key enablers, has already been adopted by many companies (Alons et al., 2019). Mode-free booking is a type of contract where LSPs are given the authority to combine shipments and use different modes of transport (Acero et al., 2022). These findings provide a positive indication of the progress in synchromodal transport development. However, the implementation of this concept faces challenges, primarily due to the need for cooperation and information exchange among various Logistic Service Providers (LSPs) (Alons et al., 2019). While the industry works towards establishing a robust framework for collaboration, scholars have the opportunity to further explore the potential of the synchromodal framework, thus ensuring optimal benefits when the concept is fully embraced by the industry.

2.2. Disruption in Multi-modal Freight Transportation

The global supply chain relies significantly on freight transportation. Any disruption in the logistic chain could result in severe economic loss to different parties. Take the example of the Covid-19 pandemic which caused a reduction in capacity in the logistic service network due to ports being closed, fewer workers, and container shortage (Rodríguez-Clare et al., 2023). It estimates roughly a 12% increase in the logistic costs globally (Rodríguez-Clare et al., 2023). In inland transport, a study simulates a barge service network under disruption scenarios in the US resulting an increase in the cost of nearly $900 million over a year period (201, 2019). At the operational level, a study estimated a loss of $15.2 million a year in 2011 due to train delays (Schlake et al., 2011). These disruptions in the multi-modal freight transport is discussed in more detail in Chapter 3

2.3. Modelling Resilient Synchromodal Framework

Synchromodal transport in hinterland freight transport has been studied for the last couple of years. Several static models have been well investigated under the synchromodal framework. The static models referred to in this thesis are the analytic models that provide results or solutions based on static information. It could be both a deterministic approach or a stochastic approach such as a robust optimization model. However, static models have their limitations when it comes to addressing the real-time planning and uncertainties in synchromodal systems (Qu et al., 2019). Therefore, this thesis focuses on the dynamic models to address this challenge.

2.3.1. Dynamic Models for Synchromodal Framework

Dynamic models for synchromodal transport have been proposed in several studies. A Synchromodal Transportation Re-planning (STP) for hinterland transport is developed using a mixed integer linear programming (MILP) (Qu et al., 2019). The model is tested and run several times under various types of disruption events including shipment uncertainties and delays in the network by providing shipment rerouting and service rescheduling. The service rescheduling includes the railway service which in practice is quite complex. This model is comprehensive and covers disruptions on both supply and demand sides. However, the model has not yet been integrated into a dynamic environment. From a different approach, a dynamic matching problem is proposed to deal with uncertain shipment requests (Guo et al., 2020). In this model, the shipment requests are not completely known, but rather sequentially announced using a rolling horizon approach. This approach is adopted by another model for a global shipment matching problem and improved by incorporating disruptions in the service network (Guo et al., 2021). The reaction to disruption in these two models are reallocation planning of the containers.

Other studies proposed an agent-based model to simulate the synchromodal framework. An agent-based model is developed to assess the implementation of Synchromodality in retail goods between distribution centers (DC) in Belgium and France. The model compares the performance of unimodal, intermodal, and synchromodal for cost, time, and emissions. The model applies a synchromodal scenario by putting logic for each agent to reroute to the nearest and cheapest terminal if there is a disruption in the network, and monitor the impact on cost, time, and emission (Ambra et al., 2019). Another study proposed an agent-based framework for cooperative planning (Di Febbraro et al., 2016). The model uses decentralized optimization with a negotiation scheme. It breaks down the problem into several sub-problems and lets the agents communicate with other agents to achieve each objective under disrupted scenarios. The model provides a sequence plan and re-plans it when exogenous events occur.

A study uses a centralized model predictive control (MPC) to crate a routing plan for containers and vehicles simultaneously. MPC allows the model to capture the dynamics in the network, including uncertainties in travel time and mode capacity, and make decisions according to predictions (Larsen et al., 2021). This model is enhanced by using a co-planning approach by having an exchange of information realistically between different parties in the logistic chain resulting in a reduction of total container transportation costs as well as the travel distance (Larsen et al., 2023a).

A decision support system is proposed using a hybrid simulation-optimization model under synchromodal framework (Hrušovský et al., 2021). It employs an offline model to create the initial plan and an online model to react to the disruptions and selects one of three possible policies: wait, transshipment, or detour. The disruptions are categorized according to frequency and duration by assigning them to a random variable in the simulation. The online model will be triggered if there is a disruption occurs. The result of the study shows that the transshipment policy has the lowest share in all scenarios. This result could be a subject for future research since transshipment or mode shift plays an important role in Synchromodal. The other policy in this model is to wait, which is essentially the traditional reaction, and detour, which is practically difficult for barge and/or freight trains.

Another simulation-based optimization model under the synchromodal framework is proposed by Layeb et al. (2018). Similar to the model proposed by Hrušovský et al. (2021), the model solves a service network design problem by finding the optimal service schedule in a real case study. It includes the stochastic behavior by assuming random variables following certain distributions for demand and travel time, while not explicitly introducing disruptions in the network. The proposed model shows a 90% on-time and full delivery performance.

In general, there are multiple ways to address disruption on the service network and provide reaction under synchromodal framework. The reaction strategy could be waiting, altering the service network (Hrušovský et al., 2021), or rerouting the container (Guo et al., 2020; Larsen et al., 2021). The summary of the literature review of the dynamic model under synchromodal framework is presented in Table 2.1.

Table 2.1: Available Models of Synchromodal Framework in Literature

Source	**Model**	**Disruption Scenarios**	**Reaction to Disruption**
Ambra et al. (2019)	Agent-based Simulation - Decentralized optimization	Service network	Wait or re-route to the other cheap and close terminal
Di Febbraro et al. (2016)	Agent-based Model - Discrete event system - Decentralized optimization - Negotiation scheme	Service network	Reoptimize the sequence plan
Hrušovský et al. (2021)	Hybrid simulation-optimization - Agent-based simulation - Service network design problem - Centralized optimization	Service network	- 3 Reaction policies: wait, transshipment, detour - Triggered every disruption occurrence
Guo et al. (2020)	Dynamic Optimization - Rolling horizon approach - Matching problem - Centralized optimization	Request/demand	Reallocation every time interval to accommodate new requests
Guo et al. (2021)	Dynamic Optimization - Rolling horizon approach - Matching problem - Stochastic approach - Centralized optimization	Service network and request/demand	Reallocation every time interval to accommodate new requests
Qu et al. (2019)	Replanning Optimization - Service network design problem - Centralized optimization	Service network and request/demand	Used after disruption occurrence. - Shipment flow replanning - Service rescheduling
Larsen et al. (2021)	Model Predictive Control - Simultaneous container and vehicle routing - Consider limited number of trucks in the network - Centralized optimization	Service network	Container and/or truck rerouting
Larsen et al. (2023a)	Co-planning Method - Consider limited number of trucks in the network - Decentralized optimization - Exchange of information in a realistic level	Service network	Container and/or truck rerouting

2.3.2. Simulation-based Models in Freight Transport

Logistics problem-solving typically employs two primary methodologies: Analytical Methods and Simulation Methods (Lyu et al., 2023). Analytical Methods encompass techniques such as linear programming and regression analysis. Among these, Mixed Integer Linear Programming (MILP), a subset of linear programming, excels at generating optimal solutions concerning decision variables and constraints. Analytical methods are frequently harnessed to address logistics issues like routing, scheduling, and location problems. However, they exhibit limitations in capturing the dynamic and transient aspects of models (Lyu et al., 2023).

Another approach is the utilization of simulation models. This category encompasses methodologies such as Monte Carlo Simulation (MCS), System Dynamics (SD), Agent-Based Models (ABM), and Discrete-Event Simulation (DES). All these simulation techniques provide the means to incorporate stochastic elements (Kogler and Rauch, 2018). Nonetheless, DES stands out due to its straightforward model structure, allowing it clearly represent a complex system through a chain of events (Kogler and Rauch, 2018). DES quantitatively represents real-world scenarios, simulating dynamics event by event and generating detailed performance reports (Goti, 2010). However, the result of a simulation model heavily relies on the data scale and accuracy as the model input (Motraghi and Marinov, 2012).

Simulation is a descriptive method and a common approach in the field of freight transportation research to provide a real representation of a freight operation for various objectives. Motraghi and Marinov (2012) develops a simulation model to analyze the existing railway system and to select options from the proposed railway system according to its utilization. Another study by Lyu et al. (2021) proposed a simulation model in combination with a Geographic Information System (GIS) to present the operations of urban pickup and delivery. This model is proposed to help the industry gain an understanding of the existing operations. On a broader scale, an integrated logistics and transportation system is simulated using discrete event simulation (Xu and Hancock, 2004). The simulation encompasses not only the physical aspect of a product supply chain but also the flow of information (Xu and Hancock, 2004).

To extend the benefit of a simulation model, a combination with optimization methods can help the decision makers to assess possible policies for a specific problem (Oliveira et al., 2016), which could be referred to as a decision support system. As elaborated in the previous section, Hrušovský et al. (2021) and Layeb et al. (2018) use a simulation model to capture the dynamic behavior and incorporate an optimization model to solve a service network design problem. Another example is proposed by Gallardo et al. (2021), a sequential optimization simulation framework in which the optimization module provides infrastructure arrangement and is evaluated by the simulation module. These studies present the extension of the simulation model by generating a solution from the optimization model instead of testing different scenarios to find the optimal solution

2.4. Learning Approaches in Freight Transport

In this thesis, the learning approach means employing machine learning (ML) techniques in solving freight transport-related problems. In general, the ML technique consists of three main categories: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Machine learning has been applied in various aspects of freight transportation. It ranges from predictive analytics such as demand forecasts to decision making tools such as operation planning (Filom et al., 2022). In a more advanced approach, a number of studies proposed a learning-based approach in the dynamic model.

2.4.1. Supervised and Unsupervised Learning in Freight Transport

A supervised learning is essentially a model to predict an output by learning from a labeled training (Barua et al., 2020). This labeled data basically tells the model which output is supposed to be, given the input data. The model is trained to extrapolate to predict a correct action for situations outside the training data (Sutton and Barto, 2018). Methods such as linear regression, logistic regression, or more sophisticated methods like artificial neural networks fall under the supervised learning category. Unsupervised learning on the other hand does not rely on labeled data, but the model is employed to find a hidden pattern within training data (Barua et al., 2020). The clustering method such as K-Means is a common method used for unsupervised learning. A raw dataset can be grouped into several clusters with similar characteristics to improve the result of supervised learning (Barua et al., 2020).

Geng et al. (2015) applies supervised learning, using a regression method to predict the container throughput of a port given socio-economic indicators. The accurate forecast of container throughput plays an important role in determining the direction of the port development, scale of investment, and berth location (Geng et al., 2015). Kourounioti et al. (2016) proposed another machine learning method to improve a decision making in container stacking policies. The research employs an artificial neural

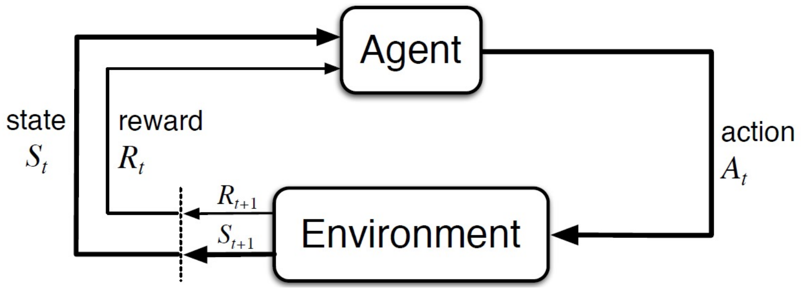


Figure 2.2: Agent-environment Interaction in Markov Decision Process (source: (Sutton and Barto, 2018))

network (ANN) to predict the container's dwell time in the stacking yard as well as identify its key determinants. Zhou et al. (2020) proposed an unsupervised learning model using K-Means algorithm to determine the location of the joint distribution center of restaurant industry in China. The proposed model is able to decrease the number of vehicles on the road as well as the delivery costs.

Reinforcement learning is another concept and different from the previous two concepts. A reinforcement learning model is developed to find a correct action under a certain state. Unlike supervised and unsupervised learning, in Reinforcement Learning, a learning agent is trained through a series of interactions with an environment instead of a given dataset (Sutton and Barto, 2018). The agent is given a reward for each action taken according to the agent's objective. By maximizing the reward, the agent can determine the best action for different states (Barua et al., 2020).

2.4.2. Reinforcement Learning in Synchromodal Framework

To make the reinforcement learning work, a Markov decision process (MDP) should be formulated to define the interaction between the learning agent and the environment (Sutton and Barto, 2018). Given the objective, the agent with a state will choose an action and interact with the environment. The environment then gives the agent the updated state and reward according to the taken action. This becomes a looping process until the agent reaches a terminal state which breaks the looping process. The illustration of that looping process can be seen in Figure 2.2.

The approach to solving the problem using reinforcement learning is divided into model-based and model-free. Model-based is when the agent has the means to predict the environment's response to any action taken which could be deterministic or stochastic, while in model-free approach the agent relies only on learning (Sutton and Barto, 2018). In the model-free approach, the agent is trained through episodic training and updates its value function. The value function represents how good the current state of an agent is relative to its goal. There are several methods to update this value function such as Monte-Carlo which uses experience from sample episodes, or Temporal Difference and Q-Learning by bootstrapping for the final outcome of expected rewards (Sutton and Barto, 2018). Using a combination of exploitation and exploring the lower immediate reward to find possible larger long-term returns, the agent generates an updated value function to be used as a reference in taking action for future exercise.

There are already a couple of studies proposing a model using a learning approach under synchromodal framework such as a study by Larsen et al. (2023a) which enhances the model in Larsen et al. (2023b) with a departure learning method. There are even fewer studies proposing a reinforcement learning technique under synchromodal framework. Guo et al. (2022) adopts the reinforcement learning approach in the global shipment matching problem under dynamic and stochastic travel time settings to address the curse of dimensionality of applying dynamic programming for solving the objective following Bellman's equation. The study formulates a transition function containing the information of the state over time to be used in each decision epoch. The states include the attributes of the shipment in the decision epoch t including the shipment's itinerary, position, set of accepted shipments, and the shipment's arrival time.

The reward system in this model uses the costs of moving a shipment from an origin terminal to its destination terminal via a certain service. The costs consist of travel costs, loading, unloading, storage, carbon tax, and delay penalty. The value function in the model is defined according to matching shipment $r \in R$ with service $s \in S$ creating a function $Q(r, s)$. This value function is updated over time using Q-Learning algorithm. This study compares the performance of RLA model with Myopic (MA) and Stochastic approach (SA). The findings show that RLA achieves higher total profits and reduced computation time across all instances when compared to MA and SA.

Another study using reinforcement learning technique under synchromodal framework is proposed by Zhang et al. (2023). This study builds on top of an Adaptive Large Neighborhood Search (ALNS) proposed in the study by Zhang et al. (2022) to address the service time uncertainty in synchromodal transport. Unlike the study by Guo et al. (2022) the learning agent in this study works side by side with the ALNS model instead of replacing its role. The learning agent receives the states from ALNS when an unexpected event occurs. The states consist of relevant information on shipments including the current time, passed terminals affected by unexpected events, travel time, and delay tolerance. The action space in the learning agent consists of request removal and insertion. Request removal means to remove the request from the assigned itinerary, while insertion means to put the shipment into a new itinerary after the removal.

The reward system used in the model by Zhang et al. (2023) is a 1 or 0 value. This reward is granted to the agent if the right action is taken with respect to the possible delay due to unexpected events. For instance, if removal action is taken, and the unexpected events cause a delay in the original plan, the learning agent receives the reward, otherwise if there is no delay. The ALNS model evaluates the actions taken by the learning agent and gives the reward accordingly. The learning agent updates the value function using a deep Q-Network making it capable of estimating a value of any state even though the learning agent has not yet visited that state. The result shows that the implementation of reinforcement learning in the decision making process improves performance by reducing the delays in the network.

2.5. Literature Review Summary

Synchromodal transport is an emerging concept to replace the previous intermodal concept with benefits providing added values to customers through more combination of multi-modal freight service (Tavasszy et al., 2015). Additionally, synchromodal transport with flexibility feature supported by real-time information on freight network, is capable of reacting to disruption and creating a more resilient freight network.

In addressing the real-time decision under synchromodal assumptions, dynamic models are proposed in various studies. Replanning model using MILP (Qu et al., 2019), shipment matching problem using rolling horizon approach (Guo et al., 2020, 2021), or simulation-based model (Di Febbraro et al., 2016; Ambra et al., 2019; Hrušovský et al., 2021) solve different problems in the network with different approaches with a same objective: To create a resilient synchromodal transport in response to disruptions. However, As the disruptions in network modelled as a stochastic problem, simulation-based would be suitable since it is proven to work well in solving stochastic problems (Layeb et al., 2018).

Furthermore, recent works by Guo et al. (2022) and Zhang et al. (2023) integrate the dynamic model with the RL technique and prove that using RL in synchromodal framework could increase its performance. Despite the same approach, both studies propose different ways of formulating the state, action, and reward system for the RL agent. This provides insights into formulating RL under synchromodal framework and could be a strong basis for integrating the RL technique with a simulation-based model to create a resilient synchromodal framework.

3 Modelling Disruptions in Multi-modal Freight Transportation

This chapter provides a more detailed elaboration of the types of disruptions in freight transportation and how to categorize them to be used in the simulation.

3.1. Disruptions in Multi-modal Freight Transportation

The disruption in the freight network varies in type and impact and may require different reaction strategies either at strategic, tactical, or operational levels. It can be distinguished according to the frequency and severity, categorized into endogenous and exogenous factors (Hrušovský et al., 2021), and come from different sources such as nature or human acts (Wang, 2016). Different types of disruptions in a freight network are mentioned in several studies as shown in the table 3.1. Another way to categorize the disruptions is by separating them into two spectrum as elaborated by Wide et al. (2022): low occurrence probability but severe impact, and high occurrence probability with low impact. The categorization of the disruptions could be useful to simplify a model while keeping the realistic behavior.

Table 3.1: Disruptions in Freight Transportation

Disruption	Source
Act of nature	Hrušovský et al. (2021), Wang (2016)
Demand changes	Hrušovský et al. (2021), Guo et al. (2020)
Travel Time Uncertainties	Guo et al. (2020)
Labor strike	Wang (2016)
Terrorist attack	Wang (2016)
Blockage, detours, and road works	Ambra et al. (2019)
Train/Railway maintenance	Ambra et al. (2019)
Railway accident	Ambra et al. (2019)
Suspension of dock operation	Pant et al. (2015)
Water level fluctuation	Jonkeren et al. (2011), Pant et al. (2015)
Customs Issue	Maersk (2023)
Port Congestion	Maersk (2023)

There are various ways to include disruptions in a model. Ambra et al. (2019) Create a simulation model under synchromodal framework and develop three profiles of disruptions capturing disruptions in train and truck network. Each profile has a certain severity represented by the duration of the event along with the probability of occurrence. Hrušovský et al. (2021) divided the disruptions into four scenarios with different durations and specific occurrence intervals and used them as input in a simulation model.

Table 3.2: Different Ways to Include Disruption in Models of Synchromodal Transport

Source	Model
Ambra et al. (2019)	Divided 3 disruption profile differ in duration and occurrence probability and simulated using Monte Carlo: **Profile 1**: Prob = 30-40%/year, Duration = uniform(1-3) hour(s) **Profile 2**: Prob = 6-9%/year, Duration = uniform(3-6) hour(s) **Profile 3**: Prob = 6-9%/year, Duration = uniform(1-3) day(s)
Di Febbraro et al. (2016)	Create a set of exogenous events including new orders, delays, and change of supply network that may stochastically occur
Hrušovský et al. (2021)	Location is chosen randomly, duration and frequency are based on literature. The frequency uses a fixed time interval 4 scenarios are provided: **Scenario 1**: Duration 2 hours, interval 2 hours **Scenario 2**: Duration 6 hours, interval 2 hours **Scenario 3**: Duration 12 hours, interval 8 hours **Scenario 4**: Duration 24 hours, interval 8 hours
Guo et al. (2021)	Spot shipment with probability distributions including: Type, origin, destination, volume, announce time, release time, due time, freight rate, and delay cost. Service uncertainty as a random variable including: Travel time, departure time, and arrival time
Qu et al. (2019)	- Shipment delay release time: 0.5h - 1.5h - Shipment volume change: -30% - +30% - Service delay 0.5h - 1.5h
Van Riessen et al. (2015)	- Late and early service departure: 6h, 12h, and 24h - Cancellation of services
Zhao and Dick (2024)	16 levels of railway disruption, modeled in different duration time ranging from 3 to 48h.
Durán-Micco et al. (2023)	Disruptions are modelled as delays and cancellations of train services following an exponential distribution.
Karam and Reinau (2022)	- Disruption on road network characterized with location, occurrence time, and duration. - Monte Carlo sampling procedure is used on historical data to generate the unexpected event. - Truck travel time uniformly distributed between 60-80km/hr
Gao and Liu (2022)	3 Specific disruptions: - Terrorist attack: random location and arcs' capacity = 0, - Earthquake: random location and reduce arcs' capacity, - Snowy weather: random location and reduce arcs' capacity.
Wide et al. (2022)	Delay scenario in train network ranging from 30-240 mins. Establised 5 scenarios of disruption with each scenario has a different level of information completeness.
Abadi and Ioannou (2014)	Disruption in port is modeled by 2% capacity reduction for every disruption scenario. The model is run under 15 scenarios from 2% to 30% capacity reduction.
Pant et al. (2015)	Uses mathematical equations to represent cargo loss due to disruption in a certain location

Guo et al. (2021) employs random variables including travel time, departure time and arrival time, request volume, and other request attributes to represent disruptions in both service network and spot shipment requests. A different approach was proposed by Gao and Liu (2022) by simulating specific high-impact disruption scenarios including terrorist attacks, earthquakes, and heavy snow represented by different capacity reductions in the affected links. The result of a thorough literature review of how to model the disruptions in freight networks can be seen in the table 3.2. Despite its various types, the different disruption impacts and severity could be generalized and represented into different categories of delay duration and capacity reduction. On the demand side, the disruption could be modeled by having a change in the request attributes. The categorization of disruptions is essential to reduce the complexity of the simulation model while still keeping the essential dynamic behavior.

3.2. Disruption Categorization

In this thesis, the disruption in the freight network is divided into two components encompassing disruptions on the service network and on the requests. The disruption profiles are created for each component with each profile representing a group of similar disruptions along with the possible impact and occurrence frequency.

On the supply side, the disruption categorization has more profiles than the request disruption. Five distinct profiles are developed to represent different types of service disruptions as detailed in Table 3.3. Each profile contains a group of disruptions with similar characteristics. The first profile is a frequent disruption that could cause a short delay on either the train or truck network. This could be caused by road congestion for trucks (Ambra et al., 2019), or technical and communication problems on trains (Palmqvist et al., 2022). The second profile is a delay in barge service lines which, for instance, is caused by congestion in the river due to locks or high traffic (Barkley and Mcleod, 2022). The third profile, adopted from (Ambra et al., 2019) is a possible delay due to more severe disruptions such as bad weather or systems maintenance. These disruptions could result in operations being halted for a certain period. The fourth profile is a disruption in the terminal such as operational problems, or port congestion (PoR, 2024). The fifth profile is a reduction on barge carrying capacity due to the fluctuation of river water level (van Dorsser et al., 2020)(CCNR, 2020). The low water level restricts barges from carrying containers with their full because the barges need to reduce the draft, while the high water level could limit the height of the stacked containers on the barge to prevent collisions with bridges

Table 3.3: Service Disruption Profile

Profile	Description	Mode/ Location	Effect in the Simulation	Duration	Capacity Reduction	Occurrence per Year
1	Operational delays, road congestions	Train, Truck	Delay	1-3h	0%	30%
2	Operational delays, canal congestion	Barge	Delay	1-6h	0%	35%
3	Bad weather, labor strike, accident, systems maintenance	Train, Barge	Delay	12-48h	0%	6%
4	Terminal congestion, operational delays	Terminal	Delay	1-3h	0%	30%
5	High and low water level	Barge	Carrying capacity reduction	12-24h	15-20%	10%

On the request side, three disruption profiles are considered, including two profiles of changes in container release time and a profile of alterations in shipment volume. In port-inland transportation, changes in the release time could happen due to several causes such as the late arrival of the mother vessels, which can cause delays of release time up to seven days (Statista, 2022) or more minor issues such as customs clearance which cause delays of less than a day. Meanwhile, the volume changes could come from the shippers due to, for instance, unexpected increases in demand beyond long-term contracts. The request disruption profiles are presented in Table 3.4.

Note that there is no duration associated with request disruptions; severity is instead defined by the delay in release time and changes in volume. Unlike delays in service disruptions, which cause operations to wait until the disruption ends, delays on the request side directly change the release time, allowing the system to 'know' the delay as soon as it occurs. This distinction is crucial because only service disruptions trigger the Reinforcement Learning agent to address unknown durations, a topic that will be discussed later in this report.

Table 3.4: Request Disruption Profile

Profile	Description	Location	Effect in the simulation	Delay Release	Valume Change	Occurrence per Year
6	Demand change	Shipment	Volume change	-	-30% to +30%	30%
7	Customs issues, main port operational delays	Shipment	Release time change	1-6h	-	30%
8	Mother vessels arrival delays	Shipment	Release time change	1-7d	-	5%

The disruption profiles are created based on two spectrums as explained by Wide et al. (2022) elaborated in Chapter 2. The high probability with a low severity level is represented by high occurrence per year and low value of severity (column 5 and 6) as attributed in Profile 1, Profile 2, Profile 4, Profile 6, and Profile 7. The other spectrum, the low probability with high severity disruption is attributed in Profile 3, Profile 5, and Profile 8. Notice that each profile can only occur in certain locations. The third column in the table indicates the possible location of disruption when it occurs. It could be either in a terminal, a service line, or directly on the shipment.

The severity and frequency in some profiles are derived from literature or other references, while the rest use assumptions due to insufficient available information. Profile 1 and 3 are adapted directly from Ambra et al. (2019) with additional information about barge disruption from TheLoadstar (2022). Profile 2 is deduced from average barge performance published by Port of Rotterdam (PoR, 2024). This source provides information about delayed and on-schedule barges in Port of Rotterdam in a year period. Profile 5 combines two information from van Dorsser et al. (2020) about the capacity reduction, and from CCNR (2020) for the occurrence probability. However, the information is for barge in general. For the container vessels, as the scope of this thesis, the information is strengthened by an analysis from another thesis work by Zhang (2024). The analysis shows low water level occurrence from 2011 to 2022, with an impact of 15-20% carrying capacity reduction during that period.

Profile 6 refers to scenarios developed in a study by Qu et al. (2019) where six different demand change scenarios are applied ranging from -30% to +30% of volume fluctuation. Finally, the severity of change in the shipment's release time is deduced from a database published by Statista (2022) as explained earlier. However, due to insufficient information, the severity for profile 7, and the occurrence probability of both profile 7 and 8 are assumed. The assumption is still based on the two spectrums (Wang, 2016) mentioned above.

4 Model Formulation

This section elaborates on the modeling part employed in this research. As explained in Chapter 1 there are three main steps encompassing: 1) Constructing an environment for a simulation, 2) Combining the simulation model with an optimization model to create a hybrid simulation-optimization model, 3) Integrating a learning approach to the hybrid simulation model. These steps are formulated under the synchromodal framework following several assumptions of the implementation of synchromodal transport.

4.1. Problem Description

The research examines hinterland freight transportation, specifically focusing on the unidirectional flow of shipments from the main port to various inland terminals. Each terminal is interconnected via dedicated service lines, which are exclusively served by one mode of transport—either barges, trains, or fleets of trucks. The study primarily considers container shipments originating from the main port and destined for different inland terminals. It does not address the final leg of transportation from these terminals to distribution centers or warehouses. The shipments may be transported directly or through multiple service lines, involving transfers at transshipment terminals, thus constituting a multi-modal transport network.

There are multiple actors involved in this hinterland multi-modal transportation including:

1. Logistic Service Providers (LSPs).
2. Flexible Transport Operators.
3. Fixed Schedule Transport Operators.
4. Centralized Planner.

In this thesis, LSPs represent shippers, functioning as customers for transport operators, who deliver cargo between terminals. These operators are categorized into flexible and fixed schedule services. Flexible services, typically trucks, can adjust their departure times and destinations according to specific requests. Meanwhile, fixed schedule services, such as trains and barges, operate on predefined routes and schedules. While capable of transporting larger volumes per vehicle resulting in lower costs due to the economic of scale, they generally take longer time to deliver compared to truck services.

In traditional intermodal transport, each operator works independently but may collaborate with LSPs and other operators to create one comprehensive logistics chain from origin to destination. Traditionally, no centralized planner coordinates the entire network or synchronizes different services across logistics chains. In contrast, synchromodal transport assumes the presence of a centralized planner or *orchestrator*(Giusti et al., 2019)) with comprehensive information from LSPs and operators, enabling them to synchronize services and achieve horizontal integration across various logistics chains.

Within this context, terminals and service lines are collectively referred to as the ”service network,” while shipments are termed ”requests” throughout the remainder of the report. Real-world operations often face disruptions in both the service network and requests, manifesting as delays, capacity reductions, or changes in shipment release times. Under a synchromodal framework, the service network adapts flexibly in real-time to these disruptions. This flexibility primarily involves reallocating containers or matching them with available services, rather than altering fixed service schedules, which is typically challenging in practice. The flexibility feature is illustrated in Figure 4.1.

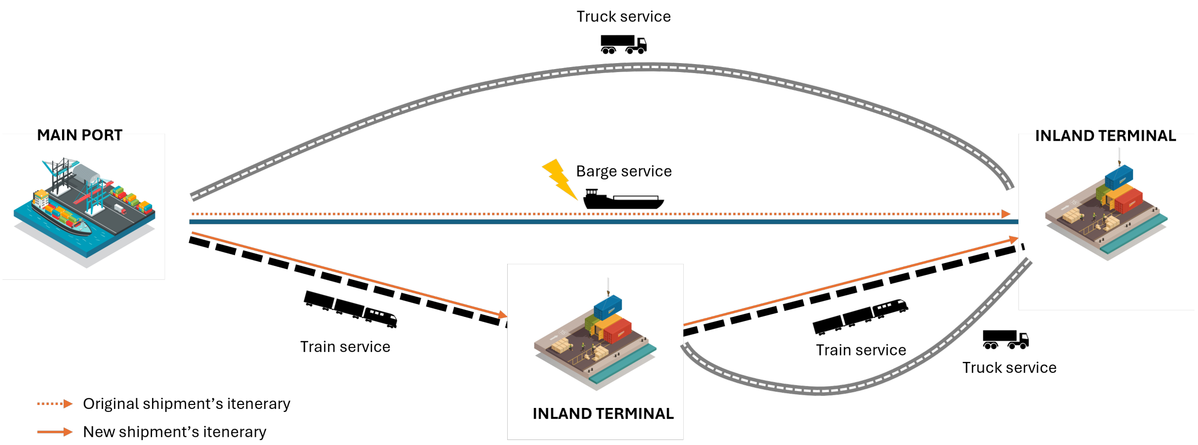


Figure 4.1: Synchromodal Flexibility

Figure 4.1 depicts a shipment originally scheduled to travel from a main port to an inland terminal via barge service. The figure illustrates a disruption in the barge service and, through the flexibility feature of the synchromodal system, shows that the shipment can be seamlessly shifted to train service, as indicated by the solid red line. This reallocation of containers does not require any changes to the current frequency or schedule of the service lines, as long as it satisfies the service time window.

To enable the synchromodal framework in the service network, the model follows several assumptions according to the definition outlined in Chapter 2. First, this research assumes full disclosure of enough required information among operators and LSPs, allowing the central planner to re-route a shipment flexibly. Moreover, the information is exchanged in real-time assuming the necessary ICT infrastructure is available. Finally, The modal free booking is applied to all shipments granting full authorization to the planning in reallocating the containers.

4.2. Simulation Module

The simulation module is designed to represent a real-world operation of hinterland multi-modal freight transport using a discrete-event simulation method. Before going into the explanation of how this module works, it is important to note that several key assumptions are held in this simulation to reduce the complexity without compromising the important aspects of the research scope.

1. There are an unlimited number of truck service lines, therefore, all the shipments assigned to truck service could always be carried.
2. Constant waiting time for truck arrival to pick up shipments. In real-world operation, this depends on the location of the depot.
3. Land side activities are neglected. The operation inside the terminal such as handling and moving a container from one yard to another yard is simplified into one activity
4. All containers in the same shipment are transported in the same service.
5. After completing a trip, a service is automatically back to the origin terminal, neglecting the return trip.
6. Service lines do not make multiple stops, meaning they only serve two nodes encompassing an origin and a destination terminal

4.2.1. Simulation Requirement and Key Performance Indicator (KPI)

This simulation module is developed to represent a real-world hinterland synchromodal freight transportation and evaluate the performance of the network. Later, it will be integrated with a optimization module and a learning agent to create a decision support tool under synchromodal framework. To achieve its ultimate objective, several requirements are set as listed below:

1. The simulation module must apply disruption in a hinterland multi-modal freight transport network.
2. The simulation module must calculate and record the transport cost of a shipment.
3. The simulation module must be capable of generating accurate necessary output anytime within the simulation time to be used as input in the optimization module
4. The simulation module must accommodate different input datasets and be capable of scaling up.

The main objective of this simulation is to capture the dynamic nature of disruptions in the hinterland freight network representing real-world operations to create an environment for implementing a decision support system. To measure the performance of the decision support system, the costs of transporting shipments from the origin to the destination terminal are chosen to be the KPI. The total costs consist of at least a storage cost, travel costs, a handling cost, and a delay penalty. These cost components are highly related to the time performance, thus, it is sufficient to set only the cost to be a single KPI.

4.2.2. Simulation Input

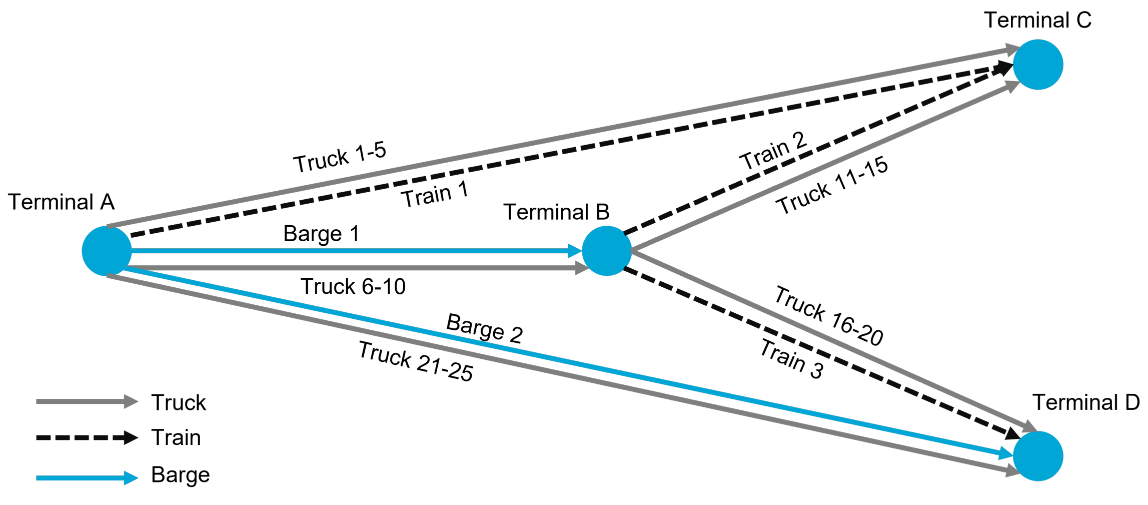


Figure 4.2: Synchromodal Network Representation

The inland terminals are represented by nodes located at various locations. Multiple nodes of inland terminals could be located within the main port and serve as origin points for loading shipments onto transport modes. Other nodes are scattered further in the hinterland as the transshipment terminals or destination points. These terminals are characterized by handling capacity affecting the loading or unloading time. In this simulation, several parameters are assumed infinite such as stacking yard capacity and vehicle buffer area. Violation of these parameters could result in terminal congestion and could cause a delay. Rather than creating parameters, the port congestion is modeled using random variables to represent unexpected events which will be discussed further in the later part.

The first input of the simulation module is the service network, containing the physical parameters of the network. Each pair of nodes are linked by arcs representing service lines of a specific mode type. The arc's physical parameter is a length indicating a physical distance from one terminal to another. Two arcs of different mode types connecting the same nodes might have different physical distances due to different physical infrastructures. For example, traveling from Terminal A to Terminal B might take only 70 km by barge while it can take 100 km by truck. The layout of the service network is illustrated in Figure 4.2.

The second input is the service lines containing the operational parameters including travel speed, carrying capacity, origin and destination, and departure time. These parameters play an essential role in determining the occurrence of events in the discrete event simulation. The service lines consist of three modes: barge, train, and truck. Barge and train services follow a fixed schedule, whereas truck services are more flexible, adapting the departure time to demands. There could be multiple lines of the same mode between two nodes. For instance, If there are three train services from Terminal A to Terminal B weekly, the input data will reflect three distinct sets of operational parameters. Truck services are modeled with a flexible capacity, assuming availability to match demand, such as deploying a fleet of 50 trucks for a request of 50 containers.

The third main input for the simulation module is the requests. These requests represent shipments from the main port to various inland terminal destinations. Each request consists of a bundle of containers bound by a single contract and shares the same origin and destination. Therefore, each request is characterized by an origin, a destination, and a container volume. Additionally, time parameters such as announcement time, release time, and due time are specified for each request. The due time reflects the customer's expectations, with any delay beyond this time incurring a penalty.

The fourth input is the cost structure divided into service network-related costs and shipment-related costs. Service network costs include storage, handling, and travel costs. Storage costs may vary between terminals due to different levels of service and terminal operators. The handling costs are unique to each type of mode as various transport modes require different handling equipment. Travel costs are split into two components: one dependent on travel time, and the other on distance traveled. The shipment-related cost is the delay penalty. The delay penalty is the cost for one container for every time unit delay. The delay penalty per container could be different for different shipments. In real practice, high-value items such as electronics incur a higher delay penalty compared to lower-value cargo like garment products. In this simulation, however, a delay penalty is considered the same for all shipments.

Finally, the last input is the disruption profile containing several categories of disruptions as discussed in Chapter 3

4.2.3. Simulation Description

To formulate the simulation module, sets and parameters are determined as presented in Table 4.1. The notation in the table is used in the equation for time and cost calculation in the simulation process. Sets and parameters in the simulation module are derived from the input while the variables are determined during the simulation process. The simulation module consists of three main components: mode service, shipment, and disruption. Each component has its process, intertwining with one another through an object-oriented model.

Mode Service Process

The fixed schedule mode service operates on a regular basis with predetermined departure times and routes, beginning at a specific origin terminal and concluding at a designated destination. In contrast, the truck services only commence the operation once a shipment is assigned, making the departure time contingent on the shipment's release time. Before departure, the process initiates with the mode of service arriving at the origin terminal. Since the return trip is neglected, the mode service is assumed to arrive and be available at the origin terminal Wt_k minutes before the scheduled departure time. This early arrival facilitates the start of the loading process, during which the mode service remains idle until loading is complete. The loading process depends on the number of containers assigned to the mode. If the loading duration extends beyond the planned departure time, this results in a late departure.

The mode of service then travels to the next destination according to its speed and distance. The simulation module calculates the travel time and determines the actual arrival time following Equation 4.4. Upon arrival at the destination terminal, the mode service waits until all the assigned requests are unloaded. As mentioned in the assumptions, the trip from the hinterland back to the origin point is neglected, therefore, in this simulation, the mode waits for the next departure time and automatically starts the process over from its origin point for the next service loop. This cycle repeats continuously until the end of the simulation period for that episode.

Table 4.1: Simulation Notation

Sets and indices		
N	Set of terminals	$i \in N$
K	Set of service lines	$k \in K$
R	Set of requests	$s \in S$
P	Set of disruption profiles	$p \in P$
Parameters		
δ_k	Travel distance for service line k	[km]
o_k	Origin of mode service k	
d_k	Destination of mode service k	
v_k	Travel speed of mode service k	[km/hour]
u_k	Carrying capacity of mode service k	[TEU]
Dt_k	Departure time of mode service k	
Wt_k	Time window for loading of mode service k	[minutes]
hs_k	Handling speed of mode service k per TEU	[minute/TEU]
C_k^n	Associated cost for mode k, $n \in [1,2,3]$, C_k^1 are the distance related travel cost [Euro/km/TEU], C_k^2 are the time related travel cost [Euro/hour/TEU], and C_k^3 are the handling cost [Euro/TEU]	
o_r	Origin of request r	
d_r	Destination of request r	
at_r	Announce time of request r	
a_r	Release time of request r	
b_r	Due time of request r	
n_r	Number of containers in one shipment for request r	[TEU]
C_r^d	Cost of delay for request r	[Euro/hour/TEU]
C^s	Storage unit cost	[Euro/hour/TEU]
Variables		
R_k	Set of assigned request(s) for mode k	$r \in R_k$
K_r	Set of assigned service mode(s) for request r	$k \in K_r$
Avt_k	Time a mode k available for loading	
Ht_k	Total handling time for loading or unloading	
Dt_k^{act}	Actual departure time of mode service k	
At_k	Arrival time of mode service k	
St_{rk}	Storage time before loading to mode k	
Avt_r	Time a request k available after unloading	
l_r	Delay time for each request r	
SC_r	Total storage cost of request r	[Euro]
HC_r	Total handling cost of request r	[Euro]
RC_r	Total travel cost of request r	[Euro]
LC_r	Delay cost of request r	[Euro]
TC_r	Total cost of request r from released to delivery	[Euro]

The whole mode service process explained above is translated into a pseudo-algorithm presented in Algorithm 1

Shipment Process

The shipment components, or requests, follow a distinct process within the simulation, interacting with the mode service component. Each shipment is generated in the simulation environment at its designated announcement time, which serves as preliminary information prior to the shipment's availability for pickup at the main port. Following the announcement, the shipment remains at the port until its release time, at which point it becomes available at the origin terminal. Once it is released, it waits until the assigned mode service is available for loading. The storage duration, from release to the start of

Algorithm 1 Service Mode Process

```
1:  Let $t$ be the current time
2:  while $t$ < simulation duration do
3:      Wait until arrival time $Avt_k$
4:      Update current location $p_k$ = origin terminal $o_k$
5:      Wait $Ht_k$ minutes until loading finishes
6:      if current location $p_k$ is disrupted then
7:          Wait until disruption ends
8:      end if
9:      Wait until actual departure time $Dt_k^{act}$
10:     Update current location $p_k$ = service line name $k$
11:     Travel to the next terminal for $\frac{\delta_k}{v_k}$
12:     if current location $p_k$ is disrupted then
13:         Wait until disruption ends
14:     end if
15:     Update current location $p_k$ = destination terminal $d_k$
16:     if current location $p_k$ is disrupted then
17:         Wait until disruption ends
18:     end if
19:     Arrive at destination and wait until unloading finishes for $Ht_k$
20:     Update scheduled departure time $Dt_k$ for next departure
```

loading, is recorded.

The assignment of a shipment to a mode service will be further discussed in the later section. During the loading process, the costs are calculated according to the shipment volume. For shipments assigned to a fixed schedule service, the departure time hinges on the completion of loading for all shipments allocated to the same mode service. Once loaded, the shipment remains with the mode service until it reaches the next terminal, where it is promptly unloaded and stored at the destination terminal. The unloading cost is computed similarly to the loading cost.

After unloading, the shipment updates its location based on the mode service's destination. If this location is not the final destination of the shipment, it updates its release time and restarts the process—from loading to transit—according to the designated itinerary. If the current location is the end destination and arrives beyond the due time, the delay penalty is calculated according to the late duration. The pseudo-algorithm of the shipment process is presented in Algorithm 2.

Although the service mode and shipment have their own process, there are some interactions between them. The formalization of these flow processes along with their interaction is illustrated in Figure 4.6.

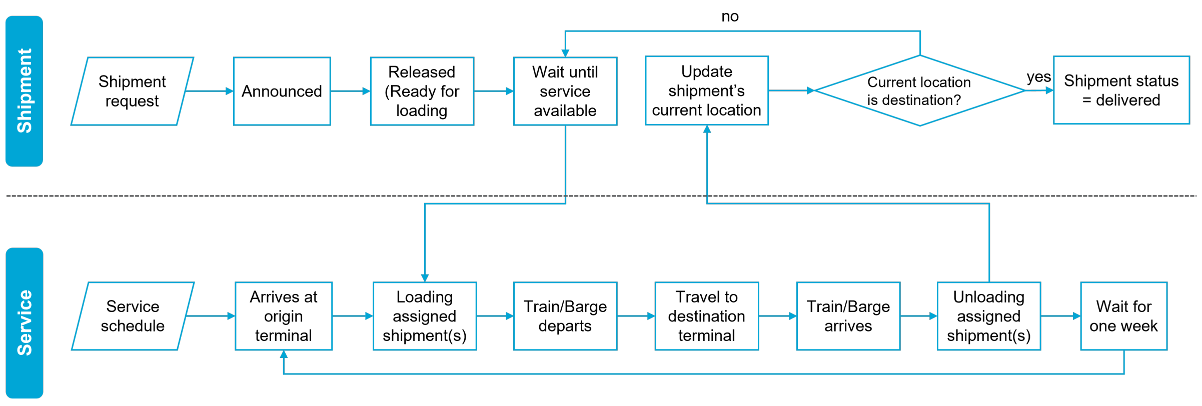

Shipment
Shipment request
Announced
Released (Ready for loading
Wait until service available
Update shipment's current location
Current location is destination?
no
yes
Shipment status = delivered
Service
Service schedule
Arrives at origin terminal
Loading assigned shipment(s)
Train/Barge departs
Travel to destination terminal
Train/Barge arrives
Unloading assigned shipment(s)
Wait for one week

Figure 4.3: Shipment and Service Simulation Flow Process

Algorithm 2 Shipment Process

```
1: Let $t$ be the current time
2: Let $k$ be the assigned service
3: Wait until announce time $at_r$
4: Update current location $p_r$ = request origin $o_r$
5: while $t$ < request release time $a_r$ do
6:     Wait until release time $a_r$
7:     if Request $r$ is disrupted then
8:         Update volume or release time
9:     end if
10: end while
11: while Current location is not request destination $d_r$ do
12:     Wait for assigned mode arrival
13:     Start being loaded
14:     Calculate storage time $ST_{rk}$
15:     Calculate handling cost $C_k^3 n_r$
16:     Update current location $p_r$ = service line name $k$
17:     Update service $k$ free capacity
18:     Wait until service $k$ arrives at destination $d_k$
19:     Calculate travel cost $C_k^1 \delta_k + C_k^2 \frac{\delta_k}{v_k}$
20:     Start being unloaded
21:     Update service $k$ free capacity
22:     Calculate handling cost $C_k^3 n_r$
23:     Update current location $p_r$ = service destination $d_k$
24:     Update $k$ to next mode in Itinerary
25: end while
26: if $t$ > request due time $b_r$ then
27:     Calculate delay penalty $LC_r$
28: end if
29: Calculate Total cost $TC_r$
30: =0
```

Simulation Variable Calculation

The chain of process in the mode service and request lies on the equations to calculate the time variables mentioned in Table 4.1. Furthermore, the time variables also determine the transport costs of each shipment. The equations to calculate the time-related variable of mode service are presented in Equation 4.1 to 4.4, while Equation 4.6 to Equation 4.8 are used to determine time-related variable of requests.

$$Avt_k = Dt_k - Wt_k \qquad k \in K \tag{4.1}$$

$$Ht_k = hs_k \sum_{r \in R_k} n_r \qquad k \in K \tag{4.2}$$

$$Dt_k^{act} = max(Dt_k, (Avt_k + Ht_k)) \qquad k \in K \tag{4.3}$$

$$At_k = Dt_k^{act} + \frac{\delta_k}{v_k} \qquad k \in K \tag{4.4}$$

$$Avt_r = At_k + Ht_k \qquad k \in K_r, r \in R \tag{4.5}$$

$$St_{rk} = Avt_k - a_r \qquad k \in K_r, r \in R \tag{4.6}$$

$$a_r = Avt_r \qquad r \in R \tag{4.7}$$

$$l_r = max(0, Avt_r - b_r) \qquad k \in K_r, r \in R \tag{4.8}$$

Equation 4.1 to Equation 4.4 regulate the calculation of mode available time Avt_k, handling time Ht_k, actual departure time Dt_k^{act}, and arrival time respectively At_k. Equation 4.6 calculates the storage time

St_{rk} depending on the request's release time a_r. The release time is updated after a request arrives in the next terminal following Equation 4.7. This update affects the storage time in the next leg. Lastly, the delay time is calculated following Equation 4.8.

Based on the determined time, the shipment cost can be calculated following Equation 4.9 to Equation 4.13. The equations are mostly straightforward just by multiplying time by unit costs. Equation 4.11 also adds a distance variable to calculate the travel cost. Equation 4.10 consists of loading and unloading for each mode service, therefore the calculation is multiplied by 2.

$$SC_r = \sum_{k \in K_r} C^s St_{rk} \qquad r \in R \tag{4.9}$$

$$HC_r = \sum_{k \in K_r} 2C^3_k n_r \qquad r \in R \tag{4.10}$$

$$RC_r = \sum_{k \in K_r} C^1_k \delta_k + \sum_{k \in K_r} C^2_k \frac{\delta_k}{v_k} \qquad r \in R \tag{4.11}$$

$$LC_r = C^d_r l_r \qquad r \in R \tag{4.12}$$

$$TC_r = SC_r + HC_r + RC_r + LC_r \qquad r \in R \tag{4.13}$$

The above calculations apply under an undisrupted scenario. Under the disrupted network, the time calculation might be affected due to unavailable or delayed service. For example, in Equation 4.3, the actual departure time Dt^{act}_k is only affected by the loading time Ht_k, while in the disrupted scenario, the actual departure time might be later because the service waits until the disruption ends.

Figure 4.4 is presented to give an illustration of how the shipment cost is calculated throughout the simulation process. The figure shows a calculation for a single trip. For an itinerary with a transshipment in between, the calculation is repeated except for the delay penalty. The delay cost is only calculated after the request is available at the request's end destination.

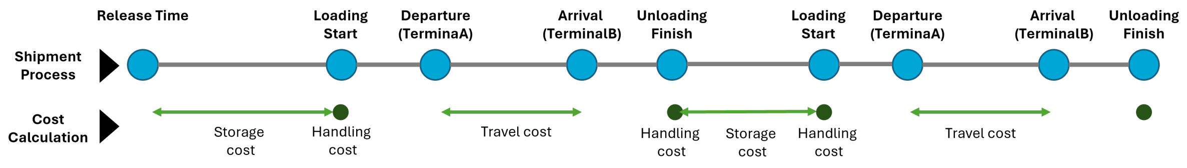


Figure 4.4: Shipment Cost Calculation (For one trip)

Disruption Process

The disruptions are applied according to the disruption profile, and each is generated independently within its designated profile. The simulation applies eight disruption profiles, five for disruptions in the service network and three others for request disruptions. The process of the generator initiates by randomly determining the time of occurrence according to each profile's occurrence probability following an exponential distribution. This is a common approach to model inter-arrival time in a simulation. The occurrence per year is converted into an occurrence rate (λ) by relating the annual frequency (f) of the disruption to the number of minutes in a year (t). as shown in Equation 4.14. For example, the occurrence probability of the service disruption Profile 1 is 30% each year, meaning it happens around 110 times every year. Since the time unit is in minutes, the frequency is divided by 525,600 minutes, resulting in the value of (λ) around 0.0002.

$$\lambda = \frac{f}{t} \tag{4.14}$$

Upon determining the time of occurrence, the simulation then identifies the specific location of the disruption. For disruption on requests, the generator selects a shipment that has already been announced but has not yet been released. For the service disruption the profile with more than one possible location, for instance, the service disruption profile 3, the algorithm applies the random variable in two steps. First, it determines in which mode type or terminal the disruption will occur, then it chooses a specific location accordingly. For instance, if the mode "Train" is selected, the subsequent step pinpoints which train line will experience the disruption. Both steps utilize a uniform distribution, assuming an equal likelihood of disruption across all locations

Next, the generator determines the severity of the disruption. For the disruption on requests, a multiplier value is generated following a uniform distribution. This multiplier value alters either the release time or volume of the request. For the service disruption, the severity is represented by the duration or capacity reduction depending on the type. It is determined according to the lower bound and upper bound in each disruption profile. The duration is chosen randomly following a uniform distribution adopting the approach in the study by Ambra et al. (2019).

Finally, the simulation module logs and implements each disruption based on its occurrence time. The impact of disruptions on requests typically results in straightforward adjustments to shipment parameters. On service lines, disruptions that create delays temporarily halt the operational process until the disruption concludes, whereas capacity reduction only decreases the parameters without stopping the operation. The entire flow of the disruption generator, including these dynamics, is illustrated in Figure 4.5.

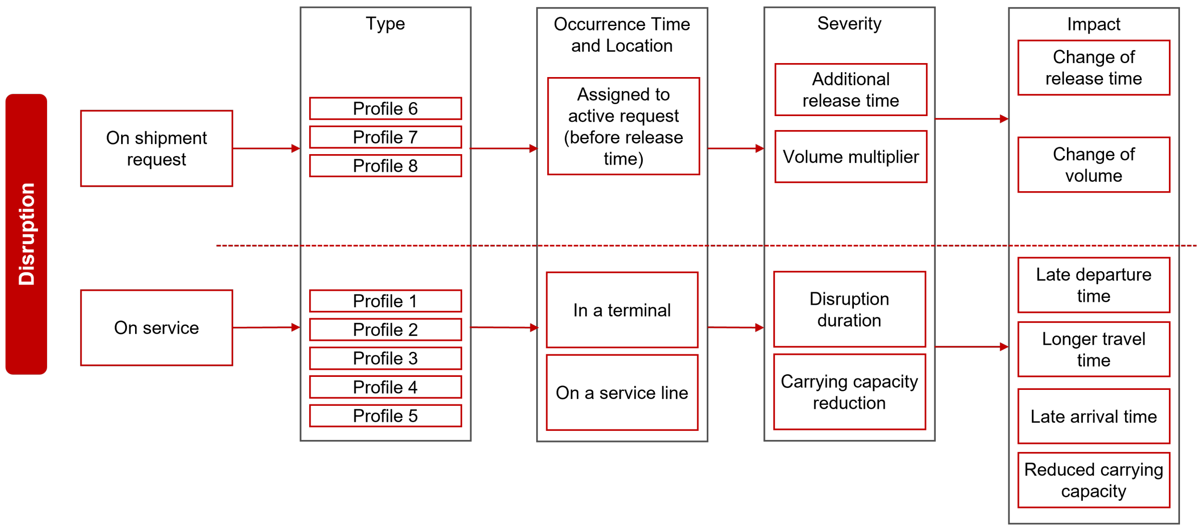


Figure 4.5: Disruption Generator Flow Process

The complete flow of the simulation encompassing the shipment and mode service process running under a disrupted network is presented in Figure 4.6. As explained before, the disruption on requests only applies in the small time window after a shipment is announced and before it is released. However, the disruption on the service line could occur anytime in the chain of events causing the disrupted request to wait until the disruption ends as illustrated in Figure 4.7. This *always wait* policy represents the absence of synchromodal framework and potentially causes severe delays in shipment delivery which will be addressed in the next section. These disruption occurrences are represented by the red arrows connecting the disruption generators (red boxes) and the shipment and service mode process in the picture.

4.3. Hybrid Simulation-Optimization

In the previous section, a variable K_r is presented in Table 4.1. This variable indicates the assigned itinerary to a shipment consisting of a set of service lines. This mode assignment to each shipment depends on various parameters and is highly correlated to the performance of the freight network. An optimal mode assignment could result in low costs of transporting all the requests from the origins to their destinations. In conventional intermodal transport, this itinerary comes from LSPs. However,

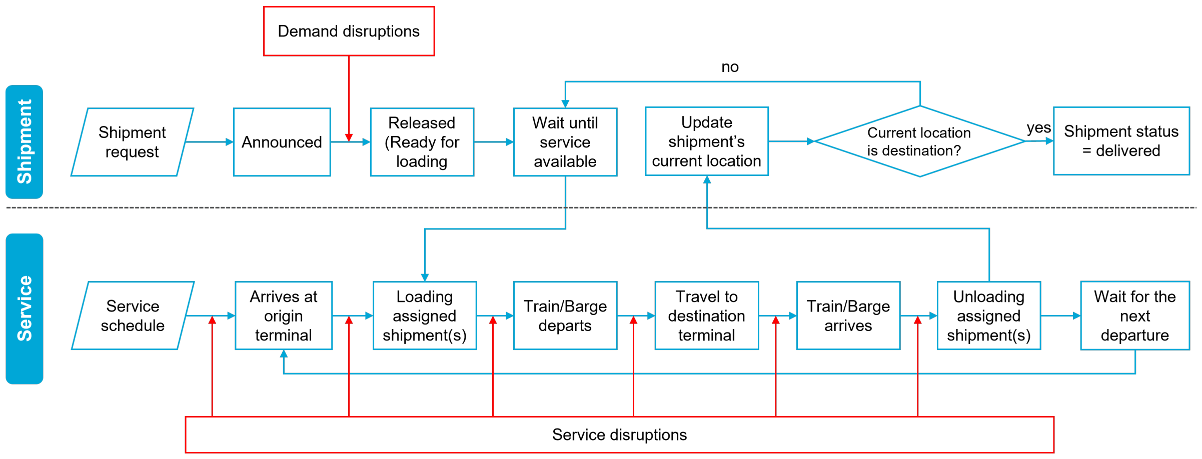

Demand disruptions
no
Shipment
Shipment request
Announced
Released (Ready for loading
Wait until service available
Update shipment's current location
Current location is destination?
yes
Shipment status = delivered
Service
Service schedule
Arrives at origin terminal
Loading assigned shipment(s)
Train/Barge departs
Travel to destination terminal
Train/Barge arrives
Unloading assigned shipment(s)
Wait for the next departure
Service disruptions

Figure 4.6: Complete Simulation Flow Process

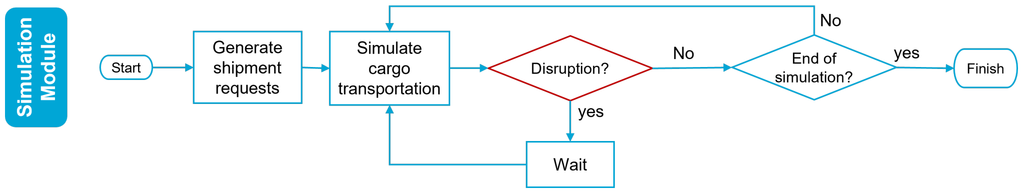


Figure 4.7: Simulation Module Flow Process

under the synchromodal framework, it is assumed that there is a centralized planner who has access to real-time shipment and service data as well as authorization to assign shipments to service modes, thus shipment requests are delivered according to their requirements. The objective could vary such as minimizing cost, maximizing on-time delivery, or minimizing emissions. Therefore, in this thesis, the centralized planner has a role in creating a shipment plan every certain time interval.

Planning and simulating the transport could normally be performed separately. For every time interval, all necessary data from requests are given to a planner to create a shipment plan. Then, the shipment plan is executed as an input in the simulation. This planning process is called an offline planning. However, if there is a disruption in the network, the shipment can only wait until the disruption ends. Through the synchromodal framework, a planning module could be integrated into the simulation and automatically triggered to create a new shipment plan in case of disruption.

There are various analytical methods to perform shipment planning, ranging from heuristic methods following first come first serve (FIFO) principal, or sophisticated optimization techniques with various objectives as mentioned earlier. The hybrid simulation-optimization model in this thesis is proposed to enable integration with different optimization models as long as it satisfies the system's requirement. The optimization model should match the announced requests with available services considering the time and cost parameters associated with the service lines. The optimization module is embedded in the simulation module and acts as the centralized planner. It is capable of performing a replanning immediately after a disruption occurs. The real-time data from the simulation is sent directly as an input of the replanning.

4.3.1. Affected Request Detection

To perform the replanning or online planning, it is important to filter the input to the affected ones. This logic is adapted from model proposed by Hrušovský et al. (2021) and Zhang et al. (2023). In both studies, an affected request algorithm is employed to determine the affected request based on the disruption occurrence time and location. This algorithm is employed to reduce the input size and number of changes in the network due to the replanning (Hrušovský et al., 2021; Zhang et al., 2022). Therefore, an affected request detection algorithm is implemented in the simulation.

The detection algorithm has access to all the requests' real-time information within the network, thus, it knows the current location of all requests. It is important to distinguish between the affected requests and disrupted requests. The affected requests are only the ones that are still possible to be re-routed. There are three specific cases where a shipment is excluded from the affected requests despite the disruption in its itinerary. First, if the shipment is in a terminal or on board of a service line and the disruption occurs in that location (case 1). Second, if the shipment is on a service line, and a disruption occurs in that service line's destination terminal (case 2). Third, if the disruption occurs at the shipment's end destinations (case 3), the replanning in these three cases will not help the shipment to avoid disruption. The illustration of them is presented in Figure 4.8.

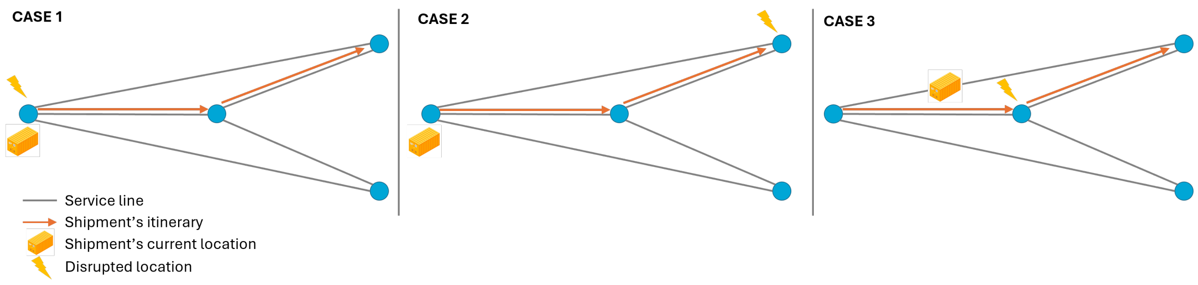


Figure 4.8: Exclusion of Affected Request

This detection algorithm as presented in Algorithm 3 is only triggered if a disruption in the service network occurs since disruptions on request are explicitly located in a specific shipment. First, the algorithm only considers the requests that are already assigned to service lines. It populates all the assigned service modes including that mode's destination, resulting in a complete route for each shipment. This route is used to identify whether or not a shipment is affected by the disruption. The detection excludes the shipment with cases as described in Figure 4.8. After all affected requests are located, the detection algorithm sends the affected shipment list to the central planner to assign new service modes.

Algorithm 3 Affected Shipment Detection

```
1:  Input: Service disruption location $p_d$, shipment data: assigned itinerary $K_r$, current location $p_r$,
    destination $d_r$
2:  Output: Set of affected request $AR$
3:  Let $R^{act}$ be the set of active requests in the network
4:  Initiate an empty list $AR$ for affected request list
5:  for each $r \in R^{act}$ do
6:      Initiate empty list $L_r$ for request route
7:      if Mode services are assigned to request $r$ then
8:          for mode $k \in K_r$ do
9:              Add $k$ and its destination $d_k$ to list $L_r$
10:             if Current location $p_r$ in $L_r$, remove from the list then
11:                 Remove end destination $d_r$ from $L_r$
12:                 if service disruption location $p_d$ is in $L_r$ then, add r to $AR$
```

4.3.2. Optimization Module

This optimization module is essentially the centralized planner of the network. The procedure of planning and replanning are the same and only differ in the input and trigger timing. For the planning process, the input will be all the announced requests during the decision epoch time and triggered in a fixed time interval while replanning input is the affected request list and triggered only during a disruption occurrence. Aside from the requests list, a list of on-going disrupted locations is also an input to the optimization module both for planning and replanning process. It removes the disrupted service line from the possible solution space if the disruption profile causes delays in the service network.

The removal of a disrupted service network is based on the logic in the simulation module that a disrupted service line stops working until the disruption ends. Since the optimization module has no information about the disruption duration, it is assumed that the disrupted service lines are unavailable when the optimization module is triggered.

The scope of this thesis is to propose a model with a modular structure, allowing the plug-ins of different optimization techniques. The optimization module could have different methods, or different objectives as long as the output is a matching plan between shipment requests and service modes. The optimization module is considered as a *black box* with expected output given the input from the simulation module. Therefore, any available optimization methods with the same input and output as explained earlier could work in this model

After receiving the request list from either the announced request list or the affected shipment detection, the optimization module is triggered to perform a planning or replanning process. The optimization module follows some constraints in making the matching decision. The matching decision is then used to assign a service line to the shipment. In the replanning process, the new assignment is used to replace the original itinerary. In contrast with the *always wait* policy presented in Figure 4.7, this hybrid simulation-optimization model *always reassigns* affected shipments to new service lines as presented in Figure 4.9 because the disrupted location is removed from the possible solution space.

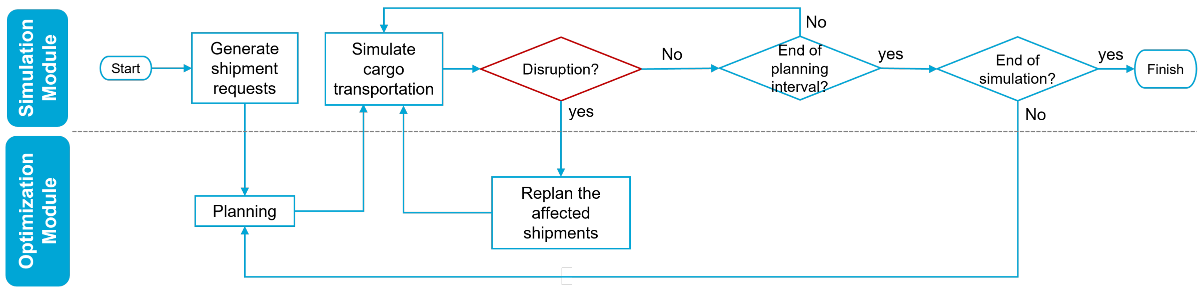

Simulation Module
Optimization Module
Start
Generate shipment requests
Simulate cargo transportation
Disruption?
No
yes
End of planning interval?
No
yes
End of simulation?
yes
No
Finish
Planning
Replan the affected shipments

Figure 4.9: Hybrid Simulation-Optimization Flow

4.4. Reinforcement Learning Approach

The combination of the optimization module with the simulation module to execute planning as well as replanning under disruption scenarios is expected to offer a more resilient performance of the network. The hybrid simulation-optimization changes the policy from *always wait* to *always reassign*. Reassigning a request to another service line under a disruption case could potentially increase the resilience of the freight network and is possible to execute under synchromodal framework. However, in some cases, it is better for a shipment request to just wait until the disruption ends and continue its journey according to its original itinerary. For instance, if a disruption occurs for a short time or it occurs in a terminal that is still far ahead in the journey and likely to end once the shipment arrives. The problem is that the duration of a disruption is usually unknown. Based on this, a reinforcement learning technique is integrated into this model, therefore, the model could decide if it is better to wait or reassign considering the experience of the learning agent.

Using a value function, a reinforcement learning (RL) agent is capable of choosing the best action given a specific state by learning from past experience and a series of training. Instead of relying on labeled data like supervised learning, the RL agent is guided by a reward system to indicate which action is the best. It is important to model the action, state, and reward system properly to have an optimal RL agent. Along with that, a Markov Decision Process (MDP) should be formulated as a basis for RL in having interaction with the environment by applying actions, updating the state, and getting rewards.

4.4.1. Markov Decision Process

MDP represents a sequential decision process in which the action taken in a state affects the next state and the reward generation. In this hybrid simulation-optimization model of synchromodal transport, the decision process could vary. Let us take a perspective from a shipment since it is assumed that the service network is fixed. In the case of disruption, the possible decision could be: ”when to take a certain service line”, ”which service line to take”, or ”should an affected shipment just wait until the disruption ends”. From those possible decisions, there is a large size of decision space. For instance, to answer the question ”when to take a certain service line”, a decision to take the same service line in different time steps are two different decisions. Furthermore, there are many service lines in the network, making the decision space even larger. Multiplying both the number of time steps and the number of service lines results in thousands of possible decisions. These possible decisions later will be the action space to be considered in the model. In addition to the action space, the state space could also be very large. Each shipment and service line has various attributes and keep changing over time. For example, the location of a shipment and a service line might be different at time step t and at time $t+1$. It is important to formulate the state properly so that the learning agent only considers the relevant states.

The action and state space in this synchromodal transport could be large as explained above. However, it is possible to narrow down that space by eliminating the irrelevant possible actions and states. Instead of considering all the possible actions, the RL agent can be set to work as a supplementary for the optimization module that has a role in providing the reassignment solution. In the previous section, the hybrid simulation-optimization works by assigning a shipment to a new service line in case of disruption. It sends back the output from the optimization module to the simulation module directly to execute. Here the process could be slightly modified. The optimization module sends the reassignment solution to the RL, then the RL agent decides if reassigning to a new service or waiting for the disruption to end would be the best action to take. By doing this, the RL does not have to consider the irrelevant actions because they are already eliminated in the optimization module. This narrows down the action space into two: Reassign and Wait. The timing of taking action is also limited to only after the disruption occurrence because RL agent is only triggered once the optimization module provides a reassignment solution.

In this approach, the environment which the RL agent has an interaction with, is the simulation module. Every time the RL agent is called to take an action due to a disruption occurrence, the simulation module will process the action and provide feedback consisting of an updated state and reward. The Markov Decision Process of interaction between RL agent and the simulation module is presented in Figure 4.10

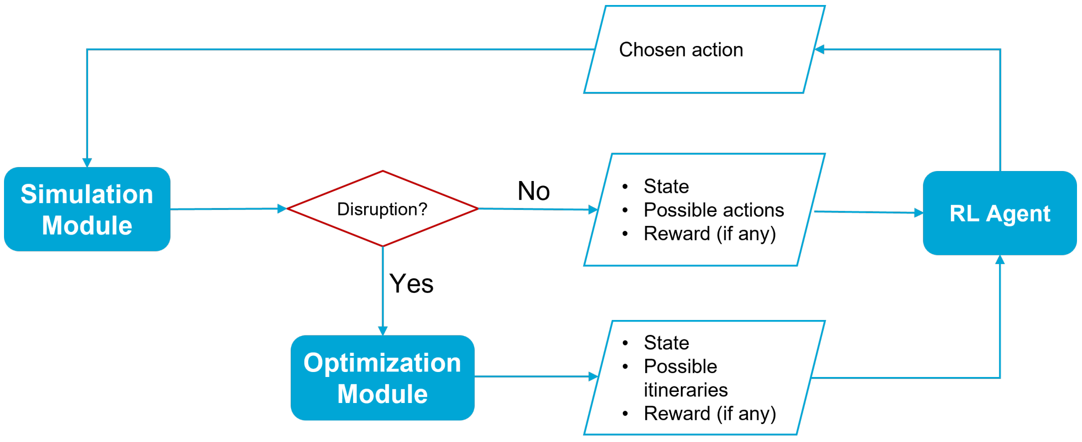


Figure 4.10: Markov Decision Process in Hybrid Simulation-Optimization

Notice that there is a branching flow from the simulation module in sending state information to the RL agent. This is because there is a probability of a case where RL agent needs to take a follow-up action before reaching the terminal state. The terminal state here is the state where the agent is no longer taking action and will be elaborated more later in this section. Before going to that explanation, it is necessary to define the action to be clearer.

4.4.2. RL Action

Now, what do the ”reassign” and ”wait” actions exactly mean? Since it is the information that will be exchanged between RL agent and the simulation module, it needs to be clear so the feedback information would also be relevant. Those actions consist of a list of service lines, indicating which combination of service lines a request is going to be carried by. Assume a Request X is assigned to Barge1 and then transferred to Train1 before reaching its end destination, then, the assigned mode K_r is [Barge1, Train1]. If there is a disruption, and the optimization module suggests the reassignment to Truck2, then the possible action set will stay with the original itinerary, *wait*: [Barge1, Train1], and *reassign*: [Truck2]. The RL then chooses between those two options. However, to make it more granular, the itinerary is considered as a series of actions, and the action itself only contains a single service line. For instance, if in that case, the RL chooses to wait, then, the action is [Barge1], and save the [Train1] for the next action in the future. Therefore, after taking Barge1 and arriving at Barge1’s destination, the simulation module will update the state to the RL agent, grant the reward, and take the next action Train1. There is no decision-making anymore at this stage since the possible action is only one, unless a disruption occurs during this process.

If there is another disruption in the network and it affects the same request as discussed above, then the optimization module is triggered again, suggesting another possible action. Here, the action from the previous decision epoch is interrupted without waiting to arrive at the next terminal. The state is immediately updated, the reward for the previous action is granted, and a new action is taken by the RL agent according to the suggested possible set of actions from the optimization module. In the Request X example where RL takes the action set Barge1 and Train1, if the disruption happens before the request arrives at Barge1’s destination, the future action (Train1) will be replaced by the new set of action taken by RL agent in this decision epoch. This process continues following the feedback loop presented by Figure 4.10 until that request is delivered, which is the terminal state.

Another problem is if there are multiple requests affected by a disruption, and this happens most of the time if a service disruption occurs. In this case, the decision made by the RL agent for each request is independent of other requests. In a way, there are multiple RL agents assigned for different requests and work in parallel, and it is called RL sub-agent. Once an RL sub-agent is assigned to a request, the loop starts until it reaches the terminal state for that request. If any disruption happens again in the future affecting the same request, that request is not assigned to a new RL sub-agent. This framework allows the RL sub-agent to make an action based on the action taken in the previous decision, such that an action within the same request is not independent of its previous action. From the explanation above, it seems like the RL agent works on a decentralized architecture. However, this is not the case in this framework. Even though the RL sub-agent makes decisions independently for each request, it has one centralized value function, updated whenever a reward is granted. This process is illustrated in Figure 4.11.

The horizontal arrows indicate the parallel process of each sub-agent assigned to a request. The vertical arrow from the decision making loop to the action value function is the information sent to the RL agent to update the action value function, which will be discussed in more detail in the later section. The arrow from the action value function back to RL-sub agent is information about the action taken specifically for each request. The action is chosen following the ϵ-greedy policy.

Since the action is chosen by referring to action value function, the greedy policy means the action with the highest value is always chosen. However, to balance the exploration and exploitation (Sutton and Barto, 2018), the action is chosen following the ϵ-greedy policy. Instead of always taking the action with the highest action value function, the ϵ value provides a small amount of probability so the RL agent sometimes chooses an action with a lower value and explores the probability of having a better reward in a longer run. To determine the action probability $\pi(a \mid s)$ according to the ϵ value, Equation 4.15 is applied. m is the number of possible actions and a^* is the best action according to action value function $Q(s, a)$.

The details of selecting action following the ϵ-greedy policy are presented in Algorithm 4. This process is performed independently for each affected request $r \in Ar$ and returns the chosen itinerary I_c based on the chosen action a.

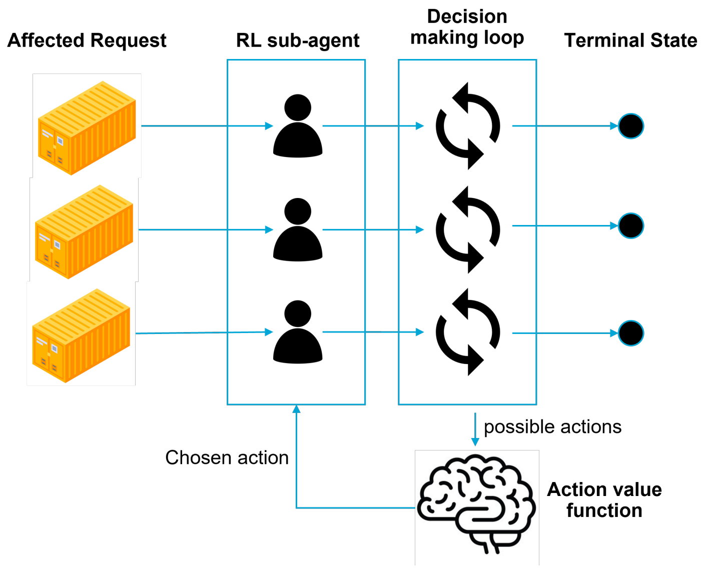


Figure 4.11: RL Agent Framework

$$\pi(a \mid s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon & \text{if } a^* = \arg\max_{a \in \mathcal{A}} Q(s,a) \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases} \tag{4.15}$$

Algorithm 4 Action Generator

```
1: Input: Affected request $r$, state of the request $s_r$, set of possible itinerary $I_r$, $\epsilon$ for $\epsilon$-greedy policy,
   action value function $Q(s,a)$
2: Output: Set of chosen itinerary $I_c$ (Wait or Reassign)
3: Wait action $a_{wait}$ is the first element of the list in wait itinerary $i_w \in I_r$
4: Reassign action $a_{reassign}$ is the first element of the list in reassign itinerary $i_r \in I_r$
5: Possible action $A = \{a_{wait}, a_{reassign}\}$
6: Determine action probability $\pi(a \mid s)$ using equation 4.15
7: Action $a$ is chosen randomly according to action probability
8: Chosen itinerary $I_c$ is determined according to the chosen action =0
```

4.4.3. RL State

From the perspective of a centralized planner of synchromodal transport, it is rather complicated to define the state of the RL agent. If all the shipments and all the service attributes are considered, the state space is going to be very large as explained earlier in this section. Since it is already elaborated that the perspective is narrowed down to a single shipment, it is easier the determine the state knowing that each request has its independent decision-making process. The state considered in this RL approach consists of 6 features: the request's current position, the request's destination, the request's due time, the request's volume, the type of disruption, and the current time.

The first 4 features are the attributes related to each request. The first one could change over time and could be either in a terminal or on a service line if it is on board. The rest are the same as given in the input of the simulation. The disruption profile is according to the type of disruption that happens when an action is being taken by RL agent. It is important to note that the RL agent only provided information about a type of disruption rather than the duration or probability distribution used in the disruption generator. It is still practical in real-world operations since the type of disruption is usually known. The last one is the time feature. it provides a sense of time to the RL agent so it can distinguish the same decisions in different time steps.

The state updating occurs every decision epoch and every time a request finishes an action. Using the previous example of Request X where the RL takes a set of actions of Barge1 and transfers to Train1, the RL agent receives a new state once the Barge1 arrives at its destination. The current location and current time will be updated, and the disruption profile will be updated to ”no disruption”. This is because the disruption feature indicates the type of disruption when an action is being taken. In this case, disruption happens when action Barge1 is taken, while Train1 is only a follow-up action without being triggered by a disruption. Anytime a disruption happens, the RL agent receives a new state from the environment. If a second disruption affects a request while the request is still waiting for the assigned mode to arrive, the RL agent still receives a new state for the new current time and updated disruption type while keeping the current location as is.

4.4.4. Reward System

The reward system plays a crucial role in determining the performance of an RL agent. Reward can be given in various ways. Take a common example provided in Sutton and Barto (2018) of a chess game, every action is rewarded 0 until the end of the game depending if a player wins, loses, or ties. The reward can also be given incrementally during the process as long as each action indeed has a value toward the end objective. Another example is a grid world where an object is rewarded -1 every time it moves one grid, and gets a huge reward when it arrives at the destination. The -1 acts like a cost to move for one grid. This way, the object will find the shortest path to achieve the destination while minimizing the cost. The incremental reward and negative reward to apply a cost to the RL agent can be adapted to synchromodal framework case.

The cost of transporting a request from a certain terminal to the next terminal is used in this model as a reward for taking an action. The cost consists of storage, travel, handling, and also delay costs. If the RL agent takes a longer route, it will be penalized by a higher travel cost, if the RL agent’s action results in the request waiting longer for departure, it will be penalized by a higher storage cost, and so on. However, taking action which has a higher cost in one aspect, can have a lower cost in other aspects. This is a trade-off that must be solved by RL in providing optimal actions. The reward is set to a negative cost, therefore, while the agent chooses an action based on the value function, it will choose the action with the negative values closest to 0, thus minimizing the costs.

Following the framework in Figure 4.11, the reward system is also given separately for each sub-agent. There are different cases of how the reward is given to the RL agent. First, in the case an action is completed without being interrupted by any disruption, a reward is given in the amount of transporting from one terminal to the next terminal starting from the time the RL takes the associated action consisting of storage, handling, travel costs, and delay cost (if any). Second, if a request is affected by a disruption while it is on board a service line, the RL selects action immediately but only executes once the request arrives at the next terminal. Therefore, the reward is only calculated after the request is unloaded at the next terminal. Third, If a request is affected by disruption multiple times, therefore the RL is triggered more than once for the same request. In this case, the reward for the last action from the previous decision epoch is given immediately after the RL agent takes the new actions. These scenarios are illustrated in Figure 4.12

The storage costs in the reward are also affected by the time of action is taken. For example, if a request starts at terminal A and is released at 8:00, the storage cost is calculated from that point. However, for the RL agent, the storage cost for rewards starts to be calculated when an action is taken by RL. So, if this request in the example is affected by a disruption at 10:00 while it is still waiting for the assigned mode, the storage cost for the reward is only calculated from 10:00 until the request is loaded to the assigned mode. Because of this, the actual storage cost is likely to be different from the storage cost in the reward. Furthermore, because the RL agent only works for affected requests, the total reward is not going to be the same as the total actual cost in the simulation. The reward generation process is presented by Algorithm 5.

4.4.5. Updating Action Value Function

Estimating a value function is the holy grail of the RL algorithm. It is a function that tells the RL agent how good it is being in a certain state (Sutton and Barto, 2018). In RL, a value function is strongly

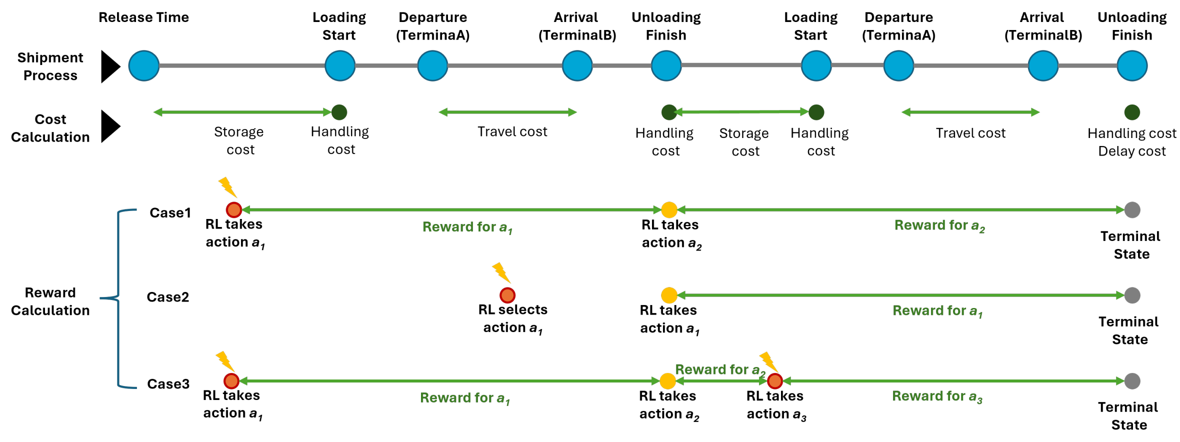


Figure 4.12: Example Cases of Reward Calculation

related to a policy. Sutton and Barto (2018) defines a policy as a map containing a probability of taking an action a given a state r. To put in the context of this thesis, *always wait* and *always reassign* are examples of a policy. It means that whatever the given state r, the probability of taking an action ”wait” is 100% for the first policy, and taking action ”reassign” is 100% for the second policy. By applying this policy through a series of training episodes, the RL agent can estimate the value function and determine how good it is being in a state r. The value function of a state (s) depends on the immediate reward and discounted expected reward for the future possible state.

The action value function is similar to the value function. The slight difference is, instead of only telling the agent how good it is being in a given state r, more specifically it tells the agent how good it is taking an action a in a given state r. To estimate the action value function of a specific policy π, the Equation 4.16 is applied.

$$q_\pi(s,a) = \mathbb{E}[G_t \mid S_t = s, A_t = a] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \tag{4.16}$$

G_t is the total reward of the immediate reward with all the discounted successive rewards in the future. The objective of the RL agent here is to find the optimal policy π so that by applying this policy, the maximum reward could be generated. This could be done just by taking every action following ϵ-greedy and keep updating the action value function $q(s,a)$ until it converges. As explained in Chapter 2, there are multiple techniques to update the action value function. In this thesis, off-policy Temporal Difference Control, Q-learning technique, is employed by following Equation 4.17. Unlike *Monte Carlo*, Q-Learning does not have to complete one full episode to update the action value function, instead uses the current estimate in the equation. This method proves to reach convergence faster (Sutton and Barto, 2018).

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{4.17}$$

A_t in Equation 4.17 represents the taken action at time-step t, while a represents any possible action from state S_{t+1}. In this model, if the action in $t+1$ is not triggered by disruption, then the action is the service line in the itinerary saved from the previous decision epoch. In the case of disruption, there are two possible actions that will be chosen according to its maximum $Q(S_{t+1}, a)$ value. In this model, the action value function $q(s,a)$ is initiated with zero values for every state and action and will be updated through a series of training episodes. All the notations in the equation refer to Sutton and Barto (2018) textbook. To distinguish the notations between reward and request, for the rests of the report in this thesis, the reward is denoted by Γ.

Algorithm 5 Reward Generator

```
1: Input: Affected request $r$, state of the request $s$, action $a$
2: Output: Immediate reward $\Gamma$
3: Let $I_c$ be the itinerary
4: Start calculating reward $\Gamma$ in the shipment process
5: Wait until $r$ is available at the next terminal
6: if $a$ is a future action (not $I_c(0)$) then
7:     Update request state $s$
8:     Wait until the $r$ is available at the next terminal
9: end if
10: if $r$ is disrupted again while waiting then
11:     if $a$ is a future action then Break
12:     end if
13:     if Request is on board of a service line then
14:         Wait until $r$ is available at the next terminal
15:     end if
16:     Interrupt the waiting process for action $a$
17: end if
18: if Request $r$ is delivered then
19:     Next state $s'$ = 0
20:     Next action $a'$ = 0
21: else
22:     Next state $s'$ = Current state
23:     Next action $a'$ = next service mode in $I_c$
24: end if
25: Grant reward $\Gamma$ for action $a$
```

4.5. Learning Assisted Hybrid Simulation-Optimization

The reinforcement learning technique is integrated with the previous hybrid simulation-optimization to create a complete decision support system in creating a resilient synchromodal framework. The simulation module acts as an environment to simulate the freight network including the disruptions in a stochastic manner. The optimization module acts as a planner providing a matching plan between requests and available service lines with the objective of minimizing shipment costs. Finally, the integration with RL agent adds the capability of learning from experience to address the problem of unknown information about disruption duration. The three modules work together by exchanging information as shown in the flow diagram presented in Figure 4.13 forming a complete model of Learning Assisted Hybrid Simulation-Optimization for hinterland freight transportation under synchromodal framework.

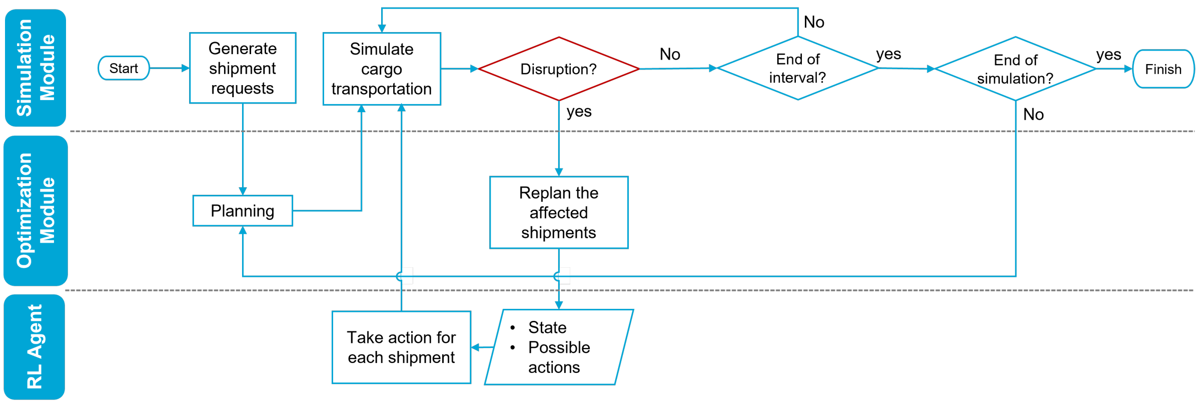


Figure 4.13: Learning Assisted Hybrid Simulation-Optimization Flow

5 Model Verification

In this chapter, verification is performed to evaluate if the proposed model works correctly. A synthetic service network and requests are generated as the input and the results will be evaluated afterwards. A simple heuristic model is constructed to be plugged into the model and act as a centralized planner. Despite the output from the heuristic model is not the optimal solution, it can still be used to evaluate the whole model since the objective of this chapter is to see if the model works rather than to provide the best solution to the network. The proposed model is implemented using *Python 3.10.6*

5.1. Model Input

This model requires five main inputs as elaborated in Chapter 4 encompassing a service network, service lines, requests, cost structure, and disruption profile. The service network and the disruption profile are as presented in Figure 4.2 and Table 3.1. The distance of each node used in the network is presented in Table 5.1. The table consists of matrices containing distances in kilometers for each service mode. The first column is the origin terminals, and the first row is the destination terminals. The dashes indicate that a certain mode service line is unavailable between two terminals.

Table 5.1: Distance Between Terminals

Barge Network					Train Network					Truck Network				
Node	**A**	**B**	**C**	**D**	**Node**	**A**	**B**	**C**	**D**	**Node**	**A**	**B**	**C**	**D**
A	0	70	-	100	**A**	0	-	100	-	**A**	0	80	120	120
B	-	0	-	-	**B**	-	0	60	60	**B**	-	0	70	70
C	-	-	0	-	**C**	-	-	0	-	**C**	-	-	0	-
D	-	-	-	0	**D**	-	-	-	0	**D**	-	-	-	0

The network consists of 2 barge lines, 3 train lines, and 25 truck fleets. The truck service is available in every connection of 2 terminals except between Terminal C and Terminal D. Fixed schedule service is only available for one departure every week. The departure times are generated randomly while considering the interval between services to ensure distributed availability. The departure time only indicates the first-week departure and will be repeated for every 7-day interval, therefore the announcement time values in the table are all 0. The snippet of the service lines input data is presented in Table 5.2. Column K is the service name and columns 'o' and 'o2' are the origin and destination respectively.

The shipment request for the model verification only contains 90 shipment requests. All requests depart from Terminal A with a destination to Terminal C or Terminal D. No request has an end destination to Terminal B to reduce the complexity and only let Terminal B be a transshipment terminal. The snippet of the input data of the request on the first week is presented in Table 5.3. ptime, dtime, and atime are pickup time, due time, and announcement time. Those time parameters are generated randomly while ensuring a well-distributed request throughout the period. The volume (qr) is set using a random

Table 5.2: Service Lines

K	o	o2	Departure (hour)	Capacity (TEU)	Speed (km/hour)
Barge1	TerminalA	TerminalB	30	160	15
Barge2	TerminalA	TerminalD	52	160	15
Train1	TerminalA	TerminalC	76	90	45
Train2	TerminalB	TerminalC	40	90	45
Train3	TerminalB	TerminalD	36	90	45
Truck1	TerminalA	TerminalC	99999	99999	75

number between 10 to 40 with an interval of 5, and the assigned mode (k) is left to 0 and will be filled up during the simulation.

Table 5.3: Shipment Requests

ID	origin	destination	ptime	dtime	qr (TEU)	k	atime
Request1	A	C	15	70	40	0	0
Request2	A	D	42	90	20	0	0
Request3	A	C	48	120	35	0	0
Request4	A	D	50	98	15	0	0
Request5	A	C	88	136	30	0	0
Request6	A	D	50	98	15	0	0
Request7	A	C	34	106	20	0	0

Next, the cost parameters are derived from the study by Zhang et al. (2022). The storage costs are assumed the same for every terminal in the network. Similar to storage costs, the delay penalty is set to be the same for every shipment without considering the different time windows as explained in Chapter 4. The cost parameters are presented in Table 5.4.

Table 5.4: Cost Parameters

	Unit	Barge	Train	Truck
Service Related Cost				
Travel cost (C_k^1)	Euro/km/TEU	0.0213	0.0653	0.2758
Travel cost (C_k^2)	Euro/hour/TEU	0.6122	7.54	30.98
Handling cost (C_k^3)	Euro/TEU	3	18	18
Storage Cost (C^s)	Euro/hour/TEU	1	1	1
Shipment Related Cost				
Delay penalty (C_r^d)	Euro/hour/TEU	1	1	1

5.2. Simulation Verification

To evaluate if the simulation module works correctly, it is verified by adjusting parameters and see if the results change according to the expectation. 10 Scenarios are developed to test if the simulation works as expected. The total cost from the benchmark model is generated to be compared in the verification process. In the benchmark model, only service disruptions are applied. Including the request disruptions in the model could result in unpredictable results because it could possibly generate either higher or lower costs in the end. In contrast, the service disruptions are expected to always increase the total costs. The total cost of the benchmark model is €434,138.

The first four scenarios involve only one request r in the input and compare the costs with manual calculations to evaluate if the process of request component works properly. Scenario 5 and 6 are also applied to request r to test the request parameters by adjusting the release time and due time and see the effect on the storage cost and delay penalty. Scenario 7 and 8 test the effect of adjusting parameters in the service line by reducing the vehicle speed and carrying capacity. The reduction of service capacity is expected to increase the total shipment costs. Finally, scenario 9 and 10 are developed to verify the disruption process in the simulation. Scenario 9 compares the non disrupted scenario with the benchmark scenario and Scenario 10 applies a bottleneck disruption in Terminal B (referring back to the network) and is set to last until the end of the simulation.

The module passes all the scenarios with the results presented in Table 5.5

Table 5.5: Simulation Module Verification Results

No	Description	Expected Results	Actual Results
1	Compare manual calculation for the storage cost	$SC_r = 666$	$SC_r = 666$
2	Compare manual calculation for the handling cost	$HC_r = 1680$	$HC_r = 1680$
3	Compare manual calculation for the travel cost	$RC_r = 728.45$	$RC_r = 728.45$
4	Compare manual calculation for the delay penalty	$LC_r = 80.67$	$LC_r = 80.67$
5	Set an earlier due time for a request	$LC_r > 80.67$	$LC_r = 880.67$
6	Set an earlier release time for a request	$SC_r > 666$	$SC_r > 866$
7	Reducing speed of all service modes	$\sum_{r \in R} TC_r > 434,138$	$\sum_{r \in R} TC_r = 507,336.99$
8	Reducing carrying capacity of fixed schedule service	$\sum_{r \in R} TC_r > 434,138$	$\sum_{r \in R} TC_r = 1,024,381.38$
9	Compare disrupted and non disrupted (nd) scenario	$\sum_{r \in R} TC_r^{nd} < 434,138$	$\sum_{r \in R} TC_r = 417,211.07$
10	Set constant disruption in transhipment terminal	There are undelivered requests	6 requests are undelivered

5.3. Heuristic Model

To fill the centralized planner role, a simple heuristic model is developed. The heuristic model works by assigning a request to a service line with the nearest departure time in the same week. It prioritizes by seeking a compatible fixed scheduled service. If there is nothing available in the same week, the heuristic model will assign the request to the truck service.

Table 5.6: Possible Combinations

ID	Pick-up	Delivery	Services
Comb1	TerminalA	TerminalC	Train1
Comb2	TerminalA	TerminalC	Barge1, Train2
Comb3	TerminalA	TerminalC	Truck1
Comb4	TerminalA	TerminalC	Truck2
Comb5	TerminalA	TerminalC	Truck3

To ensure a compatible route, possible arc combinations are pre-determined according to the service schedule given in the input. The possible combination only provides either all fixed schedule or truck-only service. There is no combination of both for example Barge1 to Terminal A, then transferred to Truck11 to reach Terminal C. Truck service from transshipment terminal (Terminal B) might only be chosen in case of replanning. Furthermore, Truck service from Terminal A to Terminal B is omitted. These restrictions are applied based on the assumption that if a truck service is chosen, then it will transport the request directly to the end terminal. Therefore, there are 26 possible combinations in total consisting of 2 multi-modal services, 4 direct fixed schedule services, and 20 truck services. A snippet of possible combinations is presented in Table 5.6.

Those possible combinations are filtered according to on-going disruptions before being processed in the heuristic model. The combination filtering and mode assignment process using a heuristic model are presented in Algorithm 6 and Algorithm 7. The on-going disruption D could be a terminal or a service line denoted with p, while the set of possible services consists of subsets of service lines combination c, as shown in the column "Services" in Table 5.6.

The filtering and the heuristic algorithm are triggered whenever the planning or the replanning is needed. Using the the available combination, the heuristic model produces a matching plan consisting of old plan and new plan for each request. In the replanning procedure, these two options will be used for the RL agent in the later process to choose between *reassign* the new mode or *wait* by staying with the old mode.

Algorithm 6 Combinations Filter

```
1: Input: Set of ongoing disruption location $D$, Set of possible service combination $S^{pc}$, Set of service
   lines $K$
2: Output: Set of available service combinations $S^{ac}$
3: Initiate $S^{ac}$ with all possible combination in $S^{pc}$
4: for each $p \in D$ do
5:     for each $c \in S^{ac}$ do
6:         if $p$ is in $c$ then
7:             remove $c$ from $S^{ac}$
8:         else
9:             if $p$ is not in $c$ then
10:                for $k \in c$ do
11:                    if $k$'s destination is $p$ then remove $c$ from $S^{ac}$
12:                        break
```

Algorithm 7 Heuristic for Matching Problem

```
1: Input: Request List (could be for planning or replanning) $R^{plan}$, Set of possible available combina-
   tion $S^{ac}$
2: Output: Mode assignment $k$ for every $r \in R^{plan}$
3: for each $r \in R^{plan}$ do
4:     Let $ar$ be the request's release time
5:     Initiate an empty list $S^{cc}$ for possible service line combinations
6:     Initiate an empty list $K_r^{old}$ for the current assigned mode (for replanning)
7:     Initiate an empty list $K_r^{new}$ for the suggested mode assignment
8:     if $r$ has mode assignment $K_r$ (not empty) then
9:         Store $K_r$ as $K_r^{old}$
10:        for each $c \in S^{ac}$ do
11:            Let $Dt_k^c$ be the departure time of the first service in list $c$
12:            if $ar < Dt_k^c$ then
13:                $K_r^{new} = c$
14:                break
15:                if there is no possible combination that fits then
16:                    $K_r^{new} = K_r^{old}$
```

5.4. Hybrid Simulation-Optimization Verification

The verification method of the integration of a replanning module in the simulation together with affected request detection is different from simulation verification. The additional module is highly related to the disruption occurrence. Therefore, instead of adjusting parameters, the verification is applied by testing the model in different random disruption scenarios, by setting a different random seed, and seeing how it behaves.

The first verification is performed to evaluate the affected request detection. The random seed is set to 10 and 11, and a sample of disruption is evaluated from the model output logs. Two random seed values are required to have all needed disruption samples. The sample disruption is a service disruption on TerminalB, Train2, and Barge2. Referring back to the network in Figure 4.2, the expected affected requests for disruption on TerminalB are the ones in Terminal A and are assigned to Barge1. Disruption in Train2 is expected to affect requests with end destination to TermincalC via Barge1 and Train2. It affects the requests in TerminalA or on Barge1. Disruption in Barge2 also only affects the request in TerminalA with the end destination to TerminalD via Barge2.

Table 5.7 presents the result of the verification. The result shows that the algorithm manages to determine the affected requests accurately as defined in the logic

Table 5.7: Affected Request Detection Verification

No	Disruption Location	Affected Requests (Model Output)
1	seed=10 Day 15 - 12:45 Disruption at TerminalB	**Request 23, Request24**. These requests are in TerminalA and are assigned to Barge1 with one of them destined to TerminalC and the other to TerminalD
2	seed=10 Day 16 - 10:26 Disruption on Train2	**Request13, Request23**. Both requests are on board (Barge1) when the disruption occurs
3	seed=11 Day 2 - 16:37 Disruption on Barge2	**Request02, Request04, Request06, Request08, Request10**. These requests are assigned to Barge2 during the weekly planning and are still waiting at Terminal A

The second step is to verify the heuristic model proposed earlier. The combination filter and heuristic algorithm are verified together by taking disruption samples from simulating different random seed. The disruption samples are selected both for single disruption and overlapping disruption. Overlapping disruption is when a disruption occurs while another disruption in a different location is still going. In these disruption samples, the replanning results are evaluated whether any of the replanned requests are assigned to disrupted modes. If none, then the verification is successful. The results are also compared to the expected reassignment solution space to evaluate if the heuristic algorithm works according to the designed logic.

For this verification, the random seed is set to 20 and 30 to generate enough variations in the samples. The results are presented in Table 5.8. The expected results are described in the second column according to the location of the requests when the disruptions occur and their end destinations. The model manages to eliminate the disrupted location from the solution space indicated by none of them in the actual reassignment result. Moreover, the actual reassignment fits the solution space described in the expected reassignment. This indicates that the heuristic model passes the verification.

5.5. Learning Assisted Model Verification

The last piece of the model is integrating the RL agent into the hybrid simulation-optimization. To verify if the integration works according to the conceptual logic, the three main functions inside the algorithm: action generation, reward generation, and action value function updating are evaluated. Four scenarios are created to cover all the function verification. Within those scenarios, three sub-scenarios are developed to test the reward generation.

The first scenario is developed to check the action value function updating. The model runs for a couple of episodes in random seed=0 and is verified if the action value function is updated for the same disruption in a different episode. The second scenario is to verify the action generation. After the first scenario, the action value function is already updated and action generation can be evaluated if RL agent selects an action based on the action value function. The model runs using random seed=0 and each decision made by RL is recorded and evaluated. The third scenario is to test the explorative actions by adjusting the ϵ value. Finally, the last scenario is to test the reward generation by comparing

Table 5.8: Heuristic Model Verification

No	Disruption Location	Expected Reassignment	Actual Reassignment
1	Seed=20 Day 9 - 18:41 Disruption on Barge1 Affected: Request13, 14, 16	**Request13** reassigned to Train1 or Truck to Truck5 **Request14 and 16** reassigned to Barge2 or Truck21 to Truck25	Request13: Train1 Request14: Barge2 Request16: Barge2
2	Seed=20 Day 15 - 21:04 Disruption on Barge2 Affected: Request22, 24	**Request22** reassigned to Barge1 or Truck6 to Truck10 **Request24 and 16** reassigned to Train3 or Truck16 to Truck20	Request22: Barge1 Request24: Train3
3	Seed=20 Day 36 - 17:48 Disruption at TerminalB Disrupted service lines: Barge1 and Truck6 to Truck10 Affected: Request51, 52	**Request51** reassigned to Train1 or Truck1 to Truck5 **Request52** reassigned to Barge2 or Truck21 to Truck25	Request51: Train1 Request52: Barge2
4	Seed=30 Day 16 - 07:34 Disruption on Train3 and Train1 Affected: Request25, 29	**Request25** reassigned to Barge1 or Truck1 to Truck5 **Request29** reassigned to Barge1 or Truck1 to Truck5	Request25: Truck3 Request29: Truck2

the output in the logs with manual calculation for 3 three cases elaborated in Chapter 4 section 3.4.4 about the reward generation.

The result presented in Table 5.9 indicates that the model works according to the conceptual logic. The action and reward generation function shows the same result in comparison with manual calculations. The action value function also shows updated values after several training episodes.

Table 5.9: Learning Assisted Model Verification

No	Description	Expected Results	Actual Results
1	Model runs for 5 episodes and Q(s,a) updating is checked for a specific disruption on a specific request	Q(s, a) for both reassign and wait are 0 in the first episode and change for every episode	Episode 1: Q(s,a) wait = 0 Q(s,a) reassign = 0 Episode 5 Q(s,a) wait = -6389.6 Q(s,a) reassign = -4709.35
2	The ϵ value is set to 0.05 and first 30 decisions are evaluated	Take most of the action based of higher Q(s, a)	30 out of 30 decisions made according to higher Q(s,a)
3	The ϵ value is set to 0.4 and first 30 decisions are evaluated	Take more explorative action than ϵ = 0.05	24 out of 30 decision made according to higher Q(s,a)
4	Compare reward calculation with manual calculation for 3 cases		
	Case1 Case2 Case3	Reward for a1 = -6380.64 Reward for a1 = -1704.41 Reward for a1 = -588.66 Reward for a2 = -485.56	Reward for a1 = -6380.64 Reward for a1 = -1704.41 Reward for a1 = -588.66 Reward for a2 = -485.56

5.6. Preliminary Results

By embedding the heuristic algorithm, the learning assisted hybrid simulation-optimization is complete. The time horizon for each simulation is ten weeks. The three possible policies consisting *always wait*, *always reassign*, and *optimal policy* are run using the model. The *always wait* is a policy without having a replanning procedure and represents a naive intermodal framework. The *always reassign* is a policy that triggers the heuristic algorithm every time a disruption occurs and always accepts the solution. This represents a naive synchromodal framework without the RL agent. The last policy is by employing the RL agent to find the *optimal policy* using ϵ-greedy policy (refer back to Chapter 4). To balance the exploration and exploitation.

In this verification, the model runs for 100 episodes to train the RL. In the episodic training, the RL agent applies the ϵ-greedy policy and updates its value function. After the episodic training finishes, the model runs again for one time with the RL agent, this time, applying a *greedy policy*. In the *greedy policy*, the RL agent always chooses the action with the highest action value. The results of the total cost of each policy are compared to see how the model performs with different policies. The random seed is set to 0, therefore each episode has the same disruption occurrences. This simplification is only used for verification, thus the performance improvement of RL agent could be seen despite the small number of training episodes.

Table 5.10 presents the result of *always wait* and *always reassign* policy. The result shows that in this particular case, the *always reassign* policy outperforms the *always wait* by having around 17% lower total cost. The preliminary results of these two policies show that the *'always reassign'* policy manages to reduce the storage cost and delay penalty by taking longer routes indicated by higher travel costs. Additionally, the different handling cost parameter of barge services with the other two modes of services also creates lower total handling cost in the *'always reassign'* policy.

Table 5.10: Two Policies Comparison

	Always Wait	**Always Reassign**
Total Storage Cost	136,137.55	99,039.67
Total Handling Cost	84,078.00	71,046.00
Total Travel Cost	100,561.85	116,529.37
Total Delay Penalty	77,166.33	42,430.45
Total Cost	397,943.73	329,045.49

Figure 5.1 and 5.2 present the results of the RL agent training throughout 100 episodes of simulation. Figure 5.1 shows the total reward generated every episode and Figure 5.2 presents the total costs. The RL agent manages to generate more reward throughout the training period and indicates a convergence after around 40 training episodes. Since the reward is the negative cost, the RL agent performs better if the total rewards are closer to 0.

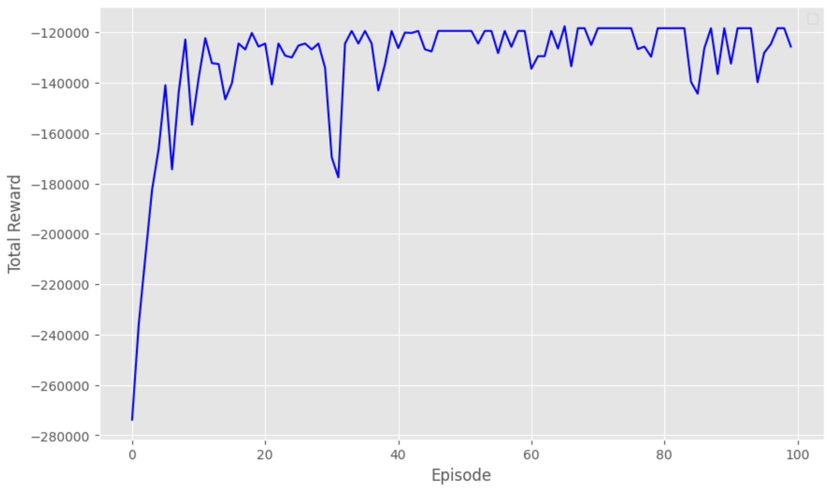


Figure 5.1: RL Agent Reward Generation

In contrast, the total costs for each training episode in Figure 5.2 shows a decrease over time. This is coherent with the generated reward over the training period. The green dashed line is the total cost of *always wait* policy while the red dashed line is the total cost of *always reassign* policy. The RL agent outperforms the two other policies way before it reaches convergence.

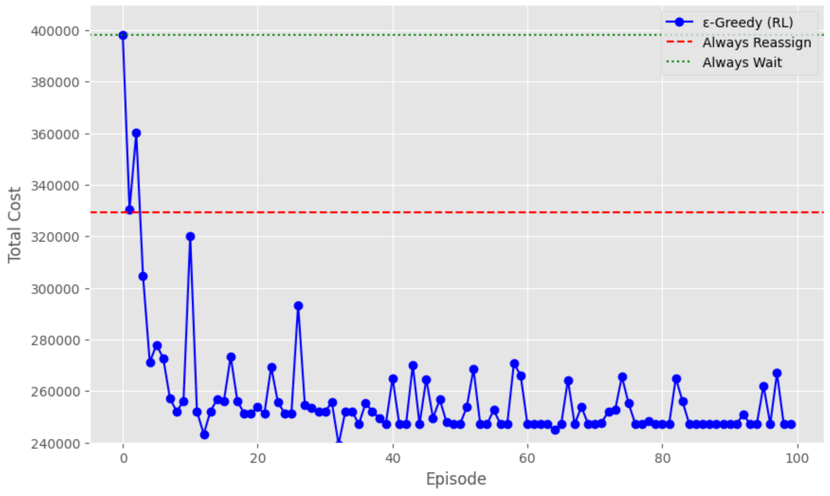


Figure 5.2: RL Agent Training Result

The result of the model runs with *greedy policy* along with the comparison with the two other policies is presented in Figure 5.3. The preliminary results show that the RL agent performs better by keeping the travel cost lower than *always reassign* and reducing the delay and storage costs even more, in this particular case, by having 24% lower total cost compared to *always reassign* policy. Given the parameters and disruption cases in this verification, it is important to note that the better performance of *always reassign* over *always wait* policy does not imply the superiority of one policy over the other in all cases. A different sequence of disruptions may result in a different outcome. As for the learning-assisted model, it is expected to always perform better, or at least mimic the performance of the best policy between the other two. This result is yet to be seen and proven in the numerical experiment with a larger network and larger shipment numbers.

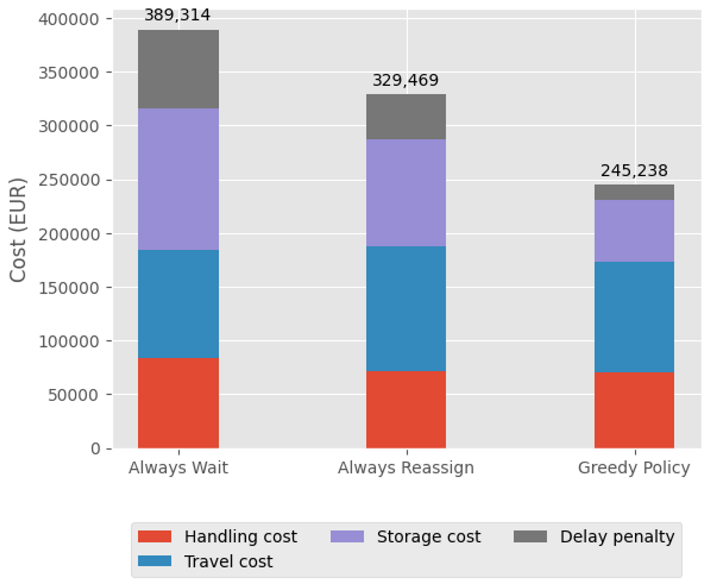


Figure 5.3: Total Cost Comparison (Verification)

The total costs between the three policies are compared in more detail presented by Figure 5.4. Each

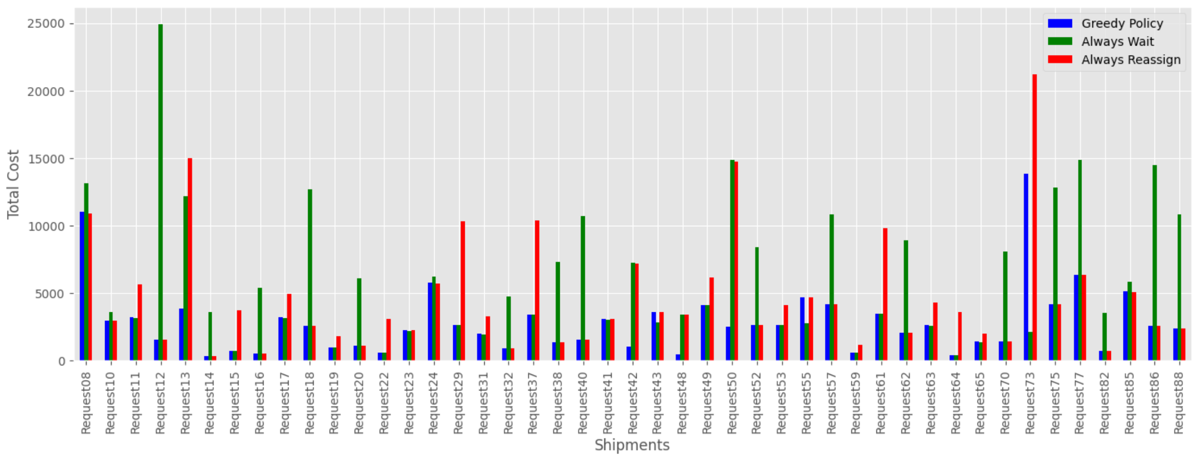


Figure 5.4: Shipment Cost Comparison

bar shows the cost for each shipment for different policies characterized by different colors. The chart only includes shipments that are affected by the disruption therefore having different across different policies. The chart shows that the RL agent, in most cases, effectively selects the action—either wait or reassign—that results in the lowest cost. In some cases, the greedy policy does not yield the lowest cost. Take the example of request 73, the wait policy generates much lower shipment costs. This might happen due to capacity restriction of the service line. Since the current model does not consider capacity, each action taken for a different shipment can cause capacity problems on other shipments, thus, that shipment needs to wait for the next departure of the service.

6
Case Study

This chapter discusses the implementation of the proposed model using a larger demand dataset and a more complex network using a real-world data. On top of that, a more sophisticated optimization model is plugged into the optimization module, replacing the FIFO heuristic model, to provide more cost-effective matching solutions and capable of handling larger networks. Furthermore, a numerical experiment is conducted to perform a sensitivity analysis using the complete model.

6.1. Case Study Input

For the case study, the service network uses a real port-hinterland network, with synthetically generated request datasets ensuring well-distributed values. The cost parameters and disruption profile remain the same with those used during the model verification in Chapter 5. The service network is adapted from the European Gateway Services (EGS) network, now known as Hutchinson Ports Europe Intermodal (HPEI). This network connects the Hutchinson Ports ECT in the Port of Rotterdam to inland terminals in the Netherlands, Belgium, and Germany through the Rhine-Alpine corridor. The details of the network, including distances between terminals and the service schedule, are derived from a study by Zhang et al. (2023).

The network comprises 10 terminals, with 3 of them located in the Port of Rotterdam and the rest scattered across inland terminals. It includes 49 barges, 33 trains, and 34 truck service lines operating between these terminals. The network is illustrated in Figure 6.1. Each service line represents one origin and one destination with a unique departure time each week. Some connections have multiple lines for the same mode of service from the same origin-destination pair but with different departure times, indicating service frequency. As explained in Chapter 4, each service line only serves two nodes, with no modes providing multiple stops at different terminals. While truck services have one line for each origin-destination pair, it is assumed that truck service capacity is unlimited and can accommodate multiple departures as per assigned requests. The carrying capacities of barge and train services are 160 TEU and 90 TEU, respectively. Travel speeds are 15 km/hour for barges, 45 km/hour for trains, and 75 km/hour for truck services. The distances between terminals vary for each service mode. All datasets used in this case study can be found in the Appendix C.

Requests are generated randomly according to the service capacity proportion of each origin-destination pair. For instance, the number of services connecting Delta to Nuremberg is much higher than those connecting Delta to Willebroek, so the request volume from Delta to Nuremberg is proportionally higher. Requests are announced in batches, each consisting of multiple shipment requests. The announcement time for each batch is set to the 6th day of each week, except for the first batch, which is announced on the first day of simulation. Since this thesis focuses on cargo flow from the main port to the hinterland, only three terminals in the Port of Rotterdam (Delta, Euromax, and HOME) are considered as origins, with the other seven being destination terminals. Release times are generated randomly between 0 to 4 days after the announcement time, while due times are selected randomly between 2 to 7 days after the release time. Shipment volumes range from 5 to 30 TEUs. For the default scenario, 800 requests

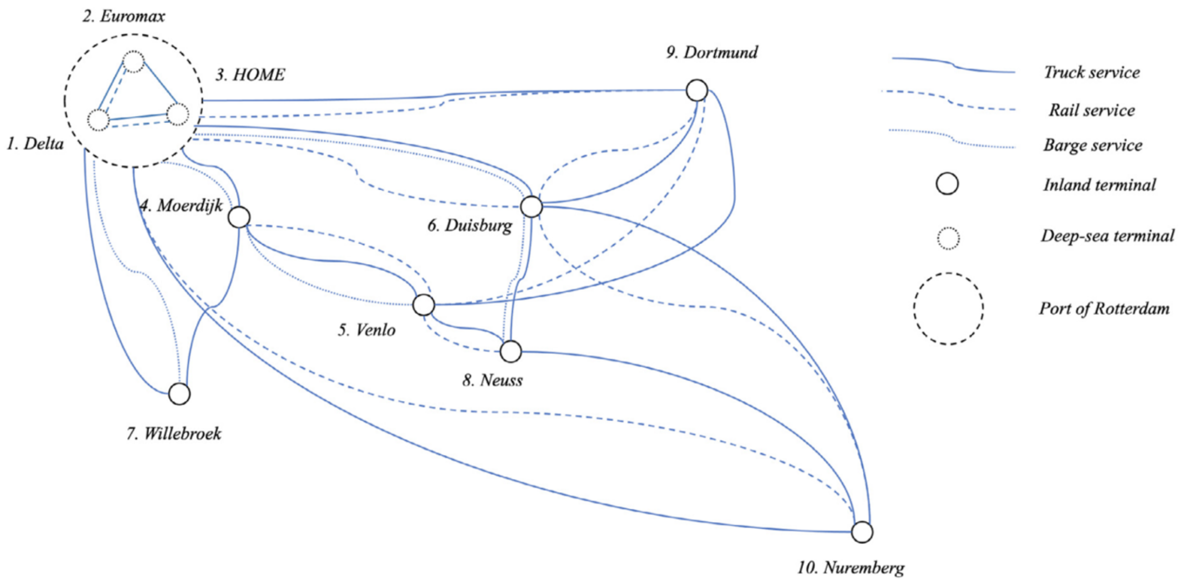


Figure 6.1: EGS Service Network
source: (Zhang et al., 2023)

are generated and announced proportionally over 12 weeks. This default demand scenario is used to train the RL agent.

6.2. Optimization Model Plug-In

For the case study, a matching problem optimization algorithm inspired from Filom and Razavi (2023) and Guo et al. (2021) is integrated as a plug-in to the optimization module, replacing the naive heuristic algorithm in Chapter 5. The model is originally designed as a learning-based robust optimization consisting of a prediction module and a decision module. The prediction module is used to derive uncertainty sets for road travel time. The uncertainty sets then are fed to the downstream robust optimization model in the decision module. Based on uncertainty sets, the robust reformulation is done to convert the robust optimization model to a deterministic model which is solved by mathematical solvers. However, from this comprehensive framework, only a fragment of the decision module, the deterministic optimization model, is plugged into the optimization module of the learning-assisted hybrid simulation-optimization, illustrated in Figure 6.2.

The inputs of the optimization model are shipment requests and service lines. It uses a path-based approach and employs a preprocessing algorithm to generate paths from available service lines to reduce the computational burden within the optimization model. The path generation algorithm consists of all possible services that satisfy the spatiotemporal requirements and transshipment feasibility. These paths are essentially the same idea as ”arc combinations” used as input in the heuristic model (Chapter 5). The difference is that the arc combinations in the previous heuristic model are pre-determined manually, while in this optimization model, a pre-processing algorithm is constructed, allowing more scalable and versatile utilization.

The path is generated according to each service line’s departure and arrival time and each path is attributed with all the cost components including the transshipment and storage costs if any. The cost calculation of each path considers all the modes of service within the path. If there are two modes of service in one path, then the travel cost is the total of those two modes’ travel costs given the travel distance and time in the path. Since the departure and arrival time is known, the storage time is calculated according to the service schedule of the mode of service in the path used for calculating the path storage cost. For the truck service, the storage time is always 1 hour because of the assumption that the truck is always available 2.5 hours after the assigned request is released and has a 1.5-hour time window used for handling within those 2.5 hours. The path capacity is determined based on the smallest service line's capacity within one path. For instance, if a path consists of a barge and a train, the path capacity is 90 TEU because the train capacity is 90 TEU and it is smaller than the barge capacity. By limiting the maximum number of arcs within one path to three, from the total 82 fixed schedule services and 34 truck lines, the algorithm generates 796 possible paths in the network.

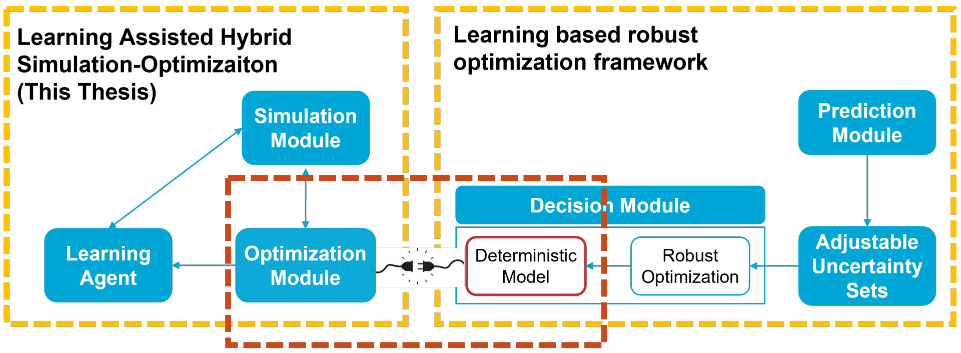

Learning Assisted Hybrid Simulation-Optimizaiton (This Thesis)
Simulation Module
Learning Agent
Optimization Module
Learning based robust optimization framework
Prediction Module
Decision Module
Deterministic Model
Robust Optimization
Adjustable Uncertainty Sets

Figure 6.2: Integration with an Existing Optimization Model

The objective of the optimization model is to minimize the total transport cost including travel, transshipment, storage delay, canal, handling, and emission. This model was originally designed using a case study in Great Lakes, Canada, therefore there is a canal cost that applies to that context. The objective function is adjusted to fit the model with the context of the case study in this thesis. The canal cost and emission cost are eliminated from the objective function resulting in the equation presented in Equation 6.1.

The deterministic mathematical formulation of the problem is presented as follows:

$$\begin{aligned}\min_{x^t,y^t} \Bigg(& \sum_{r\in R} v_r(1-y^t) + \sum_{r\in R}\sum_{s\in S} c_s x^t_{rs} u_r + \sum_{r\in R}\sum_{s\in S} c^T_{ri} x^t_{rs} u_r + \sum_{r\in R}\sum_{i\in I} c^S_i x^t_{rs} \tilde{S}^s_{ri} u_r \\ & + \sum_{r\in R} c^D_r \tilde{d}^r_{ri} + \sum_{r\in R} c^L_{ri} u_r \Bigg)\end{aligned} \tag{6.1}$$

The objective function in Equation 6.1 attempts to maximize the number of matches and minimize the total transportation cost which includes six terms: the first term enforces the model to match shipment requests as many as possible. The second term determines the transportation cost, the third term denotes the transshipment cost, the fourth term accounts for storage cost, the fifth term shows the delay penalty costs, and the final term represents the loading/unloading cost. The mathematical model works according to 22 constraints as presented in B.

The optimization model is designed to work using a rolling horizon approach to cover the dynamics of requests' announcement time and match all the requests with the same announcement time to the possible paths. Each decision making process for each announcement time is called a decision epoch. After each decision epoch, the optimization model updates the free capacity of the paths according to the matching decision. This updated free capacity is used in the next decision epoch, therefore capturing the effect of each decision epoch on the capacity. The discrete time step used in this model is an hour and with each time step the optimization model fetches the announced demand. The time horizon of the whole rolling horizon only covers one departure and does not simulate a recurring operation, therefore neglecting the possibility that unmatched requests can be covered in the next departure of the same path.

By integrating the optimization model into the optimization module, the planning process will generate shipment plans that minimize costs. As a result, the initial plan derived from offline planning will have a lower cost compared to the heuristic algorithm used in Chapter 5. This leads to a different performance for the *always wait* policy. Since the heuristic algorithm does not offer the optimal solution, the *always wait* policy performs poorly in verification. However, integrating this optimization model is expected to enhance the performance of the *always wait* policy.

6.3. Model Integration

Even though from the conceptual perspective, the optimization model is compatible with the optimization module plug-in, there are some adjustments needed for the integration. The adjustments are required to address the following differences between the two frameworks.

1. Different assumption in truck service. The hybrid simulation-optimization considers unlimited capacity and availability of trucks while it is limited in the plug-in model.
2. The plug-in model updates the path capacity after each hourly-based decision epoch, while the demand in the hybrid simulation-optimization is captured on a weekly basis.
3. The plug-in model has much less running time sensitivity compared to the learning-assisted hybrid simulation-optimization which needs a quick runtime for learning agent episodic training.

6.3.1. Adjustment in Path Generation Algorithm

The first adjustment in the integration part is in the path algorithm. In the plug-in model, the path generation algorithm considers a scenario-based truck service and affects the combination of service lines in generating the paths. The truck service also affects the path capacity due to the limited capacity. In the integration, the truck is always available whenever any requests are assigned to a truck service, therefore every path has an additional path by replacing one of the service lines in the path with a truck service as long as a truck service line is available in that specific origin-destination pair. Since the capacity of the path is based on the smallest capacity of service mode in the path, the inclusion of truck services with unlimited capacity does not affect the capacity.

6.3.2. Adjustment in Capacity Updating

The second adjustment is to address the capacity updating problem. In the plug-in model, because it works on path-based approach, the matching between one request to a certain path can affect the free capacity of other paths that consist the same service line. For instance, if a request is assigned to path1 consisting of Barge1 and Train2, this request will also consume the free capacity of the other paths that have Barge1 and Train2 within it. Therefore, a capacity updating is needed. The capacity is updated after each decision epoch, meaning the capacity is updated hourly. It works in the plug-in model because the total volume of demand announced hourly is less likely to exceed the free capacity.

In the hybrid simulation-optimization model, the planning module is triggered weekly capturing all the demand announced in the previous week and all the unmatched demand from the previous planning epoch and recovering each path capacity after delivering goods to the destination terminal. If the approach from the plug-in model is not adjusted, the path capacity is updated only after the planning process each week is done and it will result in a capacity constraint violation because of the huge volume of total demand throughout 1 week and fetched in one decision epoch. All requests might be assigned by respecting the free capacity of each path at the beginning of the planning phase, but it does not consider the capacity reduction due to the assignment to the same service line from a different path during the matching process.

The adjustment is done by modifying the rolling horizon approach. First, the function of the original rolling horizon is removed, making the plug-in model static. Then, it is triggered only during the planning and replanning phase. During the planning process, all the requests announced in the previous week are populated and passed as one input to the optimization module. From the perspective of the plug-in model, referring to its original algorithm, the requests are considered to have the same announcement time and it is the time when the optimization module is triggered. However, to address the capacity updating problem, the rolling horizon approach is needed. This time, a dummy announcement time is embedded in each request. The first request in the list will have an announcement time equal to one, while the request number n will have an announcement time equal to n. This modification in the rolling horizon algorithm will make the plug-in model match the requests one by one and update the free capacity of paths after each request matching. This means that one decision epoch is only providing a matching solution for one request.

6.3.3. Addressing the Run-Time Problem

In the model with a heuristic algorithm plugged into the optimization module, it takes a very short time to finish one training episode. It takes around 2 hours to have a 20000 episodic training. in contrast, by integrating an optimization algorithm, with the given input, the integrated model needs around 6 to 7 minutes to complete one episode. This is mainly because the optimization algorithm has much heavier computational work compared to the heuristic algorithm used during verification (Chapter 5. Additionally, the scale of the network and the number of instances used in this case study is larger. With the 6-minute run-time, it is impossible to give the RL agent a sufficient number of episodic training to see its improved performance. Therefore, adjustment is required to decrease the solution time.

It is identified that the main bottleneck of the long run-time is the solution time of the optimization model. Each time the simulation module triggers the optimization module, it takes time to generate matching solutions for each request. The planning phase takes longer than the replanning because in the planning the number of requests to consider is much larger compared to affected requests as input in the replanning phase. Therefore, the first modification is to separate the planning process from the episodic training. The optimization algorithm is run separately before starting the episodic training and matches all the 800 instances at one time. The matching results are then attached to each request in the input, so, instead of having an empty list of assigned mode k, each request has an itinerary in the shipment input table for the simulation. Without having the separation of the planning process, the simulation triggers the optimization module every week in every episode and always gets the same matching results. This separation approach saves all the unnecessary computational burden that only offers the same matching results.

To perform this separation, a modification in the optimization model needs to be made. The current rolling horizon does not consider a recurring process of a service, while in the simulation, the service is available for the next week after each departure. The modification is made by tweaking the algorithm so the departure time in the path database is updated if an announcement time of the shipment in the input is later than the service departure time. To anticipate any unmatched shipment, the simulation still triggers the optimization module but only passes the requests that do not yet have itineraries with them. This way, the computational burden during the planning phases can be significantly reduced.

The next bottleneck is in the replanning process. The computational burden in the replanning phase could vary depending on the number of affected requests. A modification is made to reduce the number of optimization algorithm triggers during the replanning process. It is by providing a solution pool consisting of K-best solutions for each shipment during the planning process. Instead of providing only one solution during the planning phase, the optimization algorithm generates multiple solutions consisting of a second, third best solution up to ten solutions in the pool. These solutions are used during the replanning process as backup itineraries. The simulation algorithm will eliminate the disrupted itineraries and take the best from the remaining solutions. Whenever a shipment departs from one terminal, it also eliminates other solutions that use different routes from that terminal. The optimization algorithm could still be triggered in the replanning process if there are shipments with no remaining backup itineraries in the solution pool. This approach could reduce the number of triggers during the replanning and ultimately reduce the runtime. However, it could also create suboptimal solutions since it does not consider all the constraints in the real-time. This will be a subject of analysis of the optimality gap in the later section by comparing the model using the default approach which triggers the optimization algorithm whenever is needed.

The solutions pool approach reduces run-time but introduces a risk where reassigning requests could consume the capacity of undisrupted shipments. For instance, if two disrupted requests, each with 20 containers, are both reassigned to Barge1 from the second-best solution in the pool, and Barge1 only has 30 units of free capacity, an issue arises. An undisrupted request originally scheduled for Barge1 might not be loaded due to the capacity limit, causing it to miss its departure and wait for the next one. This issue, present in the previous model due to the heuristic algorithm's disregard for capacity constraints, was supposed to be solved by the optimization model that considers these constraints. However, it re-emerges with the solution pool approach.

This problem is significant because the reinforcement learning (RL) algorithm cannot account for the additional costs incurred from undisrupted requests missing their departures. To address this, an al-

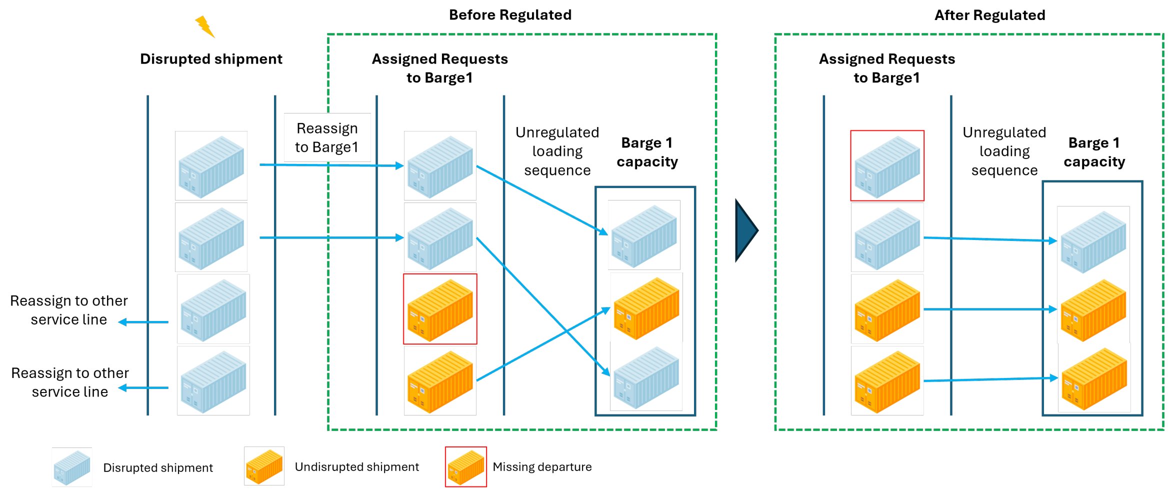

Before Regulated
After Regulated
Disrupted shipment
Assigned Requests to Barge1
Reassign to Barge1
Unregulated loading sequence
Barge 1 capacity
Reassign to other service line
Reassign to other service line
Assigned Requests to Barge1
Unregulated loading sequence
Barge 1 capacity
Disrupted shipment
Undisrupted shipment
Missing departure

Figure 6.3: Loading Sequence Regulation

gorithm was developed to regulate the loading sequence by prioritizing undisrupted requests. This ensures that only reassigned requests might miss the departure due to capacity issues. The extra costs from these delays penalize the RL agent, providing valuable experience to guide future decisions. The illustration of this is presented in Figure 6.3. The summary of the model adjustment for integration work is presented in Table 6.1.

Table 6.1: Adjustments for Model Integration

	Before Adjustment	**After Adjustment**
Path Generation Algorithm	Scenario based truck service	Flexible truck service, adding more combination of mode in the path database
Capacity Updating	Update the free capacity for every decision epoch (could include multiple requests in one decision epoch	Update the free capacity for every one request matching decision
Planning Process	Triggered every one planning interval (one week in simulation)	Provide all the itineraries for all requests in advance and use the matching solutions in every start of new episodic trainig as input
Replanning Process	Triggered every time there is an affected request	Create a solution pool during the planning process, and take the next best solution for the affected request
Loading Sequencing	Unregulated loading sequence	Prioritizing non-disrupted request

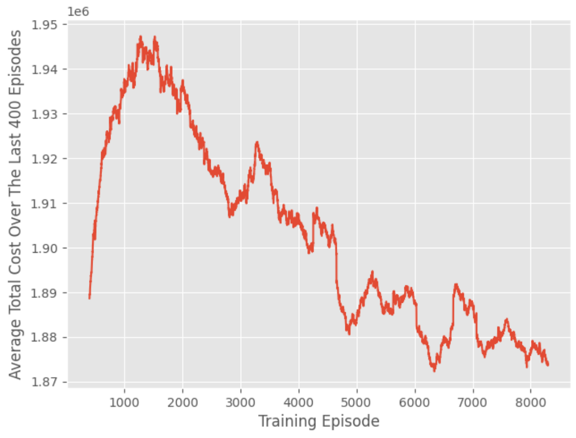
6.4. Model Training

The first step in evaluating the performance of the model is to train the RL agent. The default demand scenario consisting of 800 requests is used and the requests are distributed across 12 weeks period making it announced around 70 requests each week. To generate the path, the travel cost for each service line must be calculated. With all the properties of each service line and two parameters of travel cost comprising the time-related and distance-related cost, the total travel cost for each service line can be calculated.

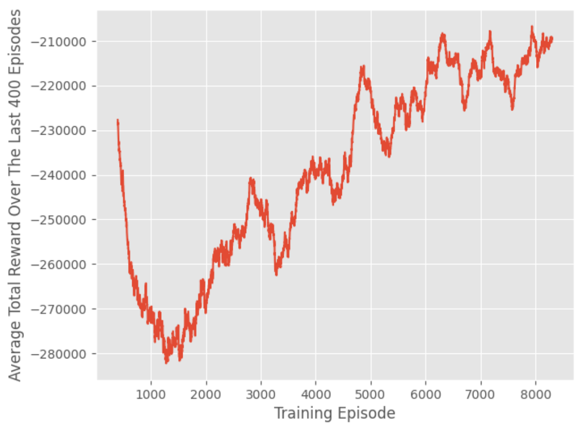
The simulation is set to have a 14-week duration to make sure all the requests are delivered in case some of them have long delays resulting in more representative results for all delivery processes. Each episode of training has a different random seed to ensure different disruptions. In the training, only the service disruption is included, because the demand disruption does not trigger the RL agent (refer back to discussion in Section 4.2.2). Finally, the model is set to have 50,000 training episodes.

6.4.1. Initial Training Result

After running for over 8,300 training episodes, which took approximately 70 hours, the model was evaluated to ensure the training was progressing as intended before continuing with additional episodic



(a) Total Cost - Episodic Training



(b) Total Reward - Episodic Training

Figure 6.4: Episodic Training Result

training. Figure 6.4 illustrates the RL agent's performance improvement over the training period. To reveal the underlying trend and remove the noise, the rolling mean technique is applied to the data with 400 episode window size. Chart 6.4a displays the total cost per episode, where a lower cost indicates better performance. In contrast, chart 6.4b shows the reward per episode, with higher rewards signifying better performance. Over the course of 8,300 training episodes, the RL agent demonstrates a downward trend in total cost and an upward trend in rewards, indicating that it is learning to take better actions based on its experience from previous episodes.

The performance of the trained RL agent is tested through a *greedy policy* against the two designated naive policies discussed in Chapter 5: "always wait" and "always reassign". The performance is evaluated through 5 different disruption datasets. The first one uses the default disruption dataset, the rest are designed to have a higher occurrence probability to apply more disruptions in the network. Figure 6.5 presents the total cost comparison for each policy in each disruption set. D1 to D5 represent the disruption sets where D1 has the less frequent occurrence probability and D5 with more disruptions.

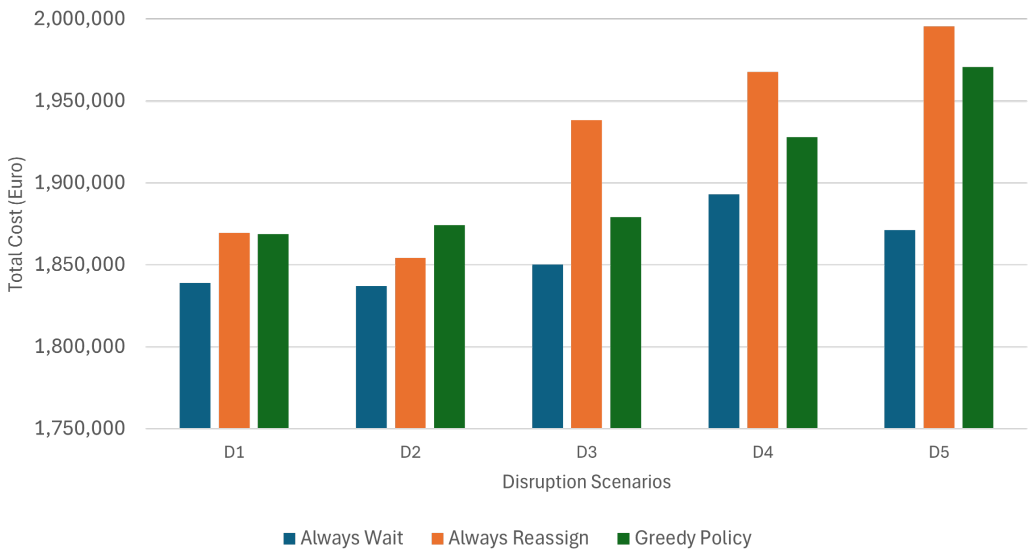


Figure 6.5: Performance Comparison

6.4.2. Initial Findings

From this result, it appears that the RL agent's greedy policy does not generate the lowest costs in all scenarios. However, it still outperforms the *always reassign* policy. On this basis, a more thorough investigation was carried out to get a better conclusion. The simulation provides the logs of events, and within it, action values in each RL's decision epoch are recorded. In the default scenario, out of 57 actions taken in response to disruptions, 45 were in states the RL agent encountered rarely or never before. When the agent has only experienced a state once, it has only tried one action (either waiting or reassigning) and has no knowledge about the alternative option, indicated with action values equal to 0. This limitation contributes to the RL agent's performance being inferior to the *always wait* policy.

Another finding is that the *always wait* policy outperforms other policies because most disruptions do not cause delays in the network, especially those involving service lines. Considering the operational time window from when a service line is available at the origin terminal to its arrival at the destination terminal, most disruptions occur during non-operational times and thus do not cause delays in the network. For instance, if a barge is available for loading at 01:30, departs at 03:30, and arrives at the destination terminal at 21:00, the operational time is about 22 hours, with the rest of the week being non-operational. Therefore, disruptions typically happen during non-operational times and do not cause delays. Although some terminal disruptions may result in late arrivals and missed departures, this is infrequent. Consequently, the RL agent has limited opportunities to demonstrate improvements in handling disruptions. Nonetheless, it is also important for the RL agent to learn to take wait actions when disruptions do not cause delays. However, the agent fails to do so due to insufficient experience.

6.4.3. Improvement for Episodic Training

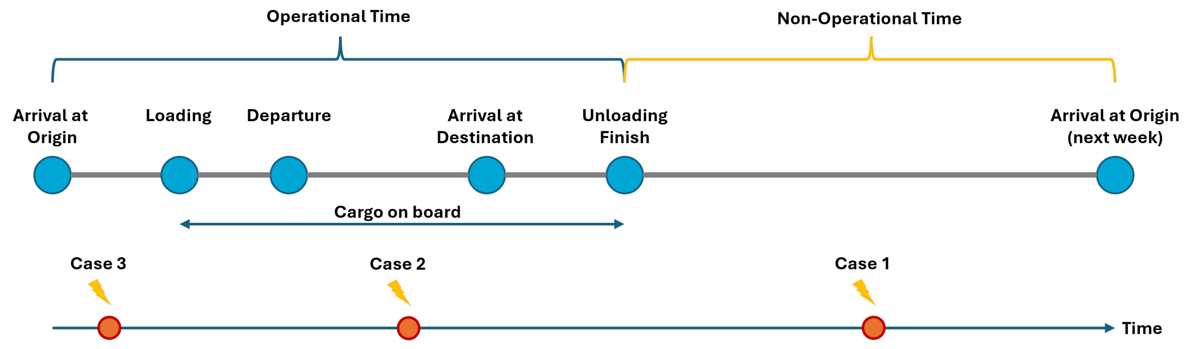


Figure 6.6: Disruption Cases in Service Line

To enhance the quality of training, the disruption scenarios need to be modified. From the input side, it has been observed that several service lines are never utilized throughout the simulation. This could be due to an imbalance between supply and demand or synchronization issues caused by limitations in the service schedule dataset. Furthermore, it is crucial to understand how disruptions occur in the simulation and their effects on requests, especially for disruptions in the service lines. Figure 6.6 illustrates an example of a disruption in one service line, along with its impact on the requests within the simulation. Generally, there are three types of disruptions:

1. **Case 1 - Non-operational Time Disruption:** This occurs during non-operational hours. While some requests might be affected because the service line is part of their itinerary, it is unlikely to cause delays unless the disruption happens right before departure and lasts long enough to delay the departure.
2. **Case 2 - In-transit Disruption:** This occurs between the loading at the origin and unloading at the destination terminal. This causes delays to the shipment already on board, but these shipments cannot alter their itinerary as they are already in transit. Therefore, they are excluded from the affected requests (refer to Section 4.3.1). The requests affected are those assigned to this service for the next departure. However, since the disruption is unlikely to last a week, it generally does not create delays for these requests.

3. **Case 3 - Pre-loading Disruption:** This occurs before shipments are loaded onto the service. This can cause delays in departure from or arrival at the origin. In this scenario, requests might be affected, and rerouting to a new itinerary could be an option.

From the above elaboration, it is clear that case 3 is more critical compared to the other two in determining the RL agent's performance. By exposing more disruptions of case 3 in the training process, the RL will learn to choose better action between reassigning and waiting depending on the type and duration of the disruption. On this basis, there are 2 main modifications applied to the model.

First, the disruption generator will prioritize applying disruptions within a smaller time window of the simulation horizon. Instead of considering all the service lines in the network, the disruption generator will focus on those within the operating time window. This approach significantly reduces the number of potential disruption locations. Figure 6.7 illustrates the operational time window over a week. The blue bars indicate operational time windows of service lines, while the rest represent non-operational times. The procedure for applying disruptions begins by sampling a start time t (red dashed line) for the disruption. Previously, the generator considered all the service lines at time t, while now, only the blue bars (operational times) at time t are considered to get disrupted indicated by yellow x marks in the figure.

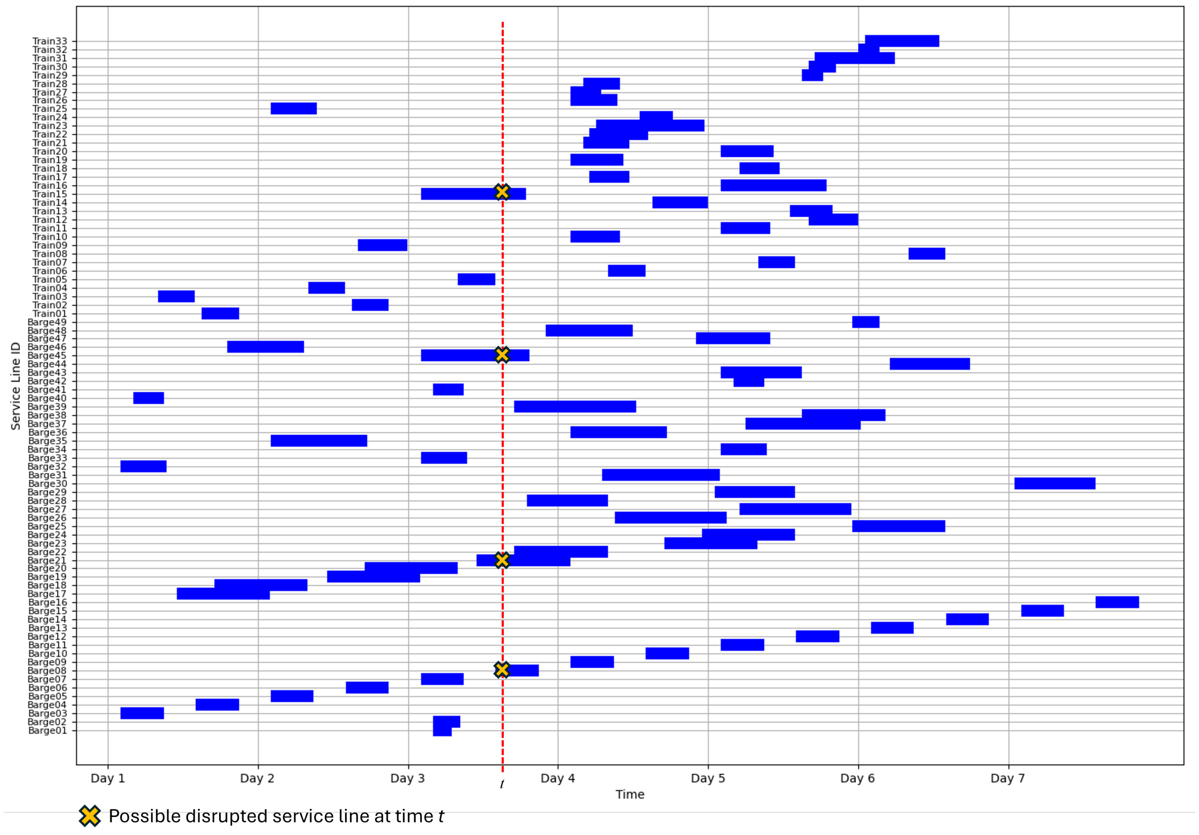


Figure 6.7: Service Lines Operational Time Window

Second, the size of the requests is reduced to decrease the computational burden and allow easier investigation. This approach is expected to ensure that more significant disruptions are exposed to the RL agent. In addition, for the training, the occurrence rate and range of duration of each disruption profile are increased to expose the RL agent to more disruptions, ensuring more efficient training as presented in Table 6.2.

6.4.4. Model Re-Training

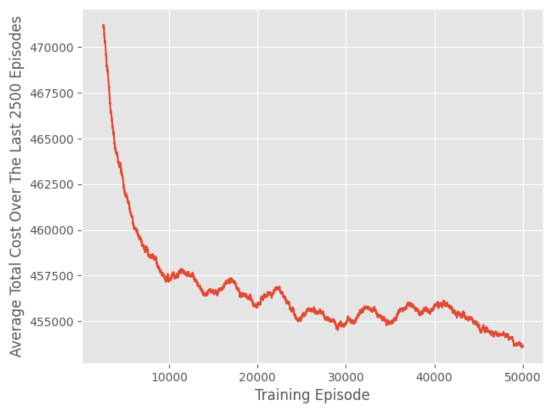
After the disruption algorithm is modified to expose the RL agent to more impactful disruptions, the RL agent is run through episodic training again. This time the number of instances used in the training

Table 6.2: Modified Service Disruption Profile

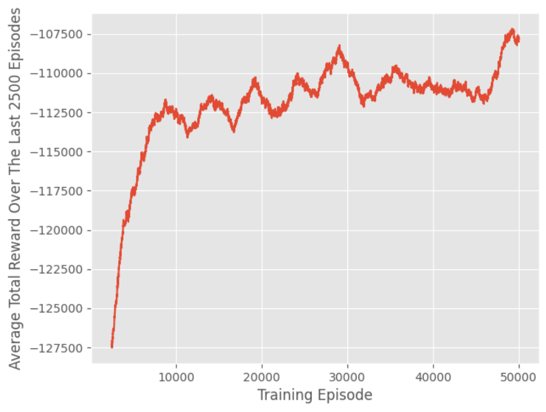
Profile	Description	Mode/ Location	Effect in the Simulation	Duration	Capacity Reduction	Occurrence Rate
1	Operational delays, road congestions	Train, Truck	Delay	3-6h	0%	0.00063
2	Operational delays, canal congestion	Barge	Delay	3-9h	0%	0.00072
3	Bad weather, labor strike, accident, systems maintenance	Train, Barge	Delay	12-48h	0%	0.00012
4	Terminal congestion, operational delays	Terminal	Delay	3-6h	0%	0.00063
5	High and low water level	Barge	Carrying capacity reduction	12-24h	15-20%	0.00012

is only 200 shipments announced for 3 week period. The simulation horizon is reduced to 5 weeks with the smaller service schedule in the input. The model is trained for 50,000 Episodes, and for every 1,000 episodes, the action value function in the form of Q-table is extracted to be used in the result analysis.

Figure 6.8 presents the result of the training over 50,000 episodes with the rolling mean applied using a 2,500-episode window. The trend shows that the cost and the reward start converging by starting to show a flat trend. After the model completes around 10,000 training episodes, both the improvement in mean total costs and rewards are not as significant as in the early training stage. Nonetheless, the training continues to expose the RL agent with more experiences, thus, resulting in a resilient model to react to various disruption scenarios.



(a) Total Cost - Episodic Training



(b) Total Reward - Episodic Training

Figure 6.8: Episodic Training for 50000 Episodes

6.5. Results and Numerical Experiment

6.5.1. Default Case Result

After the episodic training is completed, the RL agent performance is compared with the benchmark policies (*always wait* and *always reassign*). The performance comparison is performed by simulating each policy through multiple episodes with randomized disruption. Instead of sampling one case, the evaluation through multiple episodes can show how each policy responds to different cases of scenarios, thus, providing more insightful information about the resilience of the policy. In this experiment, the *greedy policy* (learning-assisted) is simulated seven times using different Q-table extracted from different amounts of training episodes ranging from 1,000 to 50,000 training episodes. The disruption profile set used in this evaluation is the same as the ones used in the training. Each policy is simulated 20 times and plotted using a box plot presented in Figure 6.9.

Figure 6.9 shows the total costs of each policy. GP{NUMBER} indicates the *greedy policy* with the

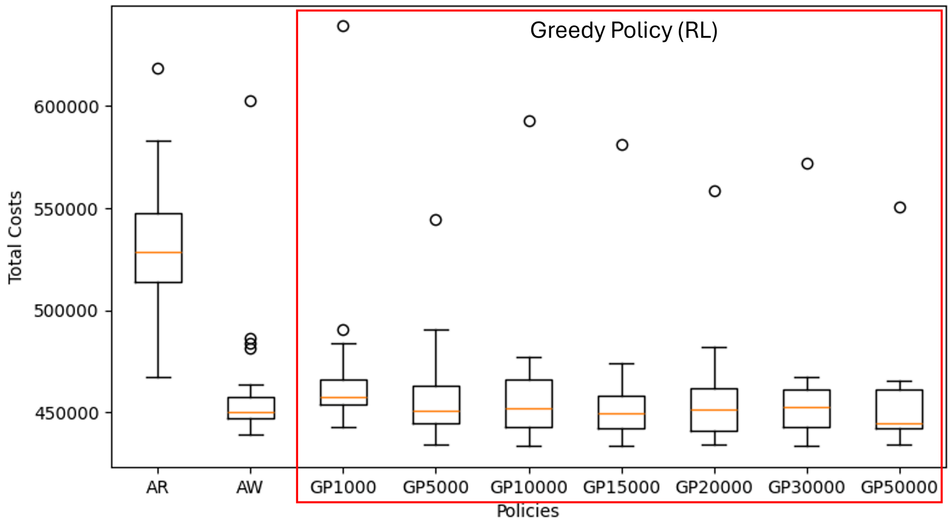


Figure 6.9: Policy Comparison (Multiple Cases)

number of training when the Q-table is extracted. On one hand, *always wait* policy significantly outperforms *always reassign* policy with the given disruption profile set. However, it shows some outliers with high costs in 4 cases indicating unstable performance. On the other hand, the *greedy policy* shows improvement along with the number of training and outperforms the *always wait* policy in some cases. In the beginning, the *greedy policy* shows unstable performance indicated by the high variance of total costs. The performance is getting stable as the training episodes increase. Additionally, the total cost average of the *greedy policy* keeps decreasing along with more completed training episodes. At 50,000 training episodes, the *greedy policy* manages to have a lower mean of total costs compared to the *always wait* policy.

Although showing the improvement relative to the benchmark policy, the chart does not clearly show how the learning-assisted model performs. To see this, the investigation is taken into more detail by comparing each episode. In this evaluation, the *always reassign* policy is excluded since it was clearly outperformed in Figure 6.9. With the same disruption profile set, the total cost in each episode is compared between *greedy policy* using the Q-table after 50,000 episodes and *always wait* policy. The result is presented in Figure 6.10

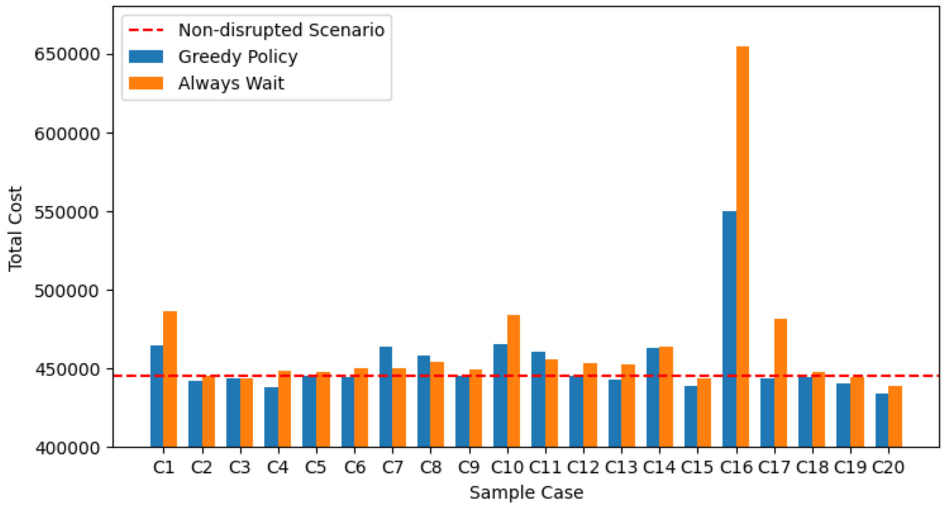
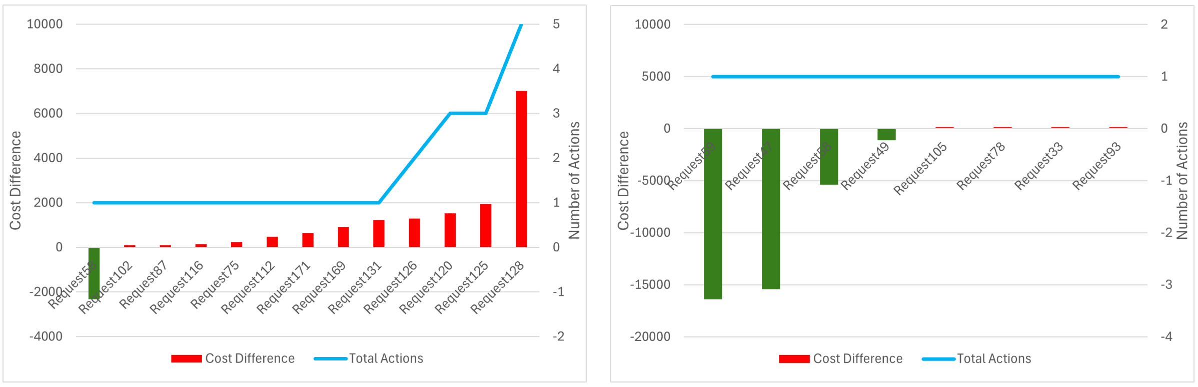


Figure 6.10: Comparison Between Always Wait and Greedy Policy in Episodic Simulations

The blue bars and orange bars represent the total cost of each policy respectively. The lower cost indicates a better performance of one policy over the other. The red dashed line is the total cost of a non-disrupted scenario. It is interesting that in a few cases, such as case C20, the total cost of a disrupted scenario appears lower than the non-disrupted. This happens because of several unique cases. For instance, in case C20, one request is planned in the first week to take service in the week after because its release time is slightly more than the departure time of the service mode in the first week. However, because of the delay in the departure time of the service, the shipment can be loaded to the assigned service in the first week, thus, resulting in lower storage costs.

The chart reveals that in only 4 out of 20 simulations, the *always wait* policy incurs lower total costs compared to the *greedy policy*. Among these 4 simulations, C3 exhibits an insignificant difference. This suggests that with the same disruption profile used for training, the *greedy policy* generally makes better decisions in response to disruptions with 80% of cases the *greedy policy* yields better performance. In these 20 sample cases, the cost savings by the *greedy policy* range from 0.08% to 15.96%. If the calculation only considers the affected requests, the highest savings from *greedy policy* over benchmark policy reaches 35,7%. The amount of savings depends on the severity of delays caused by the disruptions in the network. However, a closer examination at the action level shows instances where the *greedy policy* makes worse decisions than the *always wait* policy.



(a) Case C7 **(b)** Case C17

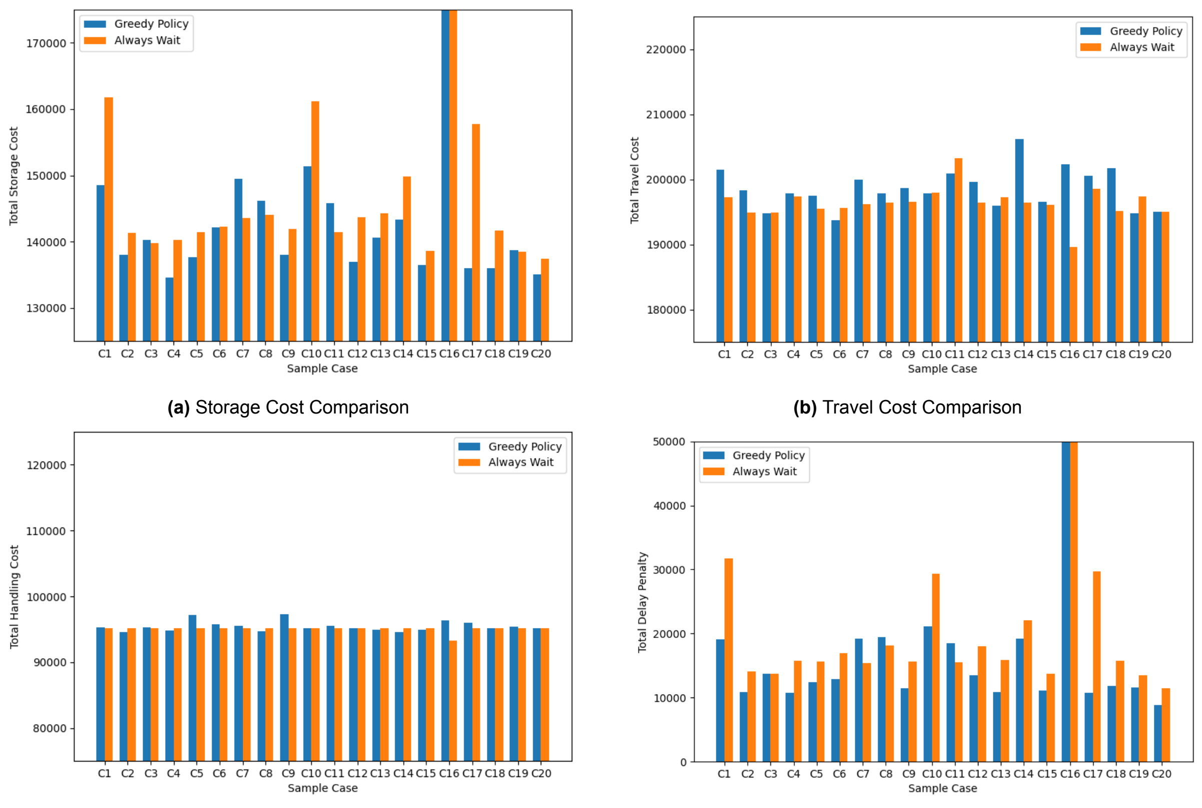
Figure 6.11: Cost Comparison per Shipment

Further investigation is conducted into case 7 (C7), where the *greedy policy* results in higher total costs. Each request with different total costs is compared between the two policies, with the results presented in Figure 6.11a. The chart shows the cost difference between shipments delivered by the *greedy policy* and the *always wait* policy. Negative values (green bars) indicate that the *greedy policy* generates lower costs, while positive values (red bars) indicate higher total costs. The solid blue lines, with reference to the right side y-axes, indicate the number of actions taken for each shipment. The chart highlights that the *greedy policy* yields a significantly higher cost for request 128. This correlates with the number of actions taken for these requests.

The Q-learning technique updates each action value in a given state by considering the reward and the discounted future state, as explained in Equation 4.17. This method captures stochastic behavior by updating the action value function after visiting a certain state. If a state occurs frequently, the action value will continuously update until it converges. Conversely, if a state is rarely visited, the action value function remains less updated according to the rewards from this state. In the case of Requests 128, the number of actions (blue line) reflects how often the shipments are affected by disruptions. These requests are disrupted multiple times, and such multiple disruptions, especially up to five in one shipment, are rare in the training set given the disruption profile. Consequently, the action value function suggests taking immediate actions without anticipating future disruptions, as the model has not frequently encountered such scenarios during training.

In contrast, the case C17 where the *greedy policy* outperforms the *always wait* policy, there is no case where the shipment is disrupted multiple times. Figure 6.11b presents the shipment cost difference for case C17 and the blue lines show that all disrupted shipments only take 1 action meaning they are only disrupted once, resulting in overall better performance of the *greedy policy*.

Another observation is made to see how the decision made by the RL agent relates to each cost component (Storage, Travel, Handling, and Delay). From the same case, each cost component is compared between 2 policies as presented in Figure 6.12. The y-axis scale is set to be the same with each tick representing 50,000 Euro to give easier comparison across different cost components. The scaling ignores case C16 since it is an extreme case with a significantly higher cost.



(a) Storage Cost Comparison

(b) Travel Cost Comparison

(c) Handling Cost Comparison

(d) Delay Penalty Comparison

Figure 6.12: Cost Element Comparison

The result shows that *greedy policy* tends to take the option with higher travel cost, indicated by Figure 6.12b, to reduce the storage cost and delay penalty as shown in Figure 6.12a and Figure 6.12d. Meanwhile, handling costs are not really affected by different policies. It is essential to understand that this tradeoff behavior is a result of the given cost parameters as it is directly linked to the reward system. The tradeoff behavior might change if the cost parameters used for the training are different

The current approach of the *greedy policy* involves selecting an itinerary from the solution pool for reassignment, and only triggers the optimization algorithm if there is no possible solution found in the solution pool. This method is compared to consistently triggering the optimization (*always optimize*) module during replanning. Figure 6.13 illustrates the comparison results. The cyan bars represent the total cost of always triggering the optimization module. The results indicate that the *greedy policy* using the solution pool generally performs better. This is because, in the *always optimize* case, the optimization module generates several solutions during replanning that are different from the solution pool given the real-time information and they are not explored during the training. Ultimately, this leads to suboptimal decision-making. However, in some cases such as case 14 (C14) and case (C10), the *always optimize* policy yields lower total costs. Further investigation reveals that the optimization model generates several solutions distinct from the solution pool that results in lower costs. This suggests that the solutions provided by the optimization module during replanning could potentially yield better performance if selected correctly.

6.5.2. Experiment on Different Disruption Sets

The second experiment is to evaluate the model that has completed 50,000 training episodes with different disruption scenarios and compare the performance with the benchmark policies. To perform

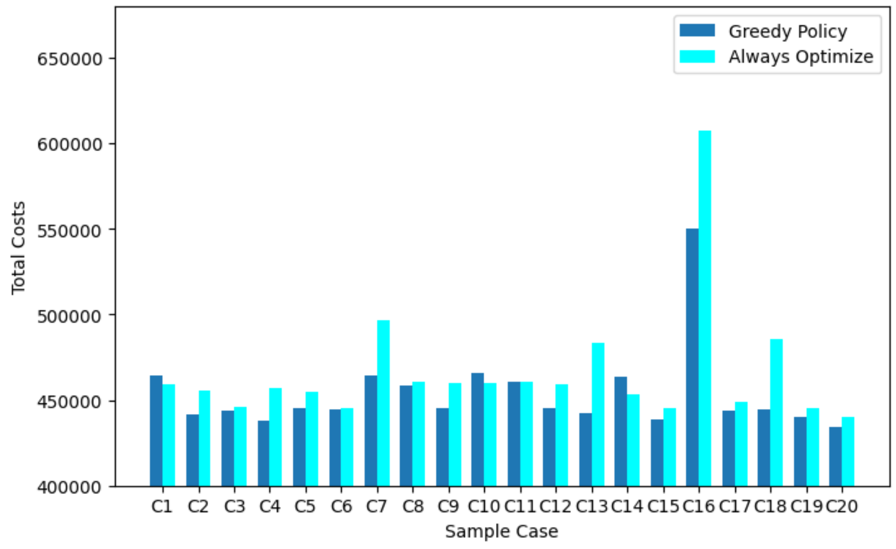


Figure 6.13: Comparison Between Solution Pool Approach and Always Optimize in Episodic Simulations

the experiment and deduce the change of behavior across different types of disruption, sets of disruption profiles are established and presented in Table 6.3. The disruption on trucks is omitted from profile1 due to technical reasons while it also hardly affects the network. The durations of the low severity profile (Profile1, Profile2, and Profile4) are the same as the disruption set used during the training. The disruption sets are characterized by different occurrence rates. S1 is scenario 1 with the lowest occurrence rate with the same values with the disruption set before modified as presented in Chapter 4. The occurrence rate keeps increasing two times for each disruption set until the set S5 with an extremely frequent occurrence rate. To put it in context, the occurrence rate of the default disruption set used in the training is between S2 and S3. The result of the second experiment is presented in Figure 6.14

Table 6.3: Disruption Scenarios

Profile	Severity	Location	Impact Type	LB Duration	UB Duration	Occurrence Rate (Lambda)				
						S1	S2	S3	S4	S5
Profile1	Low	Train	Delay	3	6	0.00021	0.00042	0.00084	0.00168	0.00336
Profile2	Low	Barge	Delay	3	9	0.00024	0.00048	0.00096	0.00192	0.00384
Profile3	High	Train, Barge	Delay	12	48	0.00004	0.00008	0.00016	0.00032	0.00064
Profile4	Low	Terminal	Delay	3	6	0.00021	0.00042	0.00084	0.00168	0.00336
Profile5	Low	Barge	Capacity reduction	12	24	0.00004	0.00008	0.00016	0.00032	0.00064

The experimental setup is nearly the same as the first experiment. The model is simulated 20 times for each disruption set and policy. The total cost of each simulation is shown in Figure 6.14. The *greedy policy* is plotted in blue colors while the *always wait* policy is in orange colors. The first notable finding is that changes in the disruption occurrence rate affect the total costs, especially in S4 and S5. As disruption frequency increases, so do the total costs, as indicated by the ascending trendline.

An interesting observation is that the learning-assisted model performs best under disruption scenarios S2 and S3. However, its performance declines compared to the benchmark policy as disruption frequency increases. Although the *greedy policy* shows more stable performance, indicated by fewer outliers, many cases yield higher costs compared to the *always wait* policy as the disruption frequency increases. This decline is due to differences between the probability distributions in this experiment and those used during episodic training. S2 and S3 are most similar to the distributions used in training, allowing the *greedy policy* to perform well in these cases.

In the less frequent disruptions, the *greedy policy* has almost the same performance as the benchmark policy. This is because the fewer disruptions in the network, the closer the *always wait* policy with the

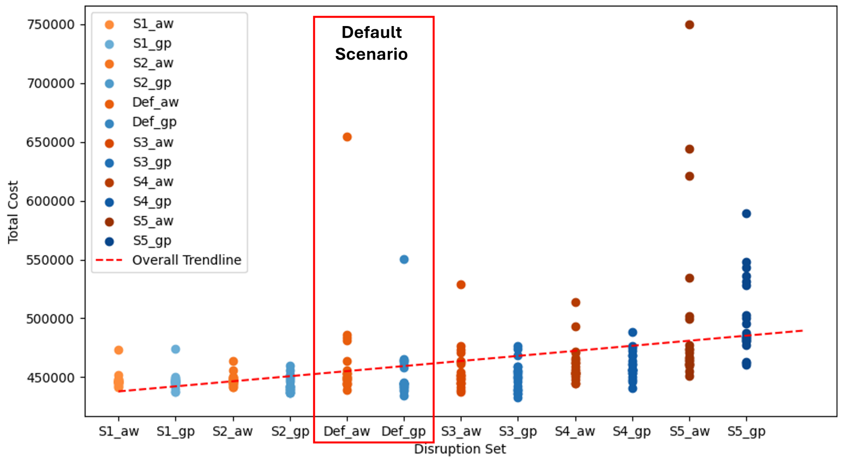


Figure 6.14: Total Costs for Different Disruption Sets

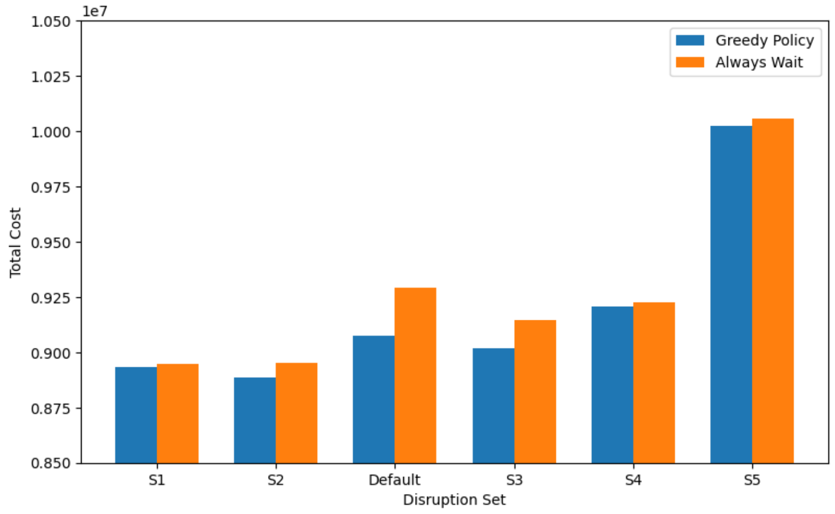


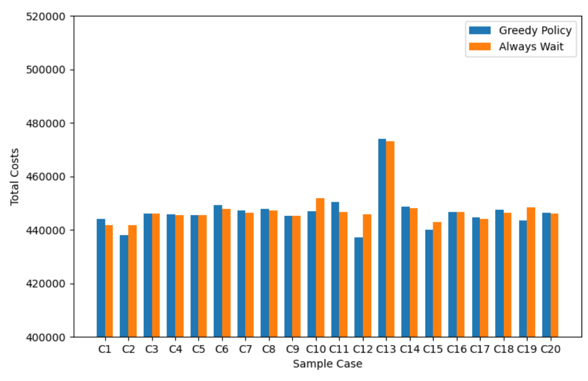
Figure 6.15: Aggregated Costs Comparison

optimal policy. This is also indicated by the fact that only 64 times the RL agent is triggered in total throughout 20 simulations. In contrast, the RL agent is triggered 219 times in the default disruption set.

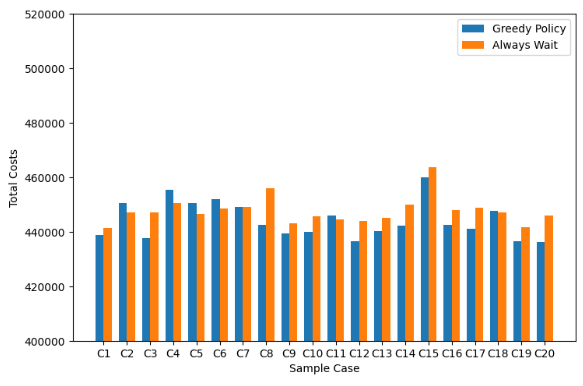
This finding is also indicated by the aggregated costs presented in Figure 6.15. The figure presents the total costs throughout the 20 simulations for each disruption set. It provides the overall performance of the policy. The chart shows that across all disruption sets, the total costs generated by the *greedy policy* are lower than the *always wait* policy. In particular, the cost gap in S2 and S3 is larger compared to the other disruption sets.

To see the comparison more clearly, the cost comparison in each disruption set is investigated. The result is presented in Figure 6.16. The 5 charts indicate that as the disruption frequency increases, the performance gets more unstable. The observation indicates that even though the *greedy policy* shows an overall poorer performance in disruption sets with more than 2 times the occurrence rate of the default scenario (S4 and S5), it can still avoid the cost getting extremely high. Meanwhile, there are a few cases where the *always wait* policy yields a significantly high cost such as C1 in disruption set S3 and C8, C10, and C17 in disruption set S5.

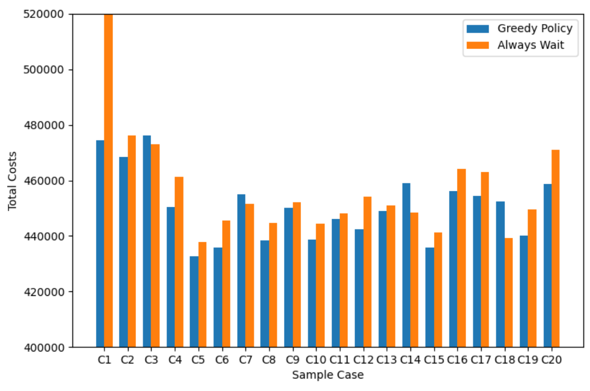
Despite overall better performance in the aggregated total costs, the data demonstrates that the *greedy policy* outperforms the benchmark policy only in scenarios S2 and S3 based on the number of simulations where it achieves better results. Specifically, Figure 6.17 summarizes the performance across various disruption sets. As explained earlier, in disruption set S1, the *greedy policy* is outperformed by the *always wait* because in the lower disruption frequency, the *always wait* policy is getting closer to the optimal solutions. A thorough investigation reveals there are several unstable values, again due to the stochastic environment.



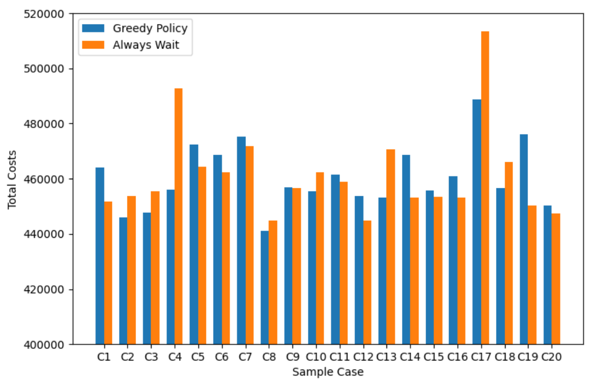
(a) Disruption (S1)



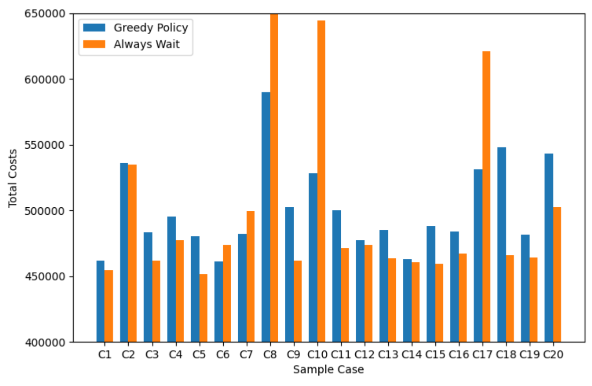
(b) Disruption (S2)



(c) Disruption (S3)



(d) Disruption (S4)



(e) Disruption (S5)

Figure 6.16: Sensitivity Analysis with Different Disruption Occurrence Probability

One sample is investigated in case 1 (C1) of disruption set S1. It is found that one request takes a reassign action from Barge-Train to Barge-Barge-Train. A detailed comparison shows that before taking the action, the value function indicates that reassign has a better value than wait. However, after taking the reassign action and updating the value function, the updated value of the reassign action becomes worse compared to wait. This could be caused by multiple disruptions occurring after the wait action during training, which alters the value function.

In disruption sets S2 and S3, the *greedy policy* performs better in 13 and 16 out of 20 simulations, respectively. This indicates that the *greedy policy* is more effective approximately 65% to 80% of the

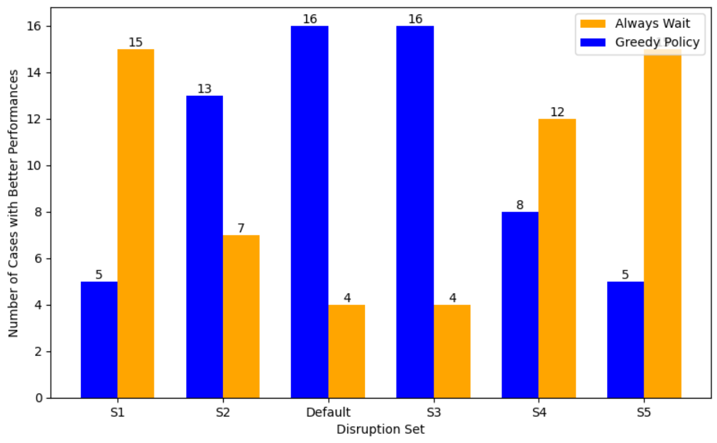
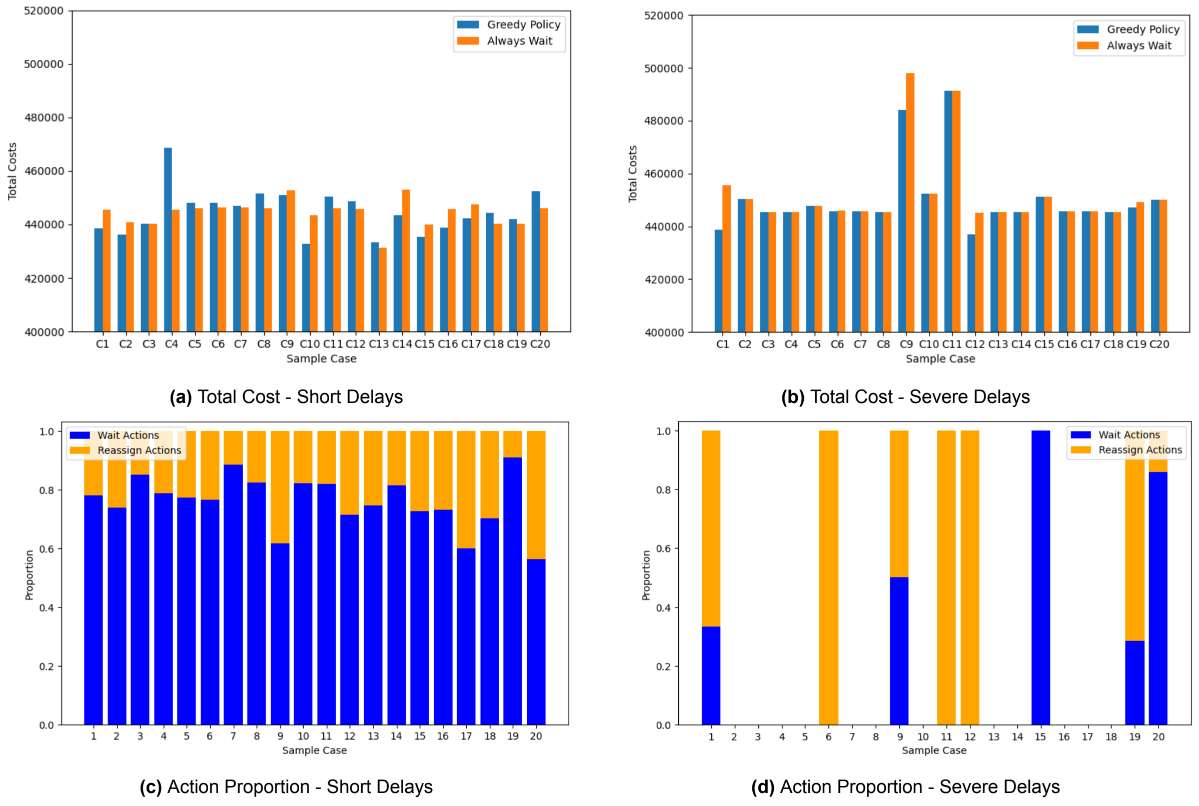


Figure 6.17: Performance Comparison by Number of Cases

time when the probability distribution closely resembles the episodic training. Meanwhile, in disruption sets S4 and S5, the *greedy policy* performance is declining due to many disruptions in the network creating more multiple disruptions for one shipment.

The next experiment on the disruption set is to simulate the model through 2 distinct scenarios: Short delay disruptions and severe delay disruptions. For the short delay, the disruption profile4 is omitted from the disruption set, while the severe delay only includes the disruption profile4. The occurrence rate used in this experiment is the same as the default case in the episodic training. However, to apply enough disruption, the probability occurrence for severe disruption set is slightly increased to the same level as in disruption set S3 from the previous experiment to apply more disruption but still keeping the distribution closer to the one used in the training. The result of this experiment is presented in Figure 6.18



(a) Total Cost - Short Delays

(b) Total Cost - Severe Delays

(c) Action Proportion - Short Delays

(d) Action Proportion - Severe Delays

Figure 6.18: Short and Sever Delays Experiment

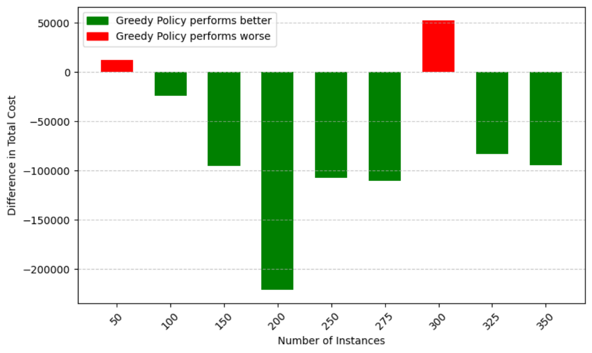
The two charts on the left represent the result of applying short delays while the charts on the right show the result of applying severe delays. Figure 6.18a shows that the disruptions create many delays indicated by only one case with the same total cost (C3). This is possible due to the high disruption occurrence rate. However, the low severity of the disruption makes the optimal solution closer to the *always wait* policy. The poor performance of the RL in some cases, again, is caused by multiple disruptions in one shipment. An important point to highlight is, due to the low severity disruption, the highest savings by the RL-agent are only 2.4%.

In contrast, in the severe delays scenario, the costs between *greedy policy* and *always wait* in most cases are almost the same due to the low occurrence rate as presented by Figure 6.18b. However, once the disruption creates delays in the network, the RL-agent can produce higher savings compared to the short-delayed disruption. The savings in the severe delay scenario reaches up to 3.65%.

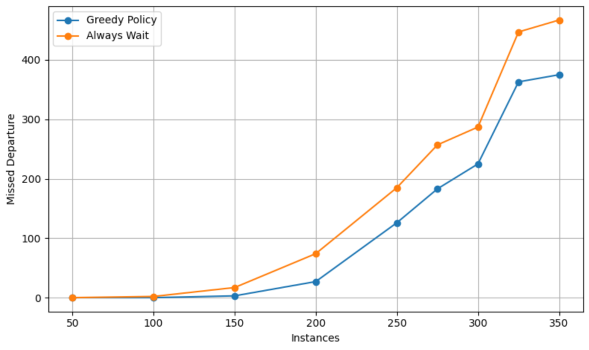
The charts at the bottom (Figure 6.18d and Figure 6.18c) present the proportion of actions between wait and reassign taken by the RL agent. Some cases in the severe delays scenario are blank because there was no action taken by the RL agent in that specific case. As expected, in the severe delays, despite only a few actions taken due to low disruption frequency, the proportion of taking reassign action is generally higher than wait action.

6.5.3. Experiment on Different Demand Instances

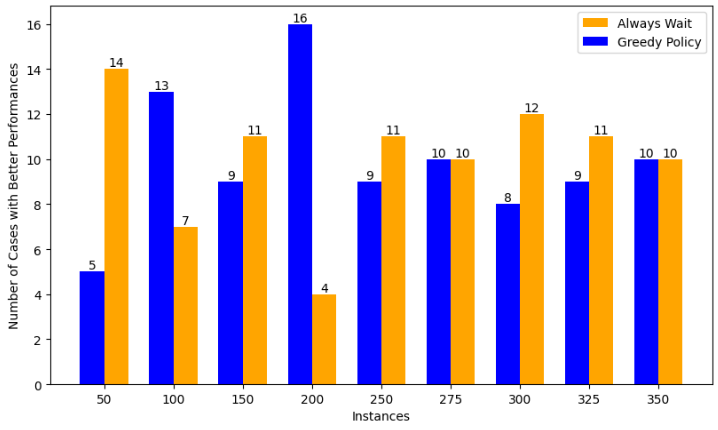
The third experiment simulates the model using different demand instances, with nine sets ranging from 50 to 350 requests. Each set employs the default disruption profiles and is repeated 20 times to ensure robustness. The aim is to analyze the model's sensitivity to varying numbers of requests by aggregating and comparing the total costs from the 20 simulations for each instance.



(a) Total Costs Differences



(b) Total Missing Departures



(c) Performance Comparison by Number of Cases

Figure 6.19: Sensitivity Analysis with Different Demand Instances

Figure 6.19a illustrates the differences in aggregated total costs between the two policies. Green bars indicate that the *greedy policy* incurs lower costs, while red bars indicate the opposite. The results show that the *greedy policy* outperforms the *always wait* policy in most cases as demand increases. This demonstrates the *greedy policy's* superior ability to handle a larger number of instances, as indicated by the number of missed services shown in Figure 6.19b. Missed departures, a result of capacity constraints, are fewer under the *greedy policy*, indicating more accurate reassignment decisions.

The poorer performance at lower demand levels can be attributed to the *always wait* policy being closer to the optimal solution due to fewer affected requests during the simulation. Additionally, any suboptimal decisions made by the reinforcement learning algorithm are more impactful on total cost when there are fewer disruptions to offset with better actions.

Figure 6.19c shows the number of simulations where each policy performs better across different demand instances. From 50 to 200 instances, the *greedy policy* gains performance as demand increases. However, beyond 200 instances, the trend does not provide a conclusive comparison, with both policies exhibiting similar performance levels. The suboptimal performance of the *greedy policy* beyond 200 instances might be due to the lack of demand variance during training. Additionally, larger instances make the network more sensitive to missed departures. In many cases, the value of a particular action becomes inaccurate because, during training, a reassign action in a certain disruption might generate a lower cost. However, in the experiment with larger instances, the same reassign action might incur a much higher cost due to missed departures caused by insufficient service line capacity.

6.5.4. Experiment on Different Cost Parameters

The final experiment investigates the model's sensitivity to changes in storage cost parameters. The storage cost was selected for modification due to its significant impact on the cost tradeoff between the *greedy policy* and the *always wait* policy, as discussed in Section 6.5.1. The default case is used as the reference where the storage cost is 1. The storage cost for the experiment is set to start at 0.2 and incrementally increases by 0.2 until it reaches 2. This experiment aims to assess the sensitivity of the greedy policy's performance across varying cost parameters.

Figure 6.20 illustrates the performance comparison of the two policies under these different storage cost parameters. The findings indicate a significant drop in the greedy policy's performance as the storage cost is lower than the default case. This decline occurs because the model tends to select itineraries with higher travel costs to avoid increased storage costs due to delays according to the cost parameters used in the training. With lower storage costs, shipments become less sensitive to delays, thus, the RL agents' value function is no longer relevant. In contrast, the greedy policy maintains its performance as the storage cost increases because the cost tradeoff is still relevant. However, the *greedy policy* performs best under the default cost scenario. This is because the reward system is determined by the cost parameters used during training. Deviations from these costs impact decision-making accuracy, even if the changes favor the tradeoff.

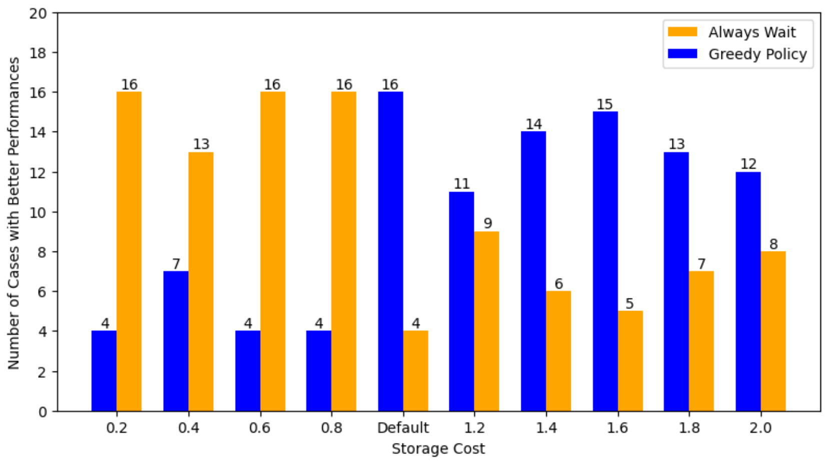


Figure 6.20: Sensitivity Analysis with Different Cost Parameters

7 Discussion

The work of this research is divided into three distinct processes comprising, constructing the freight simulation environment, integrating a plug-in matching algorithm, and developing the reinforcement learning agent. Even though the centerpiece of this project is the implementation of the RL, the two other modules play as much important role as all the information, input, and output of each module, flows among them. One small error in one module could cause a collapse in the whole model. The discussion of this research is structured according to those three modules.

7.1. Limitations

7.1.1. Simulation Module

The simulation module aims to create a dynamic environment for hinterland freight transport, capturing disruptions and simulating network delays resulting from these disruptions. By modeling the interactions among the three components—shipment, service, and disruption—delays are expected to emerge organically. More severe network delays provide greater opportunities for the RL agent to improve performance. However, the simulation must also maintain realism, ensuring the network is not excessively disrupted to create delays artificially. Striking this balance between realism and providing sufficient improvement opportunities for the RL agent is crucial to this research.

One limitation of the simulation module that influences the results is its scope and assumptions. The scope is limited to inbound cargo from the main port to the inland terminal, leading to a technical assumption that service modes return directly to their origin without considering the return trip. Including cycle-based operations could capture disruptions during the return trip, potentially causing delays in picking up shipments at the origin terminal. Additionally, the assumption that one service mode only serves one origin-destination pair limits shipment choices to either staying in the current mode or transferring to another mode or line at the next stop, which could create a more realistic yet complex problem.

The disruption profiles are formulated based on available literature and published datasets. The quality of these profiles depends on the amount of accessible data. Common disruptions, such as bad weather at the terminal, are not included due to limited information, yet these profiles could significantly impact the RL agent's outcomes. Creating disruption profiles based on historical data would enhance the RL agent's reliability. Another important disruption not included in the model is mode-specific infrastructure disruptions, such as railway disruptions affecting multiple train lines with the same origin and destination.

7.1.2. Optimization Module

The optimization module provides shipment planning both offline (regular planning) and online during disruptions. It features a plug-and-play capability for different optimization techniques or objectives, provided they meet the required input and output criteria. Different optimization algorithms yield varying results. For instance, the heuristic algorithm discussed in Chapter 5 results in the *always reassign* strategy outperforming the *always wait* strategy, significantly improving when applying the RL-assisted model. This is because the heuristic algorithm's initial solution does not aim to minimize total cost, allowing the reassign option to excel, and the RL model to select the optimal choices between the two. In the case study, the plug-in model aims to minimize costs. Therefore, the *always wait* policy already provides the best solution in a non-disrupted environment, explaining the more significant improvements observed during verification compared to the case study.

A limitation of the optimization algorithm is its runtime. Ideally, the simulation triggers the optimization module whenever required. However, to achieve convergence in training the RL agent, modifications are necessary, as detailed in Chapter 6. This affects the overall performance of the learning-assisted model because the solutions provided by the optimization module during replanning, depending on the circumstances, may differ from those in the solution pool used in the episodic training.

While the current model manages to yield better performance relying on the solution pool approach, there is a potential to improve it further by generating optimal solutions during the training, ultimately training the RL with higher action choice quality.

7.1.3. Reinforcement Learning

The central brain of the RL agent is the value function. It consists of the value map for each action taken in a state. The first limitation of this research in the RL part is that we use a tabular method, Q-Learning, for updating the value function during the training. While it manages to reach convergence and ultimately provide findings of implementing a learning agent in the whole model, it creates several limitations in this research.

The tabular method requires the learning agent to visit each state frequently to reach convergence. This caused the RL to require a huge amount of training episodes. With the limited time resources and the run-time constraint from the plug-in model. There are still states where the RL has not visited often enough, partly because it is a rare case, resulting in suboptimal decision-making in a few cases.

This also restricts the formulation of the states considered by the RL, increasing the number of features in the state will increase the exploration space during the training and increase the required amount of training. The formulation of the states is also an important part of making the RL work. The challenge is to find the sweet spot between keeping a low number of features to minimize the amount of episodic training and having many features to allow the RL to have more knowledge of different states and distinct the optimal action for the given state. The proposed model in this thesis utilizes six features of states including request's current position, the request's destination, the request's, due time, the request's volume, the type of disruption, and the current time. The possible way to increase the feature for this model could be the number and total volume of all affected requests at the given time. In the high number of disruptions, the RL agent might choose to wait to avoid capacity problems in the given reassign option.

Another interesting discussion point is the timing of decision-making. The proposed models trigger online planning and take immediate action whenever a disruption occurs. However, in some cases, delaying the decision-making process until a more decisive moment could prevent unnecessary suboptimal decisions. For example, consider a disruption in a service line affecting a shipment, request1, which is scheduled to take Barge1 while waiting at the terminal. The optimization module might suggest switching to Train1. The decision to wait or reassign could be deferred until just before one of these options is ready for loading. This approach accounts for the possibility that the disruption might resolve before the original itinerary's loading time, ultimately resulting in no delays.

7.1.4. Data Deficiency

Another important aspect of this research is the data used for training the model. The service network data in the case study is based on a real-world network derived from the literature. However, the demand is generated synthetically. Efforts have been made to balance transport supply and demand by estimating the capacity proportion of each origin-destination pair and using that proportion as a basis to generate demand. Despite these efforts, several unutilized services were still found in the network. This results in fewer affected requests because some disruptions occur in these unutilized service modes.

Cost parameters used in the training are also derived from the literature. It is acknowledged that these cost parameters do not reflect real-world costs accurately. However, the costs are the main driver of the RL agent's learning since the reward system is based on the shipment cost as a consequence of taking action.

7.2. Result Interpretation and Implication

7.2.1. Result Interpretation

The objective of the proposed model is not solely to achieve the highest cost savings. While significantly lower costs resulting from the actions taken indicate excellent performance, the amount of cost savings also depends on the delays in the network. Given the dynamic and complex nature of these delays, it is more important for the RL model to outperform the naive benchmark policy across various disruption scenarios. This aligns with the primary objective of the research: to create a resilient synchromodal framework.

Resilient freight transport is measured by a system's ability to recover from disruptions (Chen and Miller-Hooks, 2012). In this model, the key performance indicator (KPI) used to measure disruption impact is the total cost. Achieving lower costs compared to the benchmark in specific cases indicates that the framework responds more effectively to disruptions, restoring performance to its original state. Moreover, demonstrating better performance across different scenarios, as shown by the proposed model in the case study, highlights the model's ability to recover from disruptions, thus enhancing its resilience.

Given the limitations elaborated in the previous section, the proposed model evidently provides added values as a decision support system in response to disruptions. 80% better performance in 20 test cases in the default scenario provides a strong indication that the learning-assisted model manages to respond well to disruptions. Due to the stochastic environment, it is indeed possible for the RL agent to make suboptimal decisions in some cases.

It is important to address the reason behind the poor performance in several cases. From the result analysis, the primary issue is that the RL agent often selects suboptimal actions in rarely visited states. Given the disruption probability during training, scenarios where a shipment is disrupted multiple times are infrequent. When an event occurs infrequently, the value function does not get updated accordingly.

For instance, in a certain state, the value of taking a wait action might be -1000, while the value of a reassign action might be -500. Suppose during training, the RL agent encounters another disruption in the selected service line. In this new state, both action values are very low, such as -3000 and -3500. In this situation, the RL agent updates the value function of the initial reassign action, previously -500, to a lower value. However, if the first state is visited again and no second disruption occurs after selecting reassign, the value function will be updated again, gradually aligning with the most probable scenarios.

One way to address this could be by determining a feature to distinguish the infrequent events from the frequent ones. For instance, if the second disruption is a port congestion, the port congestion usually happens during a high vessel traffic. Providing a 'current vessel traffic' as a feature could help the RL agent distinguish the first state in the above case, in a way predicting the disruption. This indeed requires a huge and reliable dataset.

Another important point to highlight is the ability of the RL agent to maintain performance across different environments. RL can uncover hidden distributions in a given dataset through training, enabling it to operate effectively in stochastic environments. However, if the RL agent encounters a problem with a different distribution, its performance could be affected. Therefore, exposing the RL agent to excessive disruptions during training does not necessarily lead to better implementation performance. Instead, it is more effective to train the agent with disruptions or datasets that closely resemble real-world scenarios. Furthermore, it is crucial for a model to avoid overfitting and to be effective beyond a single specific case. This research demonstrates that the RL agent in the proposed model consistently outperforms the benchmark policy, even when disruption occurrence rates vary by up to 30%.

Finally, according to the result analysis, the concept of integrating a learning-agent into a decision support system could work. Referring to the previous discussion, the proposed model is trained using a limited dataset and several assumptions. On top of strengthening the algorithm using a more sophisticated approach, the provision of reliable datasets including the network, disruptions, and costs will be the key determinant of the learning agent's performance.

7.2.2. Implication to the real-world

Disruptions are common in freight transportation, ranging from minor delays due to congestion to extraordinary events like the Suez Canal blockage. This research proposes a model centered on an operational-level decision support system, serving as a foundation for further development in line with technological advancements in the port-logistics industry.

The synchromodal concept underpins this model, allowing shipments to be re-routed flexibly using real-time information. Although not yet fully implemented industry-wide due to various success factors, operators like Hutchinson Ports Europe Intermodal (HPEI) have begun adopting this concept, indicating its emerging significance in the industry. As technological advancements align with the requirements of synchromodality, integrating RL within the synchromodal framework could provide valuable insights for future implementation once all enablers are in place.

Even with the current model, if the RL is properly trained, a synchromodal planner could use it to assist in decision-making during disruptions. A simple implementation could involve developing software that incorporates the trained agent. When a disruption occurs, the planner identifies the affected containers and inputs this information, along with all relevant states at the time, into the software. The software then suggests actions for each shipment. While not definitive, this tool can offer valuable technical insights to planners, complementing their broader judgment.

Giusti (2019) suggests digital twins as a key enabling technology for synchromodality. In a more advanced scenario, the proposed model could be integrated with a digital twin system that replicates all operations in real-time. In the event of a disruption, the system would automatically identify the affected shipments and their real-time states, passing this information to the learning-assisted model, as demonstrated in this thesis. This integration would eliminate the need for manual information transfer by the planner, streamlining the decision-making process.

8 Conclusions and Possible Future Research

8.1. Conclusions

The objective of this research is to propose a learning-assisted model to address the unknown duration of disruption in port-inland freight transportation. The research should uncover the answer to the main research question *”To what extent does a learning approach improve the resilience of a synchromodal framework in coping with disruptions?”*. The answers to this question are dissected into several answers to the sub-questions as follows.

1. *”What are the types of disruptions in the synchromodal transport and what strategies are applied to react to them?”*
   Disruptions in synchromodal transport can be categorized according to frequency and severity and may require different reactions from strategic, tactical, to operational levels. At the operational level, the types of disruptions are mentioned in numerous studies including bad weather, water level fluctuation, labor strikes, accidents, or congestion. These various types, however, could be grouped into two distinct spectrum: Low occurrence probability with severe impact, and high occurrence with low impact. The disruptions can have an impact either on the service or demand side. On the service side, the impact could be a delay or reduction in carrying capacity, while on the demand side, the impact could be the change in demand requirements such as volume and release time. In response to the disruption, a couple of reaction strategies can be implemented including waiting, transshipment, service re-routing, or shipment reallocation.
2. *”How can a synchromodal framework under disruption be adequately modeled?”*
   To model disruptions in a synchromodal framework, a discrete event simulation is developed to capture the dynamic behavior of these disruptions. The simulation consists of three main components: mode service, shipment, and disruptions. The interaction between mode service and shipment simulates port-inland freight operations. Five service disruption profiles and three request disruption profiles are created to apply disruptions within the freight network. Each disruption profile is characterized by its type of impact, severity, occurrence probability, and potential location. The simulation module generates disruptions based on these profiles, following a specific probability distribution. Disruptions in the service network can cause delays by halting operations at disrupted locations (terminals or service lines) for a randomly determined duration. This simulation forms the basis for understanding the behavior of a synchromodal framework under disruption in this research.
3. *”How can reaction strategies under synchromodal framework be modeled in response to the disruptions?”*
   The reaction strategies developed in this research focus on operational-level decisions. Given the difficulty of altering fixed schedule services, container reallocation or re-routing is employed to respond to disruptions by assigning shipments to different service lines. The proposed model

includes an optimization module that creates shipment plans consisting of container itineraries. This optimization module acts as a centralized planning tool, facilitating both offline and online planning. Offline planning is executed according to a fixed time interval, representing a regular shipment planning. Online planning is triggered only in response to disruptions, targeting affected requests specifically. Additionally, the optimization module is designed to integrate seamlessly with various optimization techniques, making the model scalable for future enhancements.

4. *"How can a learning approach be incorporated into the synchromodal framework to provide a better solution space?"*
   After executing the online planning, shipments will have the option to take different service lines to avoid disruptions. An RL agent is developed to aid in the decision-making process, determining whether to reassign the shipment based on the online plan or to stick with the original offline itinerary. An RL sub-agent is attached to each affected shipment to monitor the sequential decision-making process and collect rewards, which are used to update a centralized value function. Rewards are based on the actual costs incurred from the chosen actions, with the value function being updated using the Q-learning technique. By exposing the RL agent to various disruptions, it learns from experience and can provide improved reaction strategies over time.
5. *"How does the learning-based synchromodal framework perform under the disruptions?"*
   After completing 50,000 episodes of training, a case study using real service network data was conducted. The learning-assisted model was simulated 20 times using a greedy policy and compared with the benchmark policy of always waiting. The greedy policy outperformed the benchmark in 16 out of 20 simulations or 80% of the time, achieving cost savings of up to 15.96% of the total cost compared to the benchmark, or 35.7% when considering only the affected requests. While the extent of cost reduction depends on network delays, the model's ability to perform better across various disruption scenarios is more crucial than the specific amount of savings. Additionally, the model proves to still provide better performance compared to the benchmark policy in a certain range of different probability distributions of the disruption, as indicated by the result of up to 30% difference in occurrence rate. The overall performance suggests that the proposed model can create a more resilient synchromdoal framework by utilizing reinforcement learning.

8.2. Recommendations for Future Research

The results of this research indicate that utilizing a learning approach within a synchromodal framework could bring added value and serve as a foundation for future research directions. Related to the disruption scenarios, incorporating data-driven disruptions and additional profiles, such as severe disruptions at terminals or service line infrastructure, could provide more realistic scenarios for the RL agent and enhance the reliability of the model. Additionally, this thesis treats disruptions at the main port as a random variable related to shipment release time. Expanding the simulation to include port operations and mother vessel arrivals could capture greater complexity and broader disruptions, leading to a better training environment for the RL agent.

Regarding the optimization module, the current model integrates only a fragment of a sophisticated learning-based optimization model. Future research could explore integrating a comprehensive predictive optimization model while streamlining datasets for both learning approaches. Another potential direction involves integrating a meta-heuristic optimization model to address run-time issues, enabling the optimization module to be triggered consistently during training and fully exploring the model's potential.

For the learning agent, three distinct future research directions are proposed. First, if the scope of the network could be expanded to between Distribution Centers (DC), the action space could include cancellation orders if the RL predicts that the late delivery might no longer be relevant to the customer's requirements. Second, In the current model, each sub-agent attached to a shipment aims to minimize costs related to that shipment, and collectively, these sub-agents are expected to lower the total system cost. However, conflicts among multiple agents could arise due to capacity constraints. Implementing a communication scheme between multiple sub-agents to generate actions leading to system-wide optimization could be a valuable future direction. The third direction involves addressing the limitations of the current model by implementing deep reinforcement learning techniques. This approach allows the model to estimate values of unvisited states, enabling the RL agent to perform well without visiting all states. This would allow for more complex state features and more accurate decisions without significantly increasing training time.

By addressing the run-time and reducing the required episodic training, the model can be improved in several ways. The experiment of different hyperparameters can help to identify their optimal values. Additionally, experiments using different inputs for different episodic training could be performed. Together, hyperparameter tuning and diverse episodic training will indicate the best training strategy to maximize the model's potential. Furthermore, the experiment of different scenarios can be extended with more case samples to provide more representative performance results.

Bibliography

(2019). Barge Traffic Disruptions and Their Effects on Shipping Costs in Agricultural Freight Corridors Summary. Technical report, U.S. Department of Agriculture, Washington, DC.

Abadi, A. and Ioannou, P. (2014). Optimization strategies for resilient freight transport and sustainability. In *Proceedings of the IEEE Conference on Decision and Control*, volume 2015-February, pages 6472–6477. Institute of Electrical and Electronics Engineers Inc.

Acero, B., Saenz, M. J., and Luzzini, D. (2022). Introducing synchromodality: One missing link between transportation and supply chain management. *Journal of Supply Chain Management*, 58(1):51–64.

Alons, K., Fontys, H., Somers, G., and Van Duin, J. H. R. (2019). MOVING FROM INTERMODAL TO SYNCHROMODAL TRANSPORT: A MATURITY MODEL APPLIED TO A CASE STUDY IN NORTH-WESTERN EUROPE MOVING FROM INTERMODAL TO SYNCHROMODAL TRANSPORT: A 1 MATURITY MODEL APPLIED TO A CASE STUDY IN NORTHWESTERN EUROPE Research Centre Sustainable Port Cities/Faculty of Technology, Policy and Management, Moving @ 8 Rotterdam. Technical report.

Ambra, T., Caris, A., and Macharis, C. (2019). Should I stay or should I Go? Assessing intermodal and synchromodal resilience from a decentralized perspective. *Sustainability (Switzerland)*, 11(6).

Barkley, A. and Mcleod, K. (2022). Congestion and consolidation: An empirical study of a barge shipping merger. *Regional Science and Urban Economics*, 93.

Barua, L., Zou, B., and Zhou, Y. (2020). Machine learning for international freight transportation management: A comprehensive review. *Research in Transportation Business and Management*, 34.

CCNR (2020). WATER LEVELS AND AVAILABLE VESSELS' DRAUGHT AT GAUGING STATIONS ON RHINE AND DANUBE.

Chen, L. and Miller-Hooks, E. (2012). Resilience: An indicator of recovery capability in intermodal freight transport. *Transportation Science*, 46(1):109–123.

De Juncker, M. A. M., Huizing, D., del Vecchyo, M. R. O., Phillipson, F., and Sangers, A. (2017). Framework of Synchromodal Transportation Problems. pages 383–403.

Delbart, T., Molenbruch, Y., Braekers, K., and Caris, A. (2021). Uncertainty in intermodal and synchromodal transport: Review and future research directions.

Di Febbraro, A., Sacco, N., and Saeednia, M. (2016). An agent-based framework for cooperative planning of intermodal freight transport chains. *Transportation Research Part C: Emerging Technologies*, 64:72–85.

Durán-Micco, J., Jordehi, S. A., and Macharis, C. (2023). Evaluating synchromodal transport with agent-based simulation. In *Transportation Research Procedia*, volume 72, pages 619–626. Elsevier B.V.

Eurostat (2020). 77% of inland freight transported by road in 2020.

Filom, S., Amiri, A. M., and Razavi, S. (2022). Applications of machine learning methods in port operations – A systematic literature review. *Transportation Research Part E: Logistics and Transportation Review*, 161.

Filom, S. and Razavi, S. N. (2023). Decarbonization through modal shift using a synchromodal platform: A case study in the Great Lakes.

Gallardo, P., Murray, R., and Krumdieck, S. (2021). A sequential optimization-simulation approach for planning the transition to the low carbon freight system with case study in the North Island of New Zealand. *Energies*, 14(11).

Gao, S. and Liu, N. (2022). Improving the resilience of port–hinterland container logistics transportation systems: A bi-level programming approach. *Sustainability (Switzerland)*, 14(1).

Geng, J., Li, M.-W., Dong, Z.-H., and Liao, Y.-S. (2015). Port throughput forecasting by MARS-RSVR with chaotic simulated annealing particle swarm optimization algorithm. *Neurocomputing*, 147:239–250.

Giusti, R., Manerba, D., Bruno, G., and Tadei, R. (2019). Synchromodal logistics: An overview of critical success factors, enabling technologies, and open research issues. *Transportation Research Part E: Logistics and Transportation Review*, 129:92–110.

Goti, A., editor (2010). *Discrete Event Simulations*. Sciyo.

Guo, W., Atasoy, B., and Negenborn, R. R. (2022). Global synchromodal shipment matching problem with dynamic and stochastic travel times: a reinforcement learning approach. *Annals of Operations Research*.

Guo, W., Atasoy, B., van Blokland, W. B., and Negenborn, R. R. (2020). A dynamic shipment matching problem in hinterland synchromodal transportation. *Decision Support Systems*, 134.

Guo, W., Atasoy, B., van Blokland, W. B., and Negenborn, R. R. (2021). Global synchromodal transport with dynamic and stochastic shipment matching. *Transportation Research Part E: Logistics and Transportation Review*, 152.

Hrušovský, M., Demir, E., Jammernegg, W., and Van Woensel, T. (2021). Real-time disruption management approach for intermodal freight transportation. *Journal of Cleaner Production*, 280.

Jonkeren, O., Jourquin, B., and Rietveld, P. (2011). Modal-split effects of climate change: The effect of low water levels on the competitive position of inland waterway transport in the river Rhine area. *Transportation Research Part A: Policy and Practice*, 45(10):1007–1019.

Karam, A. and Reinau, K. H. (2022). A Real-Time Decision Support Approach for Managing Disruptions in Line-Haul Freight Transport Networks. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):24765–24777.

Kogler, C. and Rauch, P. (2018). Discrete event simulation of multimodal and unimodal transportation in the wood supply chain: A literature review. *Silva Fennica*, 52(4).

Kourounioti, I., Polydoropoulou, A., and Tsiklidis, C. (2016). Development of Models Predicting Dwell Time of Import Containers in Port Container Terminals – An Artificial Neural Networks Application. *Transportation Research Procedia*, 14:243–252.

Larsen, R. B., Atasoy, B., and Negenborn, R. R. (2021). Model predictive control for simultaneous planning of container and vehicle routes. *European Journal of Control*, 57:273–283.

Larsen, R. B., Guo, W., and Atasoy, B. (2023a). A real-time synchromodal framework with co-planning for routing of containers and vehicles. *Transportation Research Part C: Emerging Technologies*, 157.

Larsen, R. B., Negenborn, R. R., and Atasoy, B. (2023b). A learning-based co-planning method with truck and container routing for improved barge departure times. *Annals of Operations Research*.

Layeb, S. B., Jaoua, A., Jbira, A., and Makhlouf, Y. (2018). A simulation-optimization approach for scheduling in stochastic freight transportation. *Computers and Industrial Engineering*, 126:99–110.

Lyu, Z., Pons, D., and Zhang, Y. (2023). Emissions and Total Cost of Ownership for Diesel and Battery Electric Freight Pickup and Delivery Trucks in New Zealand: Implications for Transition. *Sustainability (Switzerland)*, 15(10).

Lyu, Z., Pons, D., Zhang, Y., and Ji, Z. (2021). Freight operations modelling for urban delivery and pickup with flexible routing: Cluster transport modelling incorporating discrete-event simulation and GIS. *Infrastructures*, 6(12).

Maersk (2023). Top 9 reasons for shipping delays.

Merk, O. and Notteboom, T. (2015). Port Hinterland Connectivity. Technical report.

Motraghi, A. and Marinov, M. V. (2012). Analysis of urban freight by rail using event based simulation. *Simulation Modelling Practice and Theory*, 25:73–89.

Notteboom, T. E. and Rodrigue, J. P. (2005). Port regionalization: Towards a new phase in port development. *Maritime Policy and Management*, 32(3):297–313.

Oliveira, J. B., Lima, R. S., and Montevechi, J. A. B. (2016). Perspectives and relationships in Supply Chain Simulation: A systematic literature review. *Simulation Modelling Practice and Theory*, 62:166–191.

Palmqvist, C. W., Lind, A., and Ahlqvist, V. (2022). How and Why Freight Trains Deviate From the Timetable: Evidence From Sweden. *IEEE Open Journal of Intelligent Transportation Systems*, 3:210–221.

Pant, R., Barker, K., and Landers, T. L. (2015). Dynamic impacts of commodity flow disruptions in inland waterway networks. *Computers and Industrial Engineering*, 89:137–149.

Pfoser, S., Treiblmaier, H., and Schauer, O. (2016). Critical Success Factors of Synchromodality: Results from a Case Study and Literature Review. In *Transportation Research Procedia*, volume 14, pages 1463–1471. Elsevier B.V.

PoR (2024). Barge Performance Monitor.

Qu, W., Rezaei, J., Maknoon, Y., and Tavasszy, L. (2019). Hinterland freight transportation replanning model under the framework of synchromodality. *Transportation Research Part E: Logistics and Transportation Review*, 131:308–328.

Rodríguez-Clare, A., Ulate, M., and Vasquez, J. P. (2023). Supply Chain Disruptions, Trade Costs, and Labor Markets A framework of international trade with unemployment. Technical report.

Santos, G., Behrendt, H., Maconi, L., Shirvani, T., and Teytelboym, A. (2010). Part I: Externalities and economic policies in road transport. *Research in Transportation Economics*, 28(1):2–45.

Schlake, B. W., Barkan, C. P., and Edwards, J. R. (2011). Train delay and economic impact of in-service failures of railroad rolling stock. *Transportation Research Record*, (2261):124–133.

Statista (2022). Average monthly delays for late container vessel arrivals worldwide from January 2019 to July 2022.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning : an introduction*.

Tavasszy, L. A., Behdani, B., and Konings, R. (2015). Intermodality and Synchromodality. *SSRN Electronic Journal*.

TheLoadstar (2022). Barge delays on N Europe waterways surge to highest peak in years.

van Dorsser, C., Vinke, F., Hekkenberg, R., and van Koningsveld, M. (2020). The effect of low water on loading capacity of inland ships. *European Journal of Transport and Infrastructure Research*, 20(3):47–70.

Van Riessen, B., Negenborn, R. R., Lodewijks, G., and Dekker, R. (2015). Impact and relevance of transit disturbances on planning in intermodal container networks using disturbance cost analysis. *Maritime Economics and Logistics*, 17(4):440–463.

Wang, X. (2016). Optimal allocation of limited and random network resources to discrete stochastic demands for standardized cargo transportation networks. *Transportation Research Part B: Methodological*, 91:310–331.

Wide, P., Kalahasthi, L. K., and Roso, V. (2022). Efficiency effects of information on operational disruption management in port hinterland freight transport: simulation of a Swedish dry port case. *International Journal of Logistics Research and Applications*.

Xu, J. and Hancock, K. L. (2004). Enterprise-wide freight simulation in an integrated logistics and transportation system. In *IEEE Transactions on Intelligent Transportation Systems*, volume 5, pages 342–346.

Zhang, K. (2024). Scenario Based analysis of port-hinterland transportation modes.

Zhang, Y., Guo, W., Negenborn, R. R., and Atasoy, B. (2022). Synchromodal transport planning with flexible services: Mathematical model and heuristic algorithm. *Transportation Research Part C: Emerging Technologies*, 140.

Zhang, Y., Negenborn, R. R., and Atasoy, B. (2023). Synchromodal freight transport re-planning under service time uncertainty: An online model-assisted reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 156.

Zhao, J. and Dick, C. T. (2024). Predicting and measuring service disruption recovery time in railway gravity hump classification yards. *Journal of Rail Transport Planning and Management*, 29.

Zhou, Y., Xie, R., Zhang, T., and Holguin-Veras, J. (2020). Joint Distribution Center Location Problem for Restaurant Industry Based on Improved K-Means Algorithm With Penalty. *IEEE Access*, 8:37746–37755.

A
Scientifc Paper

Resilient Synchromodal Transport through Learning Assisted Hybrid Simulation Optimization Model

Satrya Dewantara*, Bilge Atasoy†, Mahnam Saeednia‡
**Department of Transport and Planning*, *TU Delft*, Delft, The Netherlands
muhammadsatryadewantara@student.tudelft.nl
†*Department of Maritime and Transport Technology*, *TU Delft*, Delft, The Netherlands
b.atasoy@tudelft.nl
‡*Department of Transport and Planning*, *TU Delft*, Delft, The Netherlands
m.saeednia@tudelft.nl

Abstract—**The increasing volume of global freight trade, coupled with economic growth, necessitates ongoing innovation in optimizing freight operations. Over the past decade, the concept of synchromodality has been explored to encourage a modal shift from unimodal to multimodal transport. Synchromodality, with its flexibility feature, can create more resilient freight transport systems. Various models employing different techniques have been proposed to establish a resilient synchromodal framework capable of reacting to disruptions. However, there are only few studies addressing the unknown duration of disruptions. This research proposes a learning-based modular framework comprising to capture the dynamics of disruptions in multimodal transport and learn to make more effective decisions, thus addressing the challenge of limited prior knowledge about disruptions and enabling fast responses to disruptions.**

Index Terms—**Synchromodality; Resilient freight transport; Learning-based decision support framework**

I. Introduction

Road transportation dominates inland freight movement, accounting for 77% of the EU's freight in 2020 [1]. Logistic Service Providers (LSPs) often prefer unimodal transport primarily due to its inherent reliability [2]. However, road transport introduces several externalities, including accidents, road damage, environmental harm, and congestion [3]. The Intermodal concept promotes a shift of freight transport to other modes, such as barge and train. It can offer a cheaper option for inland freight transport due to the economics of scale [2]. Despite its cost-effectiveness, the share of intermodal transport remains low due to its lack of flexibility. The concept of synchromodality aims to increase the attractiveness of intermodal transport. By having a flexibility feature within the intermodal transport, synchromodal transport provides a higher number of combined routes, thus creating added values in the trade-off between price and time to the shippers [2].

Disruptions uncertainties negatively contribute to the efficiency of conventional intermodal transport [4] and could cause a severe economic loss. Disruption with low occurrence probability but high impact such as COVID-19 increased the logistic cost by 12% globally [5]. From another spectrum where disruption occurs frequently, a study by [6] estimates \$15.2 Million loss in a year due to train delays. This estimate made in 2011 will have a significantly higher value today.

This research investigates how to create a resilient intermodal freight transport system that leverages the flexibility offered by synchromodality. The main research question of this paper is *"To what extent does a learning approach improve the resilience of a synchromodal framework in coping with disruptions?"*. The resilience of a freight transport network is defined by its ability to recover from disruptions and is measured by the effort required to restore normal operations [7]. This paper provides a learning-based modular framework capable of capturing the disruptions generating a reaction plan. Developing the comprehensive model as mentioned above with practical solutions creates a challenge for scholars. Not only filling the gap of knowledge, the model should be practical to be implemented at the industry level, thus, shifting the freight transport paradigm away from road dependency towards more sustainable and flexible options. The modular framework utilizes optimization, simulation, and machine learning techniques allowing plug-and-play possibility for connecting with different existing/under development models and is extensible, making it applicable to different ecosystems of port-inland connections, expected to improve the solution space in generating decisions to react to the disruptions.

The paper is organized as follows: Section II provides a brief literature review. In section III, the disruption profiling is explained, followed by model formulation in Section IV. The model is implemented in Section V using a network adapted from a real-world operation. Finally, the research is concluded in Section VI to answer the main research question and providing directions for possible future research.

II. Literature Review

A. Concept of Synchromodality

Synchromodality is a concept developed to address the growing freight trade and its dynamics. Its objective is to thrive in the highly competitive transportation market and meet growing customer demands by enhancing flexibility and offering more customized services [2]. In recent years, various studies have aimed to define synchromodality. A study concludes that synchromodality is a planning system of multimodal transport that integrates supply chain stakeholders to flexibly adapt transport modes based on real-time insights [8].

Another study describes synchromodality as an efficient, reliable, flexible, and sustainable service enabled by stakeholders' cooperation supported by real-time information sharing [9]. [10] describes synchromodal as real-time transport planning using real-time information from different parties to increase flexibility, reliability, efficiency, and sustainability.

The flexibility attribute of synchromodality allows for real-time mode shifts, helping to navigate unexpected challenges in a volatile and competitive market [8]. To fully capitalize on this flexibility, certain conditions must be met. It is suggested that Logistics Service Providers (LSPs) can maximize this feature when shippers agree to mode-free or a-modal requests [8]. In this scenario, LSPs have the freedom to select the mode of transport that best suits the cargo's delivery requirements. This flexibility marks a significant departure from traditional transportation approaches, offering a more responsive and adaptable solution in dynamic market conditions. Beyond the added value from increased mode combinations, flexibility is a key benefit of synchromodal transport, allowing services to adapt and react to disruptions [11], which will be the focus of this research.

B. Dynamic Models for Synchromodal Framework

Dynamic models for synchromodal transport have been proposed in several studies. A Synchromodal Transportation Re-planning (STP) for hinterland transport is developed using a mixed integer linear programming (MILP) [12]. Using a different approach, a dynamic matching problem is proposed to deal with uncertain shipment requests [13]. In this model, the shipment requests are not completely known, but rather sequentially announced using a rolling horizon approach. This approach is adopted by another model for a global shipment matching problem and improved by incorporating disruptions in the service network [14]. The reaction to disruption in these two models are reallocation planning of the containers.

Other studies propose an agent-based model to compare the performance of unimodal, intermodal, and synchromodal for cost, time, and emissions. The model applies a synchromodal scenario by putting logic for each agent to reroute to the nearest and cheapest terminal if there is a disruption in the network, and monitor the impact on cost, time, and emission [15]. Another study proposed an agent-based framework for cooperative planning [16]. The model uses decentralized optimization with a negotiation scheme. It breaks down the problem into several sub-problems and lets the agents communicate with other agents to achieve each objective under disrupted scenarios. The model provides a sequence plan and re-plans it when exogenous events occur. A decision support system is proposed using a hybrid simulation-optimization model under synchromodal framework [17]. It employs an offline model to create the initial plan and an online model to react to the disruptions and selects one of three possible policies: wait, transshipment, or detour. The disruptions are categorized according to frequency and duration by assigning them to a random variable in the simulation. The online model will be triggered if there is a disruption occurs. The result of the study shows that the transshipment policy has the lowest share in all scenarios. This result could be a subject for future research since transshipment or mode shift plays an important role in Synchromodal. The other policy in this model is to wait, which is essentially the traditional reaction, and detour, which is practically difficult for barge and/or freight trains.

More recent studies integrate a learning approach within the synchromodal framework, such as a study by [14] that adopts the Reinforcement Learning (RL) approach in the global shipment matching problem under dynamic and stochastic travel time settings to address the *curse of dimensionality* of applying dynamic programming for solving the objective following Bellman's equation. Another study using RL technique under synchromodal framework is proposed by [18]. This study builds on top of an Adaptive Large Neighborhood Search (ALNS) proposed in the study by [19] to address the service time uncertainty in synchromodal transport. Unlike the study by [14] the learning agent in this study works side by side with the ALNS model instead of replacing its role. Integration of the RL approach in the synchromodal framework has a potential to address the variation in the nature of disruptions in the network. This is considered in the study [18], in which the RL agent works together with an optimization model, and a binary reward system is employed depending on whether a delay occurs due to disruptions and the taken actions. Their approach does not account for shipment volume or the length of the delay. In contrast, this research proposes a negative cost value as the reward. This method considers a higher delay penalty for higher shipment volumes, allowing the RL agent to prioritize larger shipments. Extensive disruption scenarios are incorporated for an improved learning process. The disruptions impact both demand and services, a feature that has not been extensively studies in the literature. The modular framework offers a plug and play mechanism allowing for improving/replacing/extending the modules as needed.

III. Disruption Categorization

The disruption in the freight network varies in type and impact and may require different reaction strategies either at strategic, tactical, or operational levels. It can be distinguished according to the frequency and severity, categorized into endogenous and exogenous factors [17], and come from different sources such as nature or human acts [20]. Another way to categorize the disruptions is by separating them into two spectrum as elaborated by [21]: low occurrence probability but severe impact, and high occurrence probability with low impact. The categorization of the disruptions could be useful to simplify a model while keeping the realistic behavior.

In this research, the disruption in the freight network is divided into two components encompassing disruptions on the service network (supply side) and the requests (demand side). Each profile represents a group of similar disruptions along with the possible impact and occurrence frequency.

On the supply side, the disruption categorization has more profiles than the request disruption. Five distinct profiles are

TABLE I: Service Disruption Profile

Profile	Description	Mode/ Location	Effect in the Simulation	Duration	Capacity Reduction	Occurrence per Year
1	Operational delays, road congestions	Train, Truck	Delay	1-3h	0%	30%
2	Operational delay, canal congestion	Barge	Delay	1-6h	0%	35%
3	Bad weather, labor strike, accident, systems maintenance	Train, Barge	Delay	12-48h	0%	6%
4	Terminal congestion, operational delay	Terminal	Delay	1-3h	0%	30%
5	High and low water level	Barge	Carrying capacity reduction	12-24h	15-20%	10%

TABLE II: Request Disruption Profile

Profile	Description	Location	Effect in the simulation	Delay Release	Valume Change	Occurrence per Year
6	Demand Change	Shipment	Volume change	-	-30% to + 30%	30%
7	Customs issues, main port operational delays	Shipment	Release time change	1-6h	-	30%
8	Mother vessels arrival delays	Shipment	Release time change	1-7d	-	5%

developed to represent different types of service disruptions as detailed in Table I. Each profile contains a group of disruptions with similar characteristics. The first profile is a frequent disruption that could cause a short delay on either the train or truck network. This could be caused by road congestion for trucks [15], or technical and communication problems on trains [22]. The second profile is a delay in barge service lines which, for instance, is caused by congestion in the river due to locks or high traffic [23]. The third profile, adopted from [15] is a possible delay due to more severe disruptions such as bad weather or systems maintenance. These disruptions could result in operations being halted for a certain period. The fourth profile is a disruption in the terminal such as equipment problems or port congestion [24]. The fifth profile is a reduction on barge carrying capacity due to the fluctuation of river water level [25], [26]. The low water level restricts barges from carrying containers with their full capacity because the barges need to reduce the draft, while the high water level could limit the height of the stacked containers on the barge to prevent collisions with bridges.

On the request side, three disruption profiles are considered, including two profiles of changes in container release time and a profile of alterations in shipment volume. In port-inland transportation, changes in the release time could happen due to several causes such as the late arrival of the mother vessels which can cause delays of release time up to seven days [27] or more minor issues such as customs clearance which cause delays of less than a day. Meanwhile, the volume changes could come from the shippers due to, for instance, unexpected increases in demand beyond long-term contracts. The request disruption profiles are presented in Table II.

The disruption profiles are created based on two spectrums explained by [21]. The high probability with a low severity level is represented by high occurrence per year and low value of severity (column 5 and 6) as attributed in Profile 1, Profile 2, Profile 4, Profile 6, and Profile 7. The other spectrum, the low probability with high severity disruption is attributed in Profile 3, Profile 5, and Profile 8. Each profile can only occur in certain locations. The third column in the table indicates the possible location of disruption when it occurs. It could be either in a terminal, service line, or directly on the shipment.

IV. Model Formulation

The research examines hinterland freight transportation, specifically focusing on the unidirectional flow of shipments from the main port to various inland terminals excluding the final leg of transportation from these terminals to distribution centers or warehouses. Each terminal is interconnected via dedicated service lines, which are exclusively served by one mode of transport, i.e. barges, trains, or fleets of trucks. The shipments may be transported directly or through multiple service lines, involving transfers at transshipment terminals, thus constituting a multi-modal transport network.

Real-world operations often face disruptions in both the service network and requests, manifesting as delays, capacity reductions, or changes in shipment release times. Under a synchromodal framework, the service network adapts flexibly in real-time to these disruptions. This flexibility primarily involves reallocating containers or matching them with available services, rather than altering fixed service schedules, which is typically challenging in practice.

To enable the synchromodal framework in the service network, the proposed model follows several assumptions. This study assumes the disclosure of enough information (in real time) among stakeholders to allow the central planner to re-route a shipment flexibly, assuming the necessary ICT infrastructure is available. Moreover, the modal free booking is applied to all shipments granting full authorization to the planning in reallocating the containers.

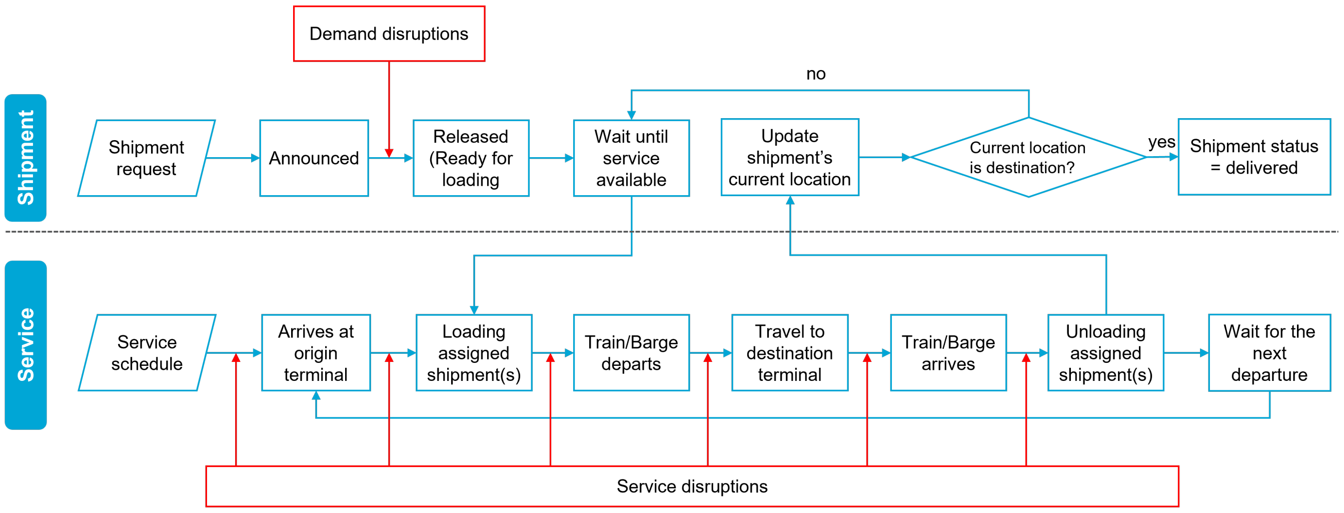

Demand disruptions
no
Shipment
Shipment request
Announced
Released (Ready for loading
Wait until service available
Update shipment's current location
Current location is destination?
yes
Shipment status = delivered
Service
Service schedule
Arrives at origin terminal
Loading assigned shipment(s)
Train/Barge departs
Travel to destination terminal
Train/Barge arrives
Unloading assigned shipment(s)
Wait for the next departure
Service disruptions

Fig. 1: Simulation Flow Process

A. *Simulation Module*

The main objective of this simulation is to capture the dynamic nature of disruptions in the hinterland freight network representing real-world operations to create an environment for implementing a decision support system.

The inland terminals are represented by nodes located at various locations. Multiple nodes of inland terminals could be located within the main port and serve as origin points for loading shipments onto transport modes. Other nodes are scattered further in the hinterland as the transshipment terminals or destination points. These terminals are characterized by handling capacity affecting the loading or unloading time. In this simulation, several parameters are assumed infinite such as stacking yard capacity and vehicle buffer area. Violation of these parameters could result in terminal congestion and could cause a delay. Rather than creating parameters, the port congestion is modeled using random variables to represent unexpected events.

The simulation module has three main components including service, shipment, and disruptions processes. The service is divided into fixed and flexible schedule services. The fixed schedule follows a predetermined departure time while flexible services depend on assigned requests. The service processes are characterized by the operation parameters of each mode including travel speed, carrying capacity, origin and destination, and departure time. These parameters play an essential role in determining the occurrence of events in the discrete event simulation.

Each shipment consists of a bundle of containers bound by a single contract and shares the same origin and destination. Therefore, each request is characterized by an origin, a destination, and a container volume. Additionally, time parameters such as announcement time, release time, and due time are specified for each request. The due time reflects the customer's expectations, with any delay beyond this time incurring a penalty. The shipment and services follow parallel processes while interacting with each other. The simulation keeps track of the actual costs for transporting each shipment from its origin the its destination. The cost components consist of storage, handling, travel, and delay penalty.

During the simulation process, the disruptions are enforced according to the profiles explained earlier. The disruption on requests only applies in a small time window after a shipment is announced and before it is released. However, the disruption on the service lines and terminals could occur anytime in the chain of events causing the disrupted request to wait until the disruption ends. The *always wait* policy represents the absence of synchromodal framework by always staying with the original itineraries despite the occurrence of disruptions, and potentially causing delays in shipment delivery. The simulation flow in Figure 1 represents the interaction between shipments, services, and disruptions.

B. *Hybrid Simulation-Optimization*

The mode assignment to each shipment depends on various parameters and is highly correlated with the performance of the freight network. An optimal mode assignment could result in low costs of transporting all requests from their origins to destinations. Under synchromodal framework, it is assumed that there is a centralized planner who is responsible for generating a shipment plan at regular intervals. Thus, shipment requests are delivered according to their requirements considering various objectives such as minimizing costs, maximizing on-time delivery, or minimizing emissions. Additionally, through the synchromodal framework, a planning module could be integrated into the simulation and automatically triggered to create a new shipment plan in case of disruption.

The modular framework presented in this study enables various analytical methods to plug-in, acting as a central planner. These could range from simple strategies such as first come first serve (FIFO) principle, heuristic methods or sophisticated optimization models with various objectives. The

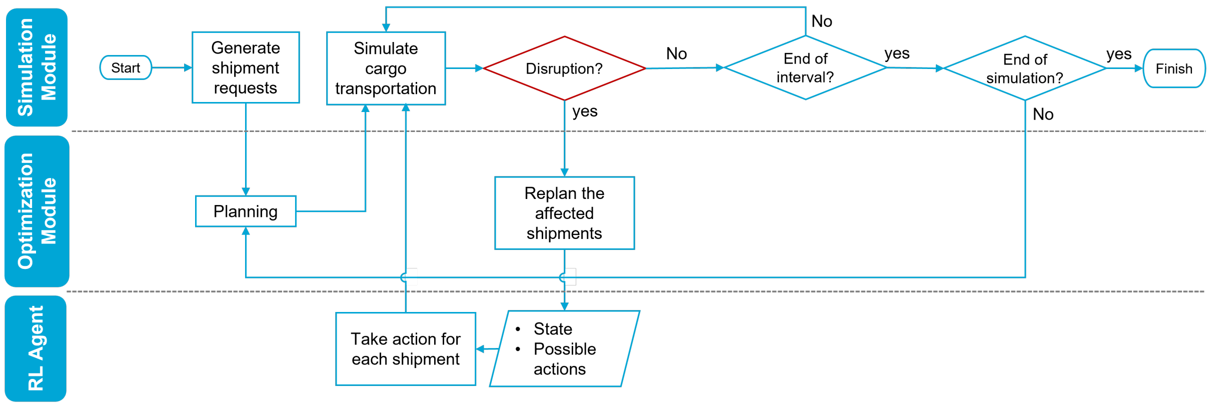


Fig. 2: Learning Assisted Hybrid Simulation-Optimization Flow

role of the optimization module that is embedded in the simulation module, is to match requests with available services considering the time and cost parameters associated with the service lines.

The detection algorithm is employed to determine the affected request based on the disruption occurrence time and location. It is important to distinguish between the affected requests and disrupted requests. The affected requests are only the ones that are still possible to be re-routed. There are three specific cases where a shipment is excluded from the affected requests despite the disruption in its itinerary. First, if the shipment is in a terminal or on board of a service line and the disruption occurs in that location (case 1). Second, if the shipment is on a service line, and a disruption occurs in that service line's destination terminal (case 2). Third, if the disruption occurs at the shipment's end destinations (case 3), the replanning in these three cases will not help the shipment to avoid disruption.

Upon receiving a new request or detecting a disrupted shipment, the optimization module is triggered to initiate the planning or re-planning process. The matching decision involves assigning a service line to the shipment. In the replanning process, the new assignment replaces the original itinerary. In contrast to the *always wait* policy, this hybrid simulation-optimization model *always reassigns* affected shipments to new service lines as the disrupted location is excluded from the possible solution space.

C. Reinforcement Learning Approach

Reassigning a shipment to a different service line during a disruption can potentially improve the resilience of the freight network and is feasible within a synchromodal framework. However, in some situations, it may be better for a shipment to wait until the disruption ends and then continue on its original itinerary. For example, if the disruption is expected to be brief or occurs at a terminal far ahead in the journey, where it is likely to be resolved before the shipment arrives. The challenge lies in the uncertainty of the disruption's duration. To address this, a reinforcement learning technique is integrated into the model, allowing it to decide whether to wait or reassign based on the learning agent's experience. This approach enables the model to make more informed decisions, balancing the benefits of reassignment with the potential advantages of waiting.

Using a value function, the RL agent can select the best action for a given state by learning from past experiences and extensive training. Unlike supervised learning, which relies on labeled data, the RL agent is guided by a reward system that indicates the effectiveness of each action. Properly modeling the action, state, and reward system is crucial for developing an optimal RL agent. Additionally, formulating a Markov Decision Process (MDP) is essential as it provides the framework for the RL agent to interact with the environment, apply actions, update states, and receive rewards as illustrated in Figure 3.

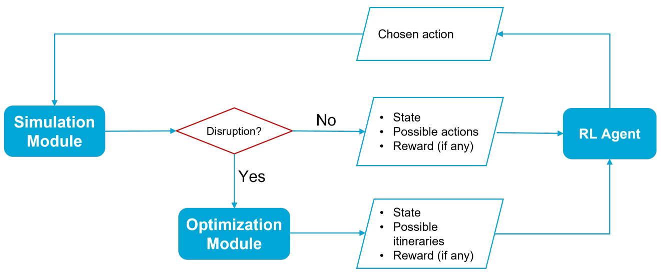


Fig. 3: Markov Decision Process in Learning Assisted Hybrid Simulation-Optimization

The action space for RL agent in this model consist of two: reassign and wait. Now, what do the "reassign" and "wait" actions exactly mean? Since it is the information that will be exchanged between RL agent and the simulation module, it needs to be clear so the feedback information would also be relevant. Those actions consist of a list of service lines, indicating which combination of service lines a request is going to be carried by. Assume a Request X is assigned to Barge1 and then transferred to Train1 before reaching its end destination, then, the assigned mode K_r is [Barge1, Train1]. If there is a disruption, and the optimization module suggests the reassignment to Truck2, then the possible action set will stay with the original itinerary, *wait*: [Barge1, Train1], and *reassign*: [Truck2]. The RL then chooses between those two options. However, to make it more granular, the itinerary is considered as a series of actions, and the action itself only contains a single service line. For instance, if in that case, the RL chooses to wait, then, the action is [Barge1], and save the [Train1] for the next action in the future. Therefore, after taking Barge1 and arriving at Barge1's destination, the simulation module will update the state to the RL agent, grant the reward, and take the next action Train1. There is no decision-making anymore at this stage since the possible action is only one, unless a disruption occurs during this process.

The decision made by the RL agent for each request is independent of other requests. In a way, there are multiple RL agents assigned for different requests and work in parallel, and it is called RL sub-agent. Once an RL sub-agent is assigned to a request, the loop starts until it reaches the terminal state for that request. If any disruption happens again in the future affecting the same request, that request is not assigned to a new RL sub-agent. This framework allows the RL sub-agent to make an action based on the action taken in the previous decision, such that an action within the same request is not independent of its previous action. From the explanation above, it seems like the RL agent works on a decentralized architecture. However, this is not the case in this framework. Even though the RL sub-agent makes decisions independently for each request, it has one centralized value function, updated whenever a reward is granted. This approach is illustrated in Figure 4.

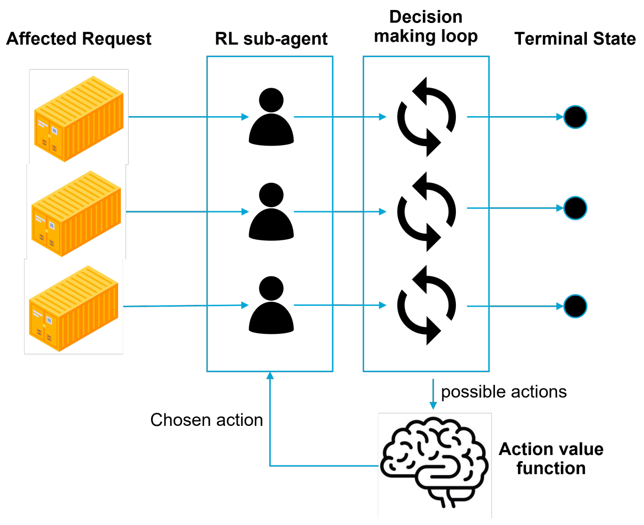


Fig. 4: RL Agent Framework

From the perspective of a centralized planner in synchromodal transport, defining the state of the RL agent is complex. The state space becomes exceedingly large if all shipments and service attributes are considered. However, by narrowing the perspective to a single shipment, defining the state becomes more manageable, given that each request has an independent decision-making process. In this approach, the state consists of six features: the request's current position, destination, due time, volume, type of disruption, and the current time. This framework allows decision making at the level of shipments, represented by RL sub-agents, enabling decision making based on the previous actions taken, ensuring that actions within the same request are not independent of each other.

The reward system is also implemented separately for each sub-agent. The RL agent uses this reward to update the action value function $Q(s, a)$ for each state s and action a using the off-policy Temporal Difference Control, Q-learning technique by following Equation 1. Unlike *Monte Carlo*, Q-Learning does not have to complete one full episode to update the action value function, instead, it uses the current estimate in the equation. This method proves to reach convergence faster [28].

$$\begin{aligned} Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) \\ - Q(S_t, A_t)\big] \end{aligned} \tag{1}$$

The total transportation cost between terminals, including transport and storage, handling, and delay costs, is used as a reward for actions. The reward calculation starts when a disruption impacts a shipment and continues until either an action resolves the issue or another disruption affects the same shipment. Since rewards are negative costs, the agent selects actions based on the value function, choosing those with values closest to zero to minimize costs.

This framework allows the RL sub-agent to make decisions based on the previous action taken, ensuring that actions within the same request are not independent of each other. The learning technique is integrated with the existing hybrid simulation-optimization to create a comprehensive decision support system for a resilient synchromodal framework. These three modules work together by exchanging information, as illustrated in the flow diagram in Figure 2, forming a complete model of Learning-Assisted Hybrid Simulation-Optimization within the synchromodal framework.

V. Case Study

For the case study, the service network uses a real port-hinterland network, with synthetically generated requests. The service network is adapted from the European Gateway Services (EGS) network, now known as Hutchinson Ports Europe Intermodal (HPEI). This network connects the Hutchinson Ports ECT in the Port of Rotterdam to inland terminals in the Netherlands, Belgium, and Germany through the Rhine-Alpine corridor. The details of the network, including distances between terminals and the service schedule, are derived from a study by [18].

The network comprises 10 terminals, with 3 of them located in the Port of Rotterdam and the rest scattered across inland terminals. It includes 49 barges, 33 trains, and 34 truck service lines operating between these terminals. The network is illustrated in Figure 5. Each service line represents one origin and one destination with a unique departure time each week. Some connections have multiple lines for the same

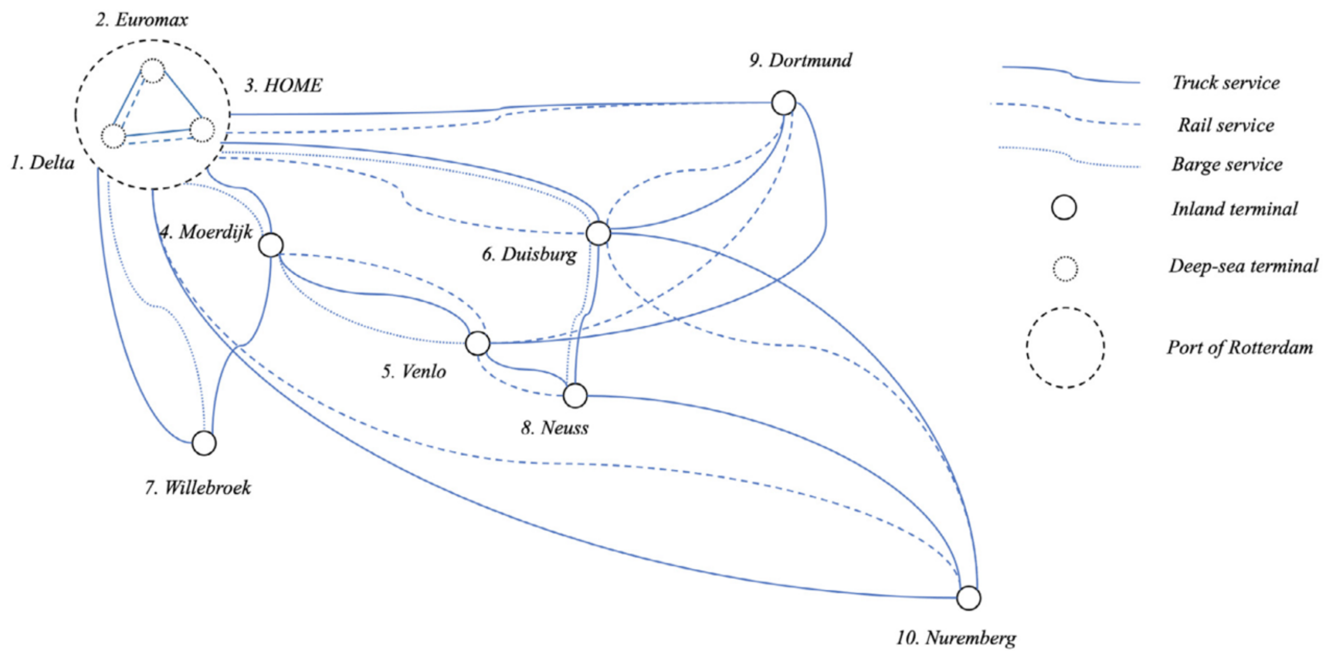


Fig. 5: EGS Service Network
source: [18]

mode of service from the same origin-destination pair but with different departure times, indicating service frequency. Each service line only serves two nodes, with no modes providing multiple stops at different terminals. While truck services have one line for each origin-destination pair, it is assumed that truck service capacity is unlimited and can accommodate multiple departures as per assigned requests. The carrying capacities of barge and train services are 160 TEU and 90 TEU, respectively. Travel speeds are 15 km/hour for barges, 45 km/hour for trains, and 75 km/hour for truck services. The distances between terminals vary for each service mode.

Requests are generated randomly according to the service capacity proportion of each origin-destination pair. For instance, the number of services connecting Delta to Nuremberg is much higher than those connecting Delta to Willebroek, so the request volume from Delta to Nuremberg is proportionally higher. Requests are announced in batches, each consisting of multiple shipment requests. For the default scenario, 200 requests are generated and announced proportionally over 3 weeks. This default demand scenario is used to train the RL agent.

A. *Optimization Model Plug-In*

For the case study, a matching problem optimization algorithm inspired by [29] and [30] is integrated as a plug-in to the optimization module. The model is originally designed as a learning-based robust optimization consisting of a prediction module and a decision module. The prediction module is used to derive uncertainty sets for road travel time. The uncertainty sets, then, are fed to the downstream robust optimization model in the decision module. Based on uncertainty sets, the robust reformulation is done to convert the robust optimization model to a deterministic model which is solved by mathematical solvers. However, from this comprehensive framework, only a fragment of the decision module, the deterministic optimization model, is plugged into the optimimzation module of the learning-assisted hybrid simulation-optimization.

The inputs of the optimization model are shipment requests and service lines. It uses a path-based approach and employs a preprocessing algorithm to generate paths from available service lines to reduce the computational burden within the optimization model. The path generation algorithm consists of all possible services that satisfy the spatiotemporal requirements and transshipment feasibility.

The path is generated according to each service line's departure and arrival time and each path is attributed with all the cost components including the transshipment and storage costs if any. The cost calculation of each path considers all the modes of service within the path. If there are two modes of service in one path, then the travel cost is the total of those two modes' travel costs given the travel distance and time in the path. Since the departure and arrival time is known, the storage time is calculated according to the service schedule of the mode of service in the path used for calculating the path storage cost.

$$\begin{aligned}\min_{x^t,y^t} \Bigg(& \sum_{r\in R} v_r(1-y^t) + \sum_{r\in R}\sum_{s\in S} c_s x^t_{rs} u_r \\ & + \sum_{r\in R}\sum_{s\in S} c^T_{ri} x^t_{rs} u_r + \sum_{r\in R}\sum_{i\in I} c^S_i x^t_{rs} \tilde{S}^s_{ri} u_r \\ & + \sum_{r\in R} c^D_r \tilde{d}^r_{ri} + \sum_{r\in R} c^L_{ri} u_r \Bigg)\end{aligned} \tag{2}$$

The objective of the optimization model is to minimize the total transport cost including travel, transshipment, storage delay, canal, handling, and emission. This model was originally designed using a case study in Great Lakes, Canada, therefore there is a canal cost that applies to that context. The objective function is adjusted to fit the model with the context of the case study in this research. The canal cost and emission cost are eliminated from the objective function resulting in the equation presented in Equation 2.

The Equation 2 attempts to maximize the number of matches and minimize the total transportation cost which includes six terms: the first term enforces the model to match shipment requests as many as possible. The second term determines the transportation cost, the third term denotes the transshipment cost, the fourth term accounts for storage cost, the fifth term shows the delay penalty costs, and the final term represents the loading/unloading cost.

by integrating an optimization algorithm, with the given input, the integrated model needs around 6 to 7 minutes to complete one episode. This is mainly because of the high computational work in solving the optimization problem. With the 6-minute run-time, it requires a huge amount of time to give the RL agent a sufficient number of episodic training to see its improved performance. Therefore, adjustment is required to decrease the solution time.

The first modification is to separate the planning process from the episodic training. The optimization algorithm is run separately before starting the episodic training and matches all the demand instances at one time. The matching results are then attached to each request in the input, so, instead of having an empty list of assigned services, each request has an itinerary in the shipment input table for the simulation. Without having the separation of the planning process, the simulation triggers the optimization module every week in every episode and always gets the same matching results. This separation approach saves all the unnecessary computational burden that only offers the same matching results.

The second modification is made to reduce the number of optimization algorithm triggers during the replanning process. It is by providing a solution pool consisting of K-best solutions for each shipment during the planning process. Instead of providing only one solution during the planning phase, the optimization algorithm generates multiple solutions consisting of a second, third best solution up to ten solutions in the pool. These solutions are used during the replanning process as backup itineraries. The simulation algorithm will eliminate the disrupted itineraries and take the best from the remaining solutions. Whenever a shipment departs from one terminal, it also eliminates other solutions that use different routes from that terminal. The optimization algorithm could still be triggered in the replanning process if there are shipments with no remaining backup itineraries in the solution pool. This approach could reduce the number of triggers during the replanning and ultimately reduce the runtime. However, it could also create suboptimal solutions since it does not consider all the constraints in the real-time.

The solutions pool approach reduces run-time but introduces a risk where reassigning requests could consume the capacity of undisrupted shipments. For instance, if two disrupted requests, each with 20 containers, are both reassigned to Barge1 from the second-best solution in the pool, and Barge1 only has 30 units of free capacity, an issue arises. An undisrupted request originally scheduled for Barge1 might not be loaded due to the capacity limit, causing it to miss its departure and wait for the next one.

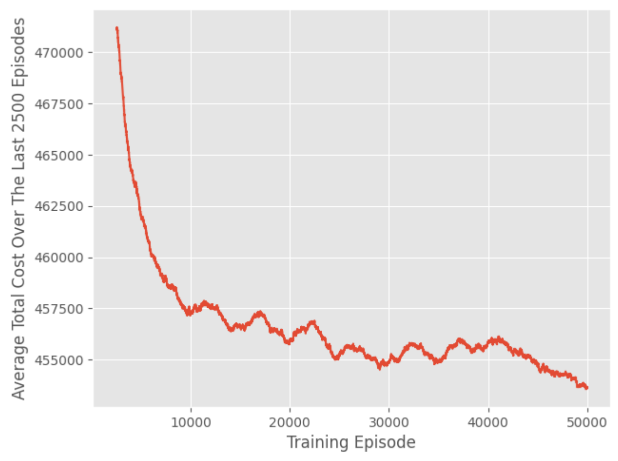
This problem is significant because the reinforcement learning (RL) algorithm cannot account for the additional costs incurred from undisrupted requests missing their departures. To address this, an algorithm was developed to regulate the loading sequence by prioritizing undisrupted requests. This ensures that only reassigned requests might miss the departure due to capacity issues. The extra costs from these delays penalize the RL agent, providing valuable experience to guide future decisions.

B. Model Training

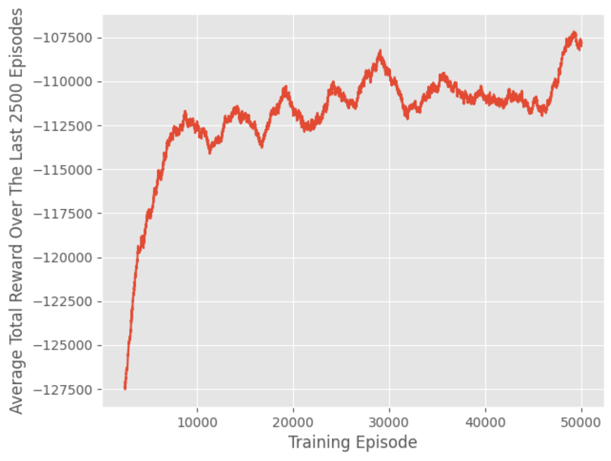
The first step in evaluating the performance of the model is to train the RL agent. The default demand scenario consisting of 200 requests is used and the requests are distributed across 3 weeks period making it announced around 60 to 70 requests each week. To generate the path, the travel cost for each service line must be calculated. With all the properties of each service line and two parameters of travel cost comprising the time-related and distance-related cost, the total travel cost for each service line can be calculated.

The simulation is set to have a 5-week duration to make sure all the requests are delivered in case some of them have long delays resulting in more representative results for all delivery processes. Each episode of training has a different random seed to ensure different disruptions. In the training, only the service disruption is included, because the demand disruption does not trigger the RL agent. Finally, the model is set to have 50,000 training episodes and for every 1,000 episodes, the action value function in the form of Q-table is extracted to be used in the result analysis.

The three possible policies consisting *always wait*, *always reassign*, and *RL-based policy* are evaluated using the model. The *always wait* is a policy without having a replanning procedure. The *always reassign* is a policy that triggers the optimization algorithm every time a disruption occurs and always accepts the solution. This represents a naive synchromodal framework without the RL agent. The last policy is by



(a) Total Cost - Episodic Training



(b) Total Reward - Episodic Training

Fig. 6: Episodic Training for 50000 Episodes

employing the RL agent to find the *optimal policy* using ϵ-*greedy policy* during the training. To balance the exploration and exploitation [28], the ϵ value provides a small amount of probability so the RL agent occasionally chooses an action with a lower value and explores the probability of having a better reward in a longer run. However, for the implementation, a *greedy policy* is applied by always selecting actions with a higher value in the given states.

Figure 6 presents the result of the training over 50,000 episodes using ϵ-greedy policy with the rolling mean applied using a 2,500-episode window. The trend shows that the mean total cost and the reward start converging by starting to show a flat trend. After the model completes around 10,000 training episodes, both the improvement in mean total costs and rewards are not as significant as in the early training stage. Nonetheless, the training continues to expose the RL agent with more experiences, thus, resulting in a resilient model to react to various disruption scenarios.

C. Results and Numerical Experiment

After the episodic training is completed, the RL agent performance using *greedy policy* is compared with the benchmark policies (*always wait* and *always reassign*). The performance comparison is performed by simulating each policy through multiple episodes with randomized disruption. Instead of sampling one case, the evaluation through multiple episodes can show how each policy responds to different cases of scenarios, thus, providing more insightful information about the resilience of the policy. In this experiment, the *greedy policy* (learning-assisted) is simulated seven times using different Q-table extracted from different amounts of training episodes ranging from 1,000 to 50,000 training episodes. The disruption profile set used in this evaluation is the same as the ones used in the training. Each policy is simulated 20 times and plotted using a box plot presented in Figure 7.

Figure 7 shows the total costs of each policy. GP{NUMBER} indicates the *greedy policy* with the number of training when the Q-table is extracted. On one hand, *always wait* policy significantly outperforms *always reassign* policy with the given disruption profile set. However, it shows some outliers with high costs in 4 cases indicating unstable performance. On the other hand, the *greedy policy* shows improvement along with the number of training and outperforms the *always wait* policy in some cases. In the beginning, the *greedy policy* shows unstable performance indicated by the high variance of total costs. The performance is getting stable as the training episodes increase. Additionally, the total cost average of the *greedy policy* keeps decreasing along with more completed training episodes. At 50,000 training episodes, the *greedy policy* manages to have a lower mean of total costs compared to the *always wait* policy.

Although showing the improvement relative to the benchmark policy, the chart does not clearly show how the learning-assisted model performs. To see this, the investigation is taken into more detail by comparing each episode. In this evaluation, the *always reassign* policy is excluded since it was clearly outperformed in Figure 7. With the same disruption profile set, the total cost in each episode is compared between *greedy*

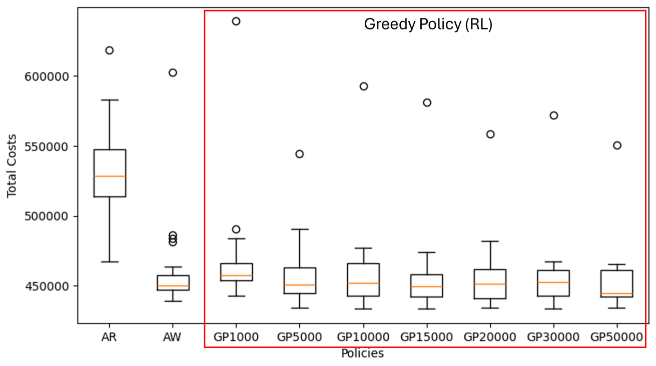


Fig. 7: Policy Comparison (Multiple Cases)

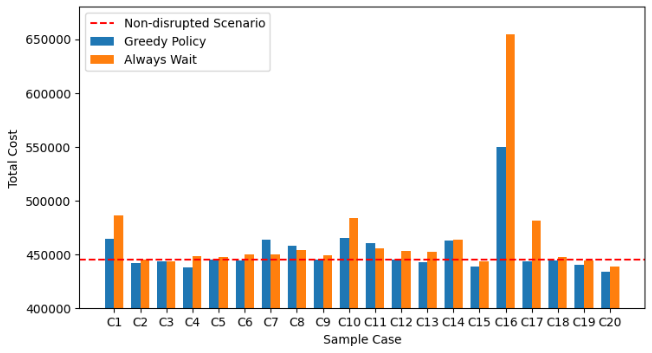


Fig. 8: Comparison Between Always Wait and Greedy Policy in Episodic Simulations

policy using the Q-table after 50,000 episodes and *always wait* policy. The result is presented in Figure 8.

The blue bars and orange bars represent the total cost of each policy respectively. The lower cost indicates a better performance of one policy over the other. The red dashed line is the total cost of a non-disrupted scenario. It is interesting that in a few cases, such as case C20, the total cost of a disrupted scenario appears lower than the non-disrupted. This happens because of several unique cases. For instance, in case C20, one request is planned in the first week to take service in the week after because its release time is slightly more than the departure time of the service mode in the first week. However, because of the delay in the departure time of the service, the shipment can be loaded to the assigned service in the first week, thus, resulting in lower storage costs.

The chart reveals that in only 4 out of 20 simulations, the *always wait* policy incurs lower total costs compared to the *greedy policy*. Among these 4 simulations, C3 exhibits an insignificant difference. This suggests that with the same disruption profile used for training, the *greedy policy* generally makes better decisions in response to disruptions with 80% of cases the *greedy policy* yields better performance. In these 20 sample cases, the cost savings by the *greedy policy* range from 0.08% to 15.96%. If the calculation only considers the affected requests, the highest savings from *greedy policy* over benchmark policy reaches 35,7%. The amount of savings depends on the severity of delays caused by the disruptions in the network.

Further investigation at the action level reveals a correlation between the number of actions taken for a request and its total cost. The RL agent, in a few cases, takes multiple actions for a single shipment because the shipment experiences multiple disruptions. The Q-learning technique updates each action value in a given state by considering the reward and the discounted future state, as explained in Equation 1. This method captures stochastic behavior by updating the action value function after visiting a certain state. If a state occurs frequently, the action value will continuously update until it converges. Conversely, if a state is rarely visited, the action value function remains less updated according to the rewards from this state. Given the disruption profile, such multiple disruptions are rare in the training set. Consequently, the action value function suggests immediate actions without anticipating future disruptions, as the model has not frequently encountered these scenarios during training.

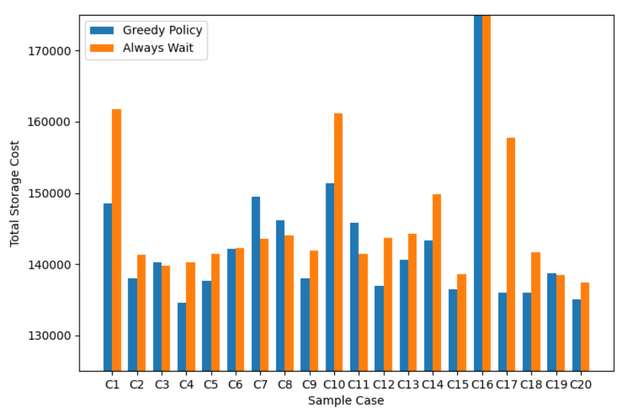
Another observation is made to see how the decision made by the RL agent relates to each cost component (Storage, Travel, Handling, and Delay). From the same case, each cost component is compared between 2 policies as presented in Figure 9. The y-axis scale is set to be the same with each tick representing 50,000 Euro to give easier comparison across different cost components. The scaling ignores case C16 since it is an extreme case with a significantly higher cost.

The result shows that *greedy policy* tends to take the option with higher travel cost, indicated by Figure 9b, to reduce the storage cost and delay penalty as shown in Figure 9a and Figure 9d. Meanwhile, handling costs are not really affected by different policies. It is essential to understand that this tradeoff behavior is a result of the given cost parameters as it is directly linked to the reward system. The tradeoff behavior might change if the cost parameters used for the training are different

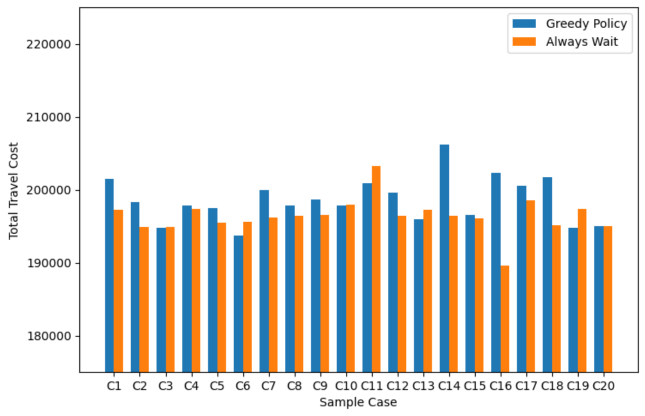
The current approach of the *greedy policy* involves selecting an itinerary from the solution pool for reassignment, and only triggers the optimization algorithm if there is no possible solution found in the solution pool. This method is compared to consistently triggering the optimization (*always optimize*) module during replanning. Figure 10 illustrates the comparison results. The cyan bars represent the total cost of always triggering the optimization module. The results indicate that the *greedy policy* using the solution pool generally performs better. This is because, in the *always optimize* case, the optimization module generates several solutions during replanning that are different from the solution pool given the real-time information and they are not yet explored during the training. Ultimately, this leads to suboptimal decision-making. However, in some cases such as case 14 (C14) and case (C10), the *always optimize* policy yields lower total costs. Further investigation reveals that the optimization model generates several solutions distinct from the solution pool that results in lower costs. This suggests that the solutions provided by the optimization module during replanning could potentially yield better performance if selected correctly.

D. *Experiment on Different Disruption Sets*

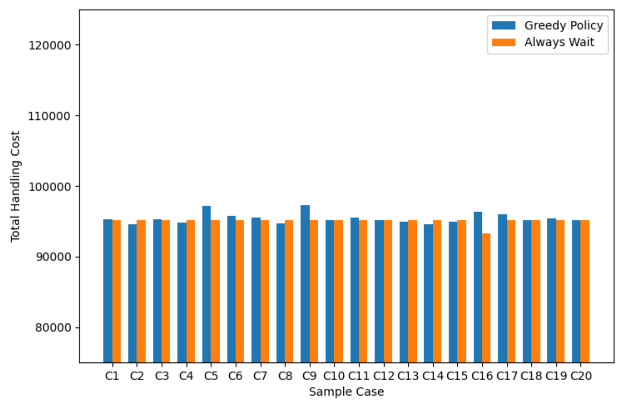
The experiment is performed to evaluate the model that has completed 50,000 training episodes with different disruption scenarios and compare the performance with the benchmark policies. To perform the experiment and deduce the change of behavior across different types of disruption, sets of disruption profiles are established and presented in Table III. The disruption on trucks is omitted from profile1 due to technical reasons while it also hardly affects the network. The disruption sets are characterized by different occurrence rates. S1 is scenario 1 with the lowest occurrence rate. The occurrence rate keeps increasing two times for each disruption set until



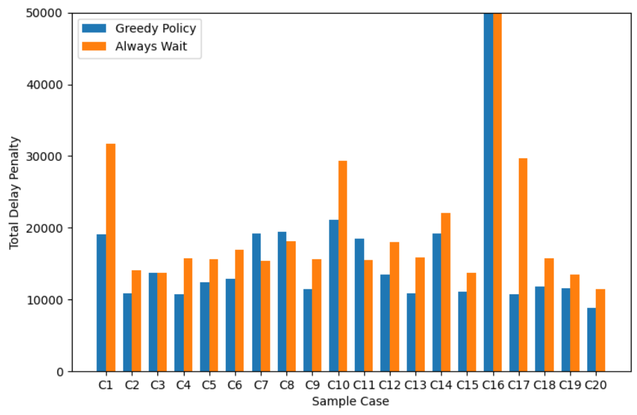
(a) Storage Cost Comparison



(b) Travel Cost Comparison



(c) Handling Cost Comparison



(d) Delay Penalty Comparison

Fig. 9: Cost Element Comparison

the set S5 with an extremely frequent occurrence rate. To put it in context, the occurrence rate of the default disruption set used in the training is between S2 and S3. The result of the second experiment is presented in Figure 11a

The experimental setup is nearly the same as the first experiment. The model is simulated 20 times for each disruption set and policy. The total cost of each simulation is shown in Figure 11a. The *greedy policy* is plotted in blue colors while the *always wait* policy is in orange colors. The first notable finding is that changes in the disruption occurrence rate affect the total costs, especially in S4 and S5. As disruption frequency increases, so do the total costs, as indicated by the ascending trendline.

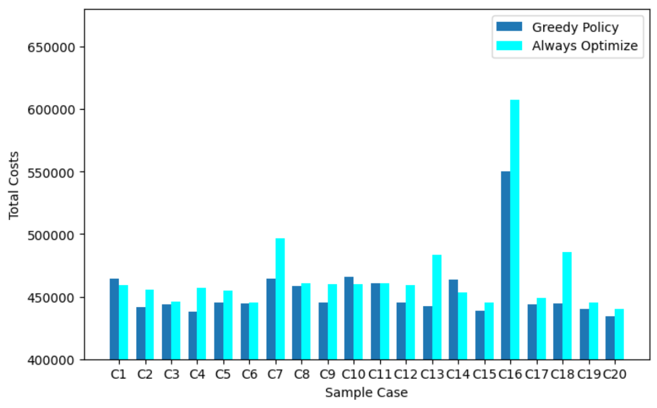


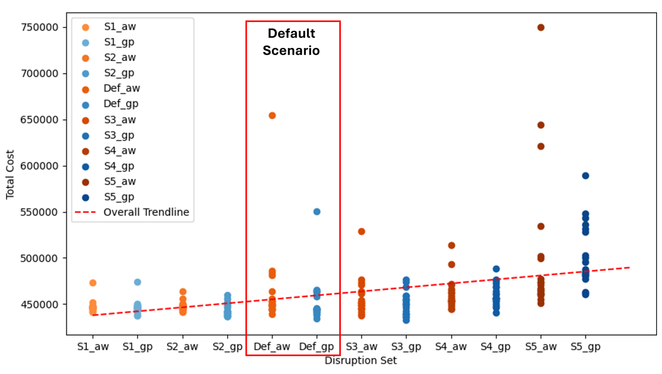
Fig. 10: Comparison Between Solution Pool Approach and Always Optimize in Episodic Simulations

An interesting observation is that the learning-assisted model performs best under disruption scenarios S2 and S3. However, its performance declines compared to the benchmark policy as disruption frequency increases. Although the *greedy policy* shows more stable performance, indicated by fewer outliers, many cases yield higher costs compared to the *always wait* policy as the disruption frequency increases. This decline is due to differences between the probability distributions in this experiment and those used during episodic training. S2 and S3 are most similar to the distributions used in training, allowing the *greedy policy* to perform well in these cases.

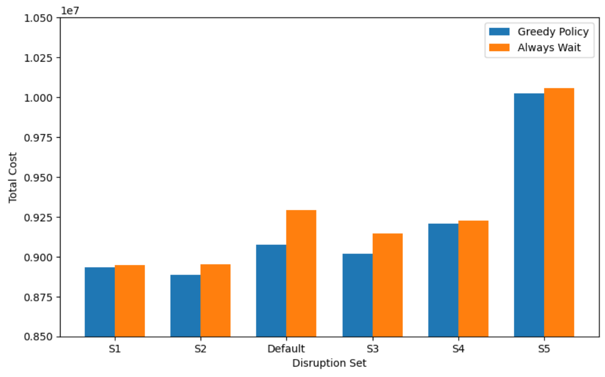
In the less frequent disruptions, the *greedy policy* has almost the same performance as the benchmark policy. This is because the fewer disruptions in the network, the closer the *always wait* policy with the optimal policy. This is also indicated by the fact that only 64 times the RL agent is triggered in total throughout

TABLE III: Disruption Scenarios

Profile	Severity	Location	Impact Type	LB Duration	UB Duration	Occurrence Rate (Lambda)				
						S1	S2	S3	S4	S5
Profile1	Low	Train	Delay	3	6	0.00021	0.00042	0.00084	0.00168	0.00336
Profile2	Low	Barge	Delay	3	9	0.00024	0.00048	0.00096	0.00192	0.00384
Profile3	High	Train, Barge	Delay	12	48	0.00004	0.00008	0.00016	0.00032	0.00064
Profile4	Low	Terminal	Delay	3	6	0.00021	0.00042	0.00084	0.00168	0.00336
Profile5	Low	Barge	Capacity reduction	12	24	0.00004	0.00008	0.00016	0.00032	0.00064



(a) Total Costs for Different Disruption Sets



(b) Aggregated Costs Comparison

Fig. 11: Experiment on Different Disruption Sets Results

20 simulations. In contrast, the RL agent is triggered 219 times in the default disruption set.

This finding is also indicated by the aggregated costs comparison presented in Figure 11b. The figure presents the total costs throughout the 20 simulations for each disruption set. It provides the overall performance of the policy. The chart shows that across all disruption sets, the total costs generated by the *greedy policy* are lower than the *always wait* policy. In particular, the cost gap in S2 and S3 is larger compared to the other disruption sets.

Despite overall better performance in the aggregated total costs, the data demonstrates that the *greedy policy* outperforms the benchmark policy only in scenarios S2 and S3 based on the number of simulations where it achieves better results. Specifically, Figure 12 summarizes the performance across various disruption sets. As explained earlier, in disruption set S1, the *greedy policy* is outperformed by the *always wait* because in the lower disruption frequency, the *always wait* policy is getting closer to the optimal solutions. A thorough investigation reveals there are several unstable values, again due to the stochastic environment.

In disruption sets S2 and S3, the *greedy policy* performs better in 13 and 16 out of 20 simulations, respectively. This indicates that the *greedy policy* is more effective approximately 65% to 80% of the time when the probability distribution closely resembles the episodic training. Meanwhile, in disruption sets S4 and S5, the *greedy policy* performance is declining due to many disruptions in the network creating more multiple disruptions for one shipment.

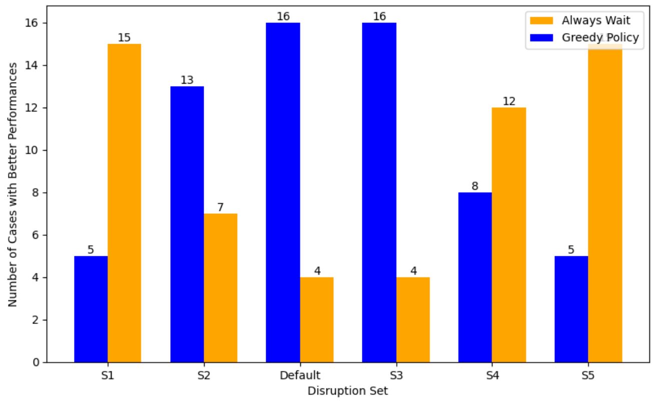
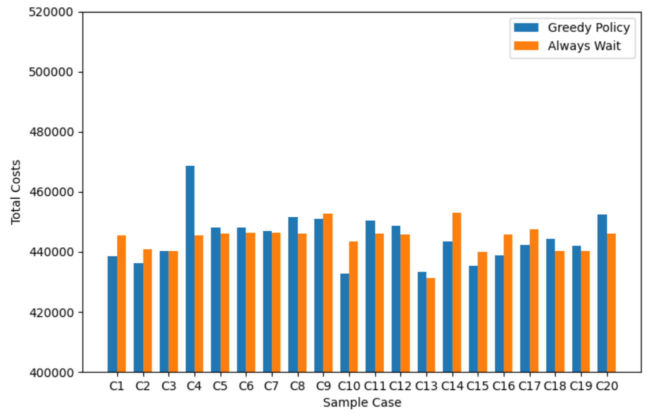
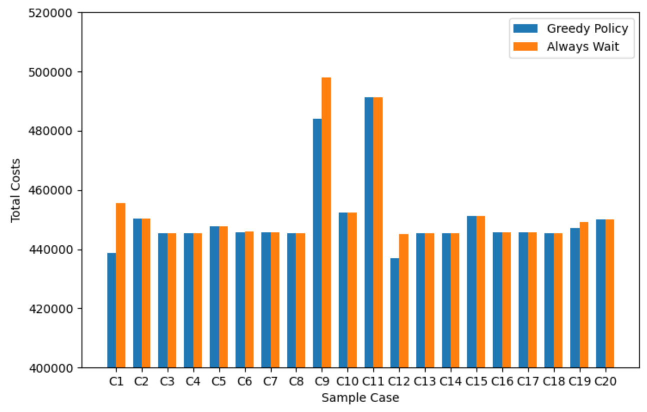


Fig. 12: Performance Comparison by Number of Cases

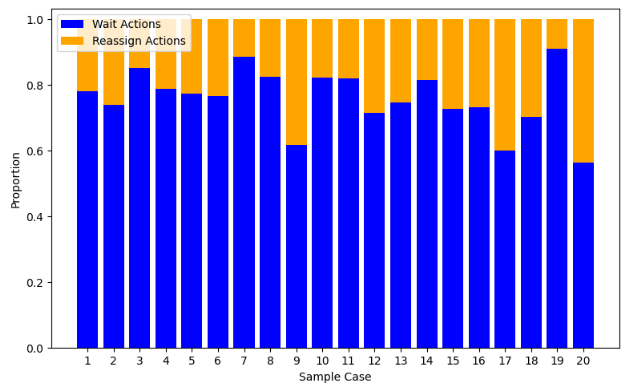
The next experiment on the disruption set is to simulate the model through 2 distinct scenarios: Short delay disruptions and severe delay disruptions. For the short delay, the disruption profile4 is omitted from the disruption set, while the severe delay only includes the disruption profile4. The occurrence rate used in this experiment is the same as the default case in the episodic training. However, to apply enough disruption, the probability occurrence for severe disruption set is slightly increased to the same level as in disruption set S3 from the previous experiment to apply more disruption but still keeping the distribution closer to the one used in the training. The result of this experiment is presented in Figure 13



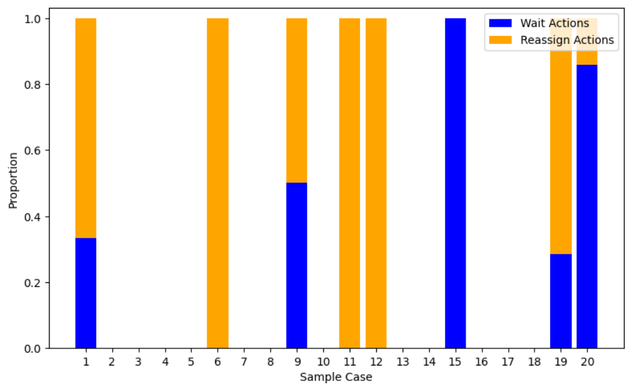
(a) Total Cost - Short Delays



(b) Total Cost - Severe Delays



(c) Action Proportion - Short Delays



(d) Action Proportion - Severe Delays

Fig. 13: Short and Sever Delays Experiment

The two charts on the left represent the result of applying short delays while the charts on the right show the result of applying severe delays. Figure 13a shows that the disruptions create many delays indicated by only one case with the same total cost (C3). This is possible due to the high disruption occurrence rate. However, the low severity of the disruption makes the optimal solution closer to the *always wait* policy. The poor performance of the RL in some cases, again, is caused by multiple disruptions in one shipment. An important point to highlight is, due to the low severity disruption, the highest savings by the RL-agent are only 2.4%.

In contrast, in the severe delays scenario, the costs between *greedy policy* and *always wait* in most cases are almost the same due to the low occurrence rate as presented by Figure 13b. However, once the disruption creates delays in the network, the RL-agent can produce higher savings compared to the short-delayed disruption. The savings in the severe delay scenario reaches up to 3.65%.

The charts at the bottom (Figure 13d and Figure 13c) present the proportion of actions between wait and reassign taken by the RL agent. Some cases in the severe delays scenario are blank because there was no action taken by the RL agent in that specific case. As expected, in the severe delays, despite only a few actions taken due to low disruption frequency, the proportion of taking reassign action is generally higher than wait action.

VI. Conclusions and Recommendations for Future Research

This paper proposes a modular simulation-optimization decision support tool to address the dynamic nature of demand and disruption in synchromodal transport. To address the unknown duration of disruptions in the synchromodal framework, an RL agent is integrated. *Always wait* policy is selected as a benchmark policy, in which a shipment always stays with its original itinerary even though disruptions occur. The other policy is the *greedy policy* through the RL-assisted model where the RL-assisted selects actions between wait or reassign in reaction to disruptions. The RL-assisted model outperforms the benchmark policy by using its experience to choose the proper actions in response to disruptions.

From the performance comparison, the research answers the underlying question *"To what extent does a learning approach improve the resilience of a synchromodal framework in coping with disruptions?"*. After completing 50,000 episodes of training, a case study using real service network data was conducted. The learning-assisted model was simulated 20 times using a *greedy policy* and compared with the benchmark policy of always waiting. The greedy policy outperformed the benchmark in 16 out of 20 simulations or 80% of the time, achieving cost savings of up to 15.96% of the total cost compared to the benchmark, or 35.7% when considering only the affected requests. While the extent of cost reduction depends on network delays, the model's ability to perform better across various disruption scenarios is more crucial than the specific amount of savings. Additionally, the model proves to still provide better performance compared to the benchmark policy in a certain range of different probability distributions of the disruption, as indicated by the result of up to 30% difference in occurrence rate. The overall performance suggests that the proposed model can create a more resilient synchromodal framework by utilizing reinforcement learning.

The results of this research indicate that utilizing a learning approach within a synchromodal framework could bring added value and serve as a foundation for future research directions. First, incorporating data-driven disruptions and extensive profiles, such as severe disruptions at terminals or service line infrastructure, could provide more realistic scenarios for the RL agent and enhance the reliability of the model. Second, integrating a meta-heuristic optimization model to address runtime issues, enabling the optimization module to be triggered consistently during training and fully exploring the model's potential. Third, Implementing a communication scheme between multiple sub-agents to generate actions leading to system-wide optimization could be a valuable future direction. Finally, addressing the limitations of the current model by implementing deep reinforcement learning techniques. This approach allows the model to estimate values of unvisited states, enabling the RL agent to perform well without visiting all states. This would allow for more complex state features and more accurate decisions without significantly increasing training time.

References

[1] Eurostat, "77% of inland freight transported by road in 2020," 2020.

[2] L. A. Tavasszy, B. Behdani, and R. Konings, "Intermodality and Synchromodality," *SSRN Electronic Journal*, 4 2015.

[3] G. Santos, H. Behrendt, L. Maconi, T. Shirvani, and A. Teytelboym, "Part I: Externalities and economic policies in road transport," *Research in Transportation Economics*, vol. 28, pp. 2–45, 1 2010.

[4] T. Delbart, Y. Molenbruch, K. Braekers, and A. Caris, "Uncertainty in intermodal and synchromodal transport: Review and future research directions," 4 2021.

[5] A. Rodríguez-Clare, M. Ulate, and J. P. Vasquez, "Supply Chain Disruptions, Trade Costs, and Labor Markets A framework of international trade with unemployment," tech. rep., 2023.

[6] B. W. Schlake, C. P. Barkan, and J. R. Edwards, "Train delay and economic impact of in-service failures of railroad rolling stock," *Transportation Research Record*, pp. 124–133, 12 2011.

[7] L. Chen and E. Miller-Hooks, "Resilience: An indicator of recovery capability in intermodal freight transport," *Transportation Science*, vol. 46, no. 1, pp. 109–123, 2012.

[8] B. Acero, M. J. Saenz, and D. Luzzini, "Introducing synchromodality: One missing link between transportation and supply chain management," *Journal of Supply Chain Management*, vol. 58, pp. 51–64, 1 2022.

[9] R. Giusti, D. Manerba, T. G. Crainic, and R. Tadei, "The synchronized multi-commodity multi-service Transshipment-Hub Location Problem with cyclic schedules," *Computers and Operations Research*, vol. 158, 10 2023.

[10] M. A. M. De Juncker, D. Huizing, M. R. O. del Vecchyo, F. Phillipson, and A. Sangers, "Framework of Synchromodal Transportation Problems," pp. 383–403, 2017.

[11] R. Giusti, D. Manerba, G. Bruno, and R. Tadei, "Synchromodal logistics: An overview of critical success factors, enabling technologies, and open research issues," *Transportation Research Part E: Logistics and Transportation Review*, vol. 129, pp. 92–110, 9 2019.

[12] W. Qu, J. Rezaei, Y. Maknoon, and L. Tavasszy, "Hinterland freight transportation replanning model under the framework of synchromodality," *Transportation Research Part E: Logistics and Transportation Review*, vol. 131, pp. 308–328, 11 2019.

[13] W. Guo, B. Atasoy, W. Beelaerts van Blokland, and R. R. Negenborn, "Dynamic and Stochastic Shipment Matching Problem in Multimodal Transportation," *Transportation Research Record*, vol. 2674, pp. 262–273, 2 2020.

[14] W. Guo, B. Atasoy, and R. R. Negenborn, "Global synchromodal shipment matching problem with dynamic and stochastic travel times: a reinforcement learning approach," *Annals of Operations Research*, 2022.

[15] T. Ambra, A. Caris, and C. Macharis, "Should I stay or should I Go? Assessing intermodal and synchromodal resilience from a decentralized perspective," *Sustainability (Switzerland)*, vol. 11, no. 6, 2019.

[16] A. Di Febbraro, N. Sacco, and M. Saeednia, "An agent-based framework for cooperative planning of intermodal freight transport chains," *Transportation Research Part C: Emerging Technologies*, vol. 64, pp. 72–85, 3 2016.

[17] M. Hrušovský, E. Demir, W. Jammernegg, and T. Van Woensel, "Real-time disruption management approach for intermodal freight transportation," *Journal of Cleaner Production*, vol. 280, 1 2021.

[18] Y. Zhang, R. R. Negenborn, and B. Atasoy, "Synchromodal freight transport re-planning under service time uncertainty: An online model-assisted reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 156, 11 2023.

[19] Y. Zhang, W. Guo, R. R. Negenborn, and B. Atasoy, "Synchromodal transport planning with flexible services: Mathematical model and heuristic algorithm," *Transportation Research Part C: Emerging Technologies*, vol. 140, 7 2022.

[20] X. Wang, "Optimal allocation of limited and random network resources to discrete stochastic demands for standardized cargo transportation networks," *Transportation Research Part B: Methodological*, vol. 91, pp. 310–331, 9 2016.

[21] P. Wide, L. K. Kalahasthi, and V. Roso, "Efficiency effects of information on operational disruption management in port hinterland freight transport: simulation of a Swedish dry port case," *International Journal of Logistics Research and Applications*, 2022.

[22] C. W. Palmqvist, A. Lind, and V. Ahlqvist, "How and Why Freight Trains Deviate From the Timetable: Evidence From Sweden," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 210–221, 2022.

[23] A. Barkley and K. Mcleod, "Congestion and consolidation: An empirical study of a barge shipping merger," *Regional Science and Urban Economics*, vol. 93, 3 2022.

[24] PoR, "Barge Performance Monitor," 2024.

[25] C. van Dorsser, F. Vinke, R. Hekkenberg, and M. van Koningsveld, "The effect of low water on loading capacity of inland ships," *European Journal of Transport and Infrastructure Research*, vol. 20, no. 3, pp. 47–70, 2020.

[26] CCNR, "WATER LEVELS AND AVAILABLE VESSELS' DRAUGHT AT GAUGING STATIONS ON RHINE AND DANUBE," 2020.

[27] Statista, "Average monthly delays for late container vessel arrivals worldwide from January 2019 to July 2022," 2022.

[28] R. S. Sutton and A. G. Barto, *Reinforcement learning : an introduction*. 2018.

[29] S. Filom and S. N. Razavi, "Decarbonization through modal shift using a synchromodal platform: A case study in the Great Lakes," 2023.

[30] W. Guo, B. Atasoy, W. B. van Blokland, and R. R. Negenborn, "Global synchromodal transport with dynamic and stochastic shipment matching," *Transportation Research Part E: Logistics and Transportation Review*, vol. 152, 8 2021.

B Optimization Model Constraints

Constraints for the complete optimization model used for the optimization module are as follows:

$$y_r^t \leq \sum_{s \in S_{o_r}^+} x_{rs}^t, \quad \forall r \in R^t, \tag{B.1}$$

$$y_r^t \leq \sum_{s \in S_{d_r}^-} x_{rs}^t, \quad \forall r \in R^t \tag{B.2}$$

$$\sum_{s \in S_{o_r}^+} x_{rs}^t \leq 1, \quad \forall r \in R^t \tag{B.3}$$

$$\sum_{s \in S_{d_r}^-} x_{rs}^t \leq 1, \quad \forall r \in R^t \tag{B.4}$$

$$\sum_{s \in S_{o_r}^-} x_{rs}^t \leq 0, \quad \forall r \in R^t \tag{B.5}$$

$$\sum_{s \in S_{d_r}^+} x_{rs}^t \leq 0, \quad \forall r \in R^t \tag{B.6}$$

$$\sum_{s \in S_i^+} x_{r,s}^t \leq 1, \quad \forall r \in R^t, i \in I \setminus \{o_r, d_r\} \tag{B.7}$$

$$\sum_{s \in S_i^-} x_{r,s}^t \leq 1, \quad \forall r \in R^t, i \in I \setminus \{o_r, d_r\} \tag{B.8}$$

$$\sum_{s \in S_i^+} x_{r,s}^t = \sum_{s \in S_i^-} x_{r,s}^t, \quad \forall r \in R^t, i \in I \setminus \{o_r, d_r\} \tag{B.9}$$

$$\sum_{r \in R^t} x_{r,s}^t u_r \leq U_s^t, \quad \forall s \in S \tag{B.10}$$

$$D_s + B(1 - x_{rs}^t) \geq a_r + f_i^L u_r, \quad \forall r \in R^t, s \in S \tag{B.11}$$

$$z_{rsp}^t \leq x_{rs}^t \quad \forall r \in R^t, s \in S, p \in S \tag{B.12}$$

$$z_{rsp}^t \leq x_{rp}^t \quad \forall r \in R^t, s \in S, p \in S \tag{B.13}$$

$$z_{rsp}^t \geq x_{rp}^t + x_{rs}^t - 1 \quad \forall r \in R^t, s \in S, p \in S \tag{B.14}$$

$$D_s + t_s + 2f_i^L \leq D_p + B(1 - z_{rsp}^t), \quad \forall r \in R^t, s \in S^{\text{marine}} \cup S^{\text{rail}} \cup S^{\text{truck}}, p \in S, i \in I \setminus \{o_r, d_r\} \tag{B.15}$$

$$c^L_{ri} = \sum_{s \in S^+_i} c^L_i x^t_{r,s}, \quad \forall r \in R^t, i = o_r \tag{B.16}$$

$$c^L_{ri} = \sum_{s \in S^-_i} c^L_i x^t_{r,s}, \quad \forall r \in R^t, i = d_r \tag{B.17}$$

$$c^T_{ri} = \sum_{s \in S^+_i} \sum_{p \in S^-_i} c^T_i z^t_{rsp}, \quad \forall r \in R^t, i \in I \setminus \{o_r, d_r\} \tag{B.18}$$

$$\tilde{S}^s_{ri} = max(0, (D_s - a_r - f^L_i u_r).x^t_{rs}) \quad \forall r \in R^t, s \in S, i = o_r \tag{B.19}$$

$$\tilde{S}^p_{ri} = max(0, (D_{rp} - D_{rs} - t_s - 2f^L_i).z^t_{rsp}) \quad \forall r \in R^t, s \in S^{\text{marine}} \cup S^{\text{rail}} \cup S^{\text{truck}}, p \in S, i \in I \setminus \{o_r, d_r\} \tag{B.20}$$

$$\tilde{S}^s_{ri} = max(0, (e_r - D_s - t_s - f^L_i).x^t_{rs}) \quad \forall r \in R^t, s \in S^{\text{marine}} \cup S^{\text{rail}} \cup S^{\text{truck}}, i = d_r \tag{B.21}$$

$$\tilde{d}^s_{ri} = max(0, (A_s + f^L_i - e_r).x^t_{rs}) \quad \forall r \in R^t, s \in S^{\text{marine}} \cup S^{\text{rail}} \cup S^{\text{truck}}, i = d_r \tag{B.22}$$

Constraints B.1 and B.2 guarantee the platform accepts requests only if there are available services departing from the request's origin and arriving at the request's destination, respectively. Constraints B.3 and B.4 check that at most one service matches with request $r \in R^t$ with the same origin and destination, respectively. Constraints B.5 to B.8 are responsible for eliminating subtours from the solution in which constraints B.6 and B.7 are designed to remove the subtours from shipment request origin and destination, respectively. Moreover, constraints B.8 and B.9 are designed so that each itinerary must have one origin and destination. Constraint B.9 ensures flow conservation at transshipment terminals. Constraint B.10 ensures that the total amount of containers matched with service $s \in S$ does not surpass the service available capacity at decision epoch $t \in T$. Constraint B.11 guarantees that the departure time of service minus loading time (based on container volumes) should be earlier than the request release time, for matched requests and services. B is a large number to make the constraint valid when a request is matched with a service (i.e. $x^t_{rs} = 1$).

Constraints B.12-B.14 maintain the logic of the transshipment problem through binary variables x^t_{rp} and z^t_{rsp}. The first one denotes the potential service $p \in S$ that could be matched with service $s \in S$ at d_s where the destination of the transshipment service is similar to the request destination (i.e, $d_p = d_r$). The binary variable z^t_{rsp} equals 1 if and only if $x^t_{rs} = 1$ and $x^t_{rp} = 1$ indicating that the transshipment occurred between service s and p. Constraint B.15 ensures the temporal feasibility of transshipment at the intermediate terminal. Constraints B.16 and B.17 calculate the loading and unloading cost at the origin and destination of the request, respectively. Constraint B.18 determines the transshipment cost including both loading and unloading costs at the transshipment terminal. Constraints B.19-B.21 are designed to calculate storage cost. Constraint B.19 determines the storage cost at the origin terminal. Constraint B.20 computes the storage cost at the transshipment terminal, and constraint B.21 computes the storage cost at the destination terminal. Constraint B.22 determines the delay time at the destination intermodal terminal of the request.

C Case Study Datasets

Table C.1: Barge Line Distance Network

N	Delta	Euromax	HOME	Moerdijk	Venlo	Duisburg	Willebroek	Neuss	Dortmund	Nuremberg
Delta	0	15	37.5	75	195	240	165	255	-	-
Euromax	15	0	-	82.5	202.5	247.5	172.5	262.5	-	-
HOME	37.5	-	0	45	165	232.5	157.5	-	-	-
Moerdijk	75	82.5	45	0	150	180	-	-	-	-
Venlo	195	202.5	165	150	0	-	-	-	-	-
Duisburg	240	247.5	232.5	180	-	0	-	37.5	-	-
Willebroek	165	172.5	157.5	-	-	-	0	-	-	-
Neuss	255	262.5	-	-	-	37.5	-	0	-	-
Dortmund	-	-	-	-	-	-	-	-	0	-
Nuremberg	-	-	-	-	-	-	-	-	-	0

Table C.2: Train Line Distance Network

N	Delta	Euromax	HOME	Moerdijk	Venlo	Duisburg	Willebroek	Neuss	Dortmund	Nuremberg
Delta	0	-	-	-	180	270	-	225	315	675
Euromax	-	0	-	-	202.5	292.5	-	247.5	337.5	697.5
HOME	-	-	0	-	157.5	247.5	-	-	-	-
Moerdijk	-	-	-	0	135	180	-	-	-	-
Venlo	180	202.5	157.5	135	0	-	-	67.5	112.5	495
Duisburg	270	292.5	247.5	180	-	0	-	-	67.5	450
Willebroek	-	-	-	-	-	-	0	-	-	-
Neuss	225	247.5	-	-	67.5	-	-	0	-	-
Dortmund	315	337.5	-	-	112.5	67.5	-	-	0	-
Nuremberg	675	697.5	-	-	495	450	-	-	-	0

Table C.3: Truck Line Distance Network

N	Delta	Euromax	HOME	Moerdijk	Venlo	Duisburg	Willebroek	Neuss	Dortmund	Nuremberg
Delta	0	15	37.5	75	195	240	150	262.5	300	675
Euromax	15	0	45	90	210	247.5	165	270	315	712.5
HOME	37.5	45	0	45	172.5	202.5	112.5	225	255	660
Moerdijk	75	90	45	0	135	180	95	-	-	-
Venlo	195	210	172.5	135	0	60	-	67.5	112.5	495
Duisburg	240	247.5	202.5	180	60	0	-	37.5	67.5	450
Willebroek	150	165	112.5	95	-	-	0	-	-	-
Neuss	262.5	270	225	-	67.5	37.5	-	0	-	-
Dortmund	300	315	255	-	112.5	67.5	-	-	0	-
Nuremberg	675	712.5	660	-	495	450	-	-	-	0

Table C.4: Fixed Service Lines

K	Origin	Destinatin	Departure	Arrival	Travel Time	Capacity	Speed
Barge1	Delta	Euromax	53	55	2	160	15
Barge2	Delta	HOME	53	56.5	3.5	160	15
Barge3	Delta	Moerdijk	3	9	6	160	15
Barge4	Delta	Moerdijk	15	21	6	160	15
Barge5	Delta	Moerdijk	27	33	6	160	15
Barge6	Delta	Moerdijk	39	45	6	160	15
Barge7	Delta	Moerdijk	51	57	6	160	15
Barge8	Delta	Moerdijk	63	69	6	160	15
Barge9	Delta	Moerdijk	75	81	6	160	15
Barge10	Delta	Moerdijk	87	93	6	160	15
Barge11	Delta	Moerdijk	99	105	6	160	15
Barge12	Delta	Moerdijk	111	117	6	160	15
Barge13	Delta	Moerdijk	123	129	6	160	15
Barge14	Delta	Moerdijk	135	141	6	160	15
Barge15	Delta	Moerdijk	147	153	6	160	15
Barge16	Delta	Moerdijk	159	165	6	160	15
Barge17	Delta	Venlo	12	26	14	160	15
Barge18	Delta	Venlo	18	32	14	160	15
Barge19	Delta	Venlo	36	50	14	160	15
Barge20	Delta	Venlo	42	56	14	160	15
Barge21	Delta	Venlo	60	74	14	160	15
Barge22	Delta	Venlo	66	80	14	160	15
Barge23	Delta	Venlo	90	104	14	160	15
Barge24	Delta	Venlo	96	110	14	160	15
Barge25	Delta	Venlo	120	134	14	160	15
Barge26	Delta	Duisburg	82	99	17	160	15
Barge27	Delta	Duisburg	102	119	17	160	15
Barge28	Delta	Willebroek	68	80	12	160	15
Barge29	Delta	Willebroek	98	110	12	160	15
Barge30	Delta	Willebroek	146	158	12	160	15
Barge31	Delta	Neuss	80	98	18	160	15
Barge32	Euromax	Moerdijk	3	9.5	6.5	160	15
Barge33	Euromax	Moerdijk	51	57.5	6.5	160	15
Barge34	Euromax	Moerdijk	99	105.5	6.5	160	15
Barge35	Euromax	Venlo	27	41.5	14.5	160	15
Barge36	Euromax	Venlo	75	89.5	14.5	160	15
Barge37	Euromax	Duisburg	103	120.5	17.5	160	15
Barge38	Euromax	Willebroek	112	124.5	12.5	160	15
Barge39	Euromax	Neuss	66	84.5	18.5	160	15
Barge40	HOME	Moerdijk	5	9	4	160	15
Barge41	HOME	Moerdijk	53	57	4	160	15
Barge42	HOME	Moerdijk	101	105	4	160	15
Barge43	HOME	Venlo	99	111	12	160	15
Barge44	HOME	Venlo	126	138	12	160	15
Barge45	HOME	Duisburg	51	67.5	16.5	160	15
Barge46	HOME	Willebroek	20	31.5	11.5	160	15
Barge47	Moerdijk	Venlo	95	106	11	160	15
Barge48	Moerdijk	Duisburg	71	84	13	160	15
Barge49	Duisburg	Neuss	120	123.5	3.5	160	15
Train1	Delta	Venlo	16	21	5	90	45
Train2	Delta	Venlo	40	45	5	90	45
Train3	Delta	Venlo	9	14	5	90	45
Train4	Delta	Venlo	33	38	5	90	45

Table C.4 continued from previous page

K	Origin	Destinatin	Departure	Arrival	Travel Time	Capacity	Speed
Train5	Delta	Venlo	57	62	5	90	45
Train6	Delta	Venlo	81	86	5	90	45
Train7	Delta	Venlo	105	110	5	90	45
Train8	Delta	Venlo	129	134	5	90	45
Train9	Delta	Duisburg	41	48	7	90	45
Train10	Delta	Duisburg	75	82	7	90	45
Train11	Delta	Duisburg	99	106	7	90	45
Train12	Delta	Duisburg	113	120	7	90	45
Train13	Delta	Neuss	110	116	6	90	45
Train14	Delta	Dortmund	88	96	8	90	45
Train15	Delta	Nuremberg	51	67	16	90	45
Train16	Delta	Nuremberg	99	115	16	90	45
Train17	Euromax	Venlo	78	83.5	5.5	90	45
Train18	Euromax	Venlo	102	107.5	5.5	90	45
Train19	Euromax	Duisburg	75	82.5	7.5	90	45
Train20	Euromax	Duisburg	99	106.5	7.5	90	45
Train21	Euromax	Neuss	77	83.5	6.5	90	45
Train22	Euromax	Dortmund	78	86.5	8.5	90	45
Train23	Euromax	Nuremberg	79	95.5	16.5	90	45
Train24	HOME	Venlo	86	90.5	4.5	90	45
Train25	HOME	Duisburg	27	33.5	6.5	90	45
Train26	HOME	Duisburg	75	81.5	6.5	90	45
Train27	Moerdijk	Venlo	75	79	4	90	45
Train28	Moerdijk	Duisburg	77	82	5	90	45
Train29	Venlo	Neuss	112	114.5	2.5	90	45
Train30	Venlo	Dortmund	113	116.5	3.5	90	45
Train31	Venlo	Nuremberg	114	126	12	90	45
Train32	Duisburg	Dortmund	121	123.5	2.5	90	45
Train33	Duisburg	Nuremberg	122	133	11	90	45

Table C.5: Truck Service Lines

K	Origin	Destination	Departure	Arrival	Travel Time	Capacity	Speed
Truck1	Delta	Euromax	99999	99999	0.2	99999	75
Truck2	Delta	HOME	99999	99999	0.5	99999	75
Truck3	Delta	Moerdijk	99999	99999	1	99999	75
Truck4	Delta	Venlo	99999	99999	2.6	99999	75
Truck5	Delta	Duisburg	99999	99999	3.2	99999	75
Truck6	Delta	Willebroek	99999	99999	2	99999	75
Truck7	Delta	Neuss	99999	99999	3.5	99999	75
Truck8	Delta	Dortmund	99999	99999	4	99999	75
Truck9	Delta	Nuremberg	99999	99999	9	99999	75
Truck10	Euromax	HOME	99999	99999	0.6	99999	75
Truck11	Euromax	Moerdijk	99999	99999	1.2	99999	75
Truck12	Euromax	Venlo	99999	99999	2.8	99999	75
Truck13	Euromax	Duisburg	99999	99999	3.3	99999	75
Truck14	Euromax	Willebroek	99999	99999	2.2	99999	75
Truck15	Euromax	Neuss	99999	99999	3.6	99999	75
Truck16	Euromax	Dortmund	99999	99999	4.2	99999	75
Truck17	Euromax	Nuremberg	99999	99999	9.5	99999	75
Truck18	HOME	Moerdijk	99999	99999	0.6	99999	75
Truck19	HOME	Venlo	99999	99999	2.3	99999	75
Truck20	HOME	Duisburg	99999	99999	2.7	99999	75

Table C.5 continued from previous page

K	Origin	Destination	Departure	Arrival	Travel Time	Capacity	Speed
Truck21	HOME	Willebroek	99999	99999	1.5	99999	75
Truck22	HOME	Neuss	99999	99999	3	99999	75
Truck23	HOME	Dortmund	99999	99999	3.4	99999	75
Truck24	HOME	Nuremberg	99999	99999	8.8	99999	75
Truck25	Moerdijk	Venlo	99999	99999	1.8	99999	75
Truck26	Moerdijk	Duisburg	99999	99999	2.4	99999	75
Truck27	Moerdijk	Willebroek	99999	99999	1.266667	99999	75
Truck28	Venlo	Duisburg	99999	99999	0.8	99999	75
Truck29	Venlo	Neuss	99999	99999	0.9	99999	75
Truck30	Venlo	Dortmund	99999	99999	1.5	99999	75
Truck31	Venlo	Nuremberg	99999	99999	6.6	99999	75
Truck32	Duisburg	Neuss	99999	99999	0.5	99999	75
Truck33	Duisburg	Dortmund	99999	99999	0.9	99999	75
Truck34	Duisburg	Nuremberg	99999	99999	6	99999	75

Table C.6: Request Dataset

Request ID	Origin	Destination	Release Time	Due Time	Volume	Assigned Mode	Announcement Time
Request1	1	8	3	76	7	0	0
Request2	1	9	0	112	29	0	0
Request3	1	4	9	122	11	0	0
Request4	1	9	44	166	22	0	0
Request5	1	8	74	168	24	0	0
Request6	1	6	94	151	28	0	0
Request7	2	8	13	90	28	0	0
Request8	2	9	40	152	12	0	0
Request9	2	8	12	143	27	0	0
Request10	1	9	59	149	14	0	0
Request11	1	6	3	151	15	0	0
Request12	1	9	0	88	9	0	0
Request13	2	8	70	162	6	0	0
Request14	2	6	73	191	27	0	0
Request15	1	5	2	153	15	0	0
Request16	3	6	94	168	22	0	0
Request17	1	9	96	217	11	0	0
Request18	1	5	75	179	21	0	0
Request19	3	10	39	199	7	0	0
Request20	2	10	42	102	5	0	0
Request21	1	9	59	204	15	0	0
Request22	2	10	15	169	28	0	0
Request23	3	8	30	115	28	0	0
Request24	1	9	7	137	5	0	0
Request25	1	4	32	174	10	0	0
Request26	3	9	55	217	18	0	0
Request27	1	5	47	138	17	0	0
Request28	1	6	80	186	26	0	0
Request29	1	10	43	159	21	0	0
Request30	1	5	74	215	15	0	0
Request31	1	9	61	141	10	0	0
Request32	1	8	55	114	27	0	0
Request33	1	9	11	63	13	0	0
Request34	3	8	7	79	18	0	0

Table C.6 continued from previous page

Request ID	Origin	Destination	Release Time	Due Time	Volume	Assigned Mode	Announcement Time
Request35	1	9	28	81	17	0	0
Request36	1	8	8	121	28	0	0
Request37	1	8	7	60	30	0	0
Request38	1	6	64	135	22	0	0
Request39	1	9	9	68	9	0	0
Request40	1	5	62	223	22	0	0
Request41	1	10	47	202	28	0	0
Request42	3	10	25	133	15	0	0
Request43	1	10	0	84	16	0	0
Request44	1	8	76	182	6	0	0
Request45	1	5	92	231	9	0	0
Request46	1	8	20	144	25	0	0
Request47	1	9	87	152	29	0	0
Request48	3	8	29	147	17	0	0
Request49	1	9	82	228	5	0	0
Request50	2	9	14	111	14	0	0
Request51	2	8	25	121	21	0	0
Request52	1	10	28	160	22	0	0
Request53	3	10	41	206	23	0	0
Request54	1	5	45	175	21	0	0
Request55	1	9	13	74	23	0	0
Request56	1	10	59	189	5	0	0
Request57	2	10	45	155	17	0	0
Request58	2	6	50	133	9	0	0
Request59	1	9	78	181	27	0	0
Request60	2	9	10	74	19	0	0
Request61	1	5	37	147	24	0	0
Request62	3	10	3	73	12	0	0
Request63	1	5	74	196	7	0	0
Request64	1	8	78	154	22	0	0
Request65	1	10	25	95	15	0	0
Request66	1	9	86	162	23	0	0
Request67	1	8	238	372	30	0	144
Request68	2	8	181	347	23	0	144
Request69	2	10	166	231	24	0	144
Request70	1	8	199	308	9	0	144
Request71	1	9	184	234	16	0	144
Request72	3	5	208	260	25	0	144
Request73	1	5	218	277	14	0	144
Request74	2	10	172	263	25	0	144
Request75	1	8	198	295	27	0	144
Request76	1	10	230	311	10	0	144
Request77	2	10	197	362	30	0	144
Request78	1	9	147	314	26	0	144
Request79	1	4	228	388	20	0	144
Request80	1	6	154	260	14	0	144
Request81	2	8	198	346	30	0	144
Request82	1	9	167	222	30	0	144
Request83	1	9	157	272	28	0	144
Request84	1	9	223	273	8	0	144
Request85	1	8	203	292	10	0	144
Request86	3	10	211	314	17	0	144
Request87	1	8	203	279	12	0	144

Table C.6 continued from previous page

Request ID	Origin	Destination	Release Time	Due Time	Volume	Assigned Mode	Announcement Time
Request88	1	10	220	378	21	0	144
Request89	2	9	181	331	10	0	144
Request90	1	8	203	256	20	0	144
Request91	1	8	180	315	6	0	144
Request92	1	10	229	296	9	0	144
Request93	1	5	177	292	27	0	144
Request94	2	10	162	330	13	0	144
Request95	1	8	183	282	9	0	144
Request96	2	9	216	285	28	0	144
Request97	3	9	214	340	28	0	144
Request98	1	4	189	333	8	0	144
Request99	2	4	155	310	11	0	144
Request100	2	5	180	327	13	0	144
Request101	1	8	238	342	20	0	144
Request102	1	9	174	315	24	0	144
Request103	2	6	188	262	8	0	144
Request104	2	5	178	275	8	0	144
Request105	1	8	152	226	25	0	144
Request106	2	10	189	299	12	0	144
Request107	1	7	179	285	23	0	144
Request108	1	10	197	275	15	0	144
Request109	1	6	198	259	14	0	144
Request110	1	9	229	312	9	0	144
Request111	1	9	228	385	30	0	144
Request112	2	9	239	366	10	0	144
Request113	1	9	215	280	10	0	144
Request114	3	8	187	302	9	0	144
Request115	1	10	203	353	9	0	144
Request116	1	8	208	277	16	0	144
Request117	2	5	186	336	22	0	144
Request118	1	10	172	269	15	0	144
Request119	1	10	155	204	26	0	144
Request120	2	9	216	288	5	0	144
Request121	3	8	230	311	14	0	144
Request122	1	6	192	318	13	0	144
Request123	3	8	201	361	8	0	144
Request124	1	10	190	342	6	0	144
Request125	2	9	214	376	9	0	144
Request126	1	9	197	359	23	0	144
Request127	1	6	172	291	28	0	144
Request128	3	10	208	299	17	0	144
Request129	2	10	240	360	9	0	144
Request130	1	10	150	273	14	0	144
Request131	2	9	226	347	26	0	144
Request132	1	10	222	313	11	0	144
Request133	2	10	356	487	19	0	312
Request134	3	10	397	548	25	0	312
Request135	1	10	326	415	8	0	312
Request136	1	8	386	440	8	0	312
Request137	1	8	338	408	27	0	312
Request138	3	10	338	401	26	0	312
Request139	2	7	404	551	9	0	312
Request140	1	8	361	519	10	0	312

Table C.6 continued from previous page

Request ID	Origin	Destination	Release Time	Due Time	Volume	Assigned Mode	Announcement Time
Request141	1	9	331	479	17	0	312
Request142	1	8	324	492	15	0	312
Request143	3	10	333	486	26	0	312
Request144	1	8	378	477	24	0	312
Request145	1	8	400	546	5	0	312
Request146	1	8	388	509	27	0	312
Request147	1	6	324	399	29	0	312
Request148	1	8	342	465	14	0	312
Request149	1	9	354	504	13	0	312
Request150	1	9	361	506	6	0	312
Request151	3	6	323	484	11	0	312
Request152	1	5	327	392	28	0	312
Request153	1	6	385	513	7	0	312
Request154	1	9	360	489	12	0	312
Request155	1	5	387	458	29	0	312
Request156	1	10	401	468	14	0	312
Request157	1	8	358	504	5	0	312
Request158	1	8	360	488	13	0	312
Request159	1	6	392	478	24	0	312
Request160	1	5	346	441	25	0	312
Request161	2	8	335	395	19	0	312
Request162	1	5	387	466	27	0	312
Request163	3	6	381	518	25	0	312
Request164	1	8	354	422	5	0	312
Request165	3	8	359	519	7	0	312
Request166	2	6	343	471	14	0	312
Request167	1	9	354	499	21	0	312
Request168	1	5	316	443	30	0	312
Request169	1	9	345	422	28	0	312
Request170	3	6	328	442	17	0	312
Request171	1	10	403	533	25	0	312
Request172	1	9	331	422	24	0	312
Request173	1	8	356	472	22	0	312
Request174	1	5	395	474	12	0	312
Request175	2	10	385	506	8	0	312
Request176	3	8	361	471	7	0	312
Request177	1	5	351	506	27	0	312
Request178	1	8	355	503	25	0	312
Request179	1	8	328	390	25	0	312
Request180	3	9	348	485	10	0	312
Request181	3	9	323	457	17	0	312
Request182	1	9	356	411	16	0	312
Request183	1	8	323	487	16	0	312
Request184	3	9	360	517	12	0	312
Request185	1	5	372	457	20	0	312
Request186	2	10	389	466	20	0	312
Request187	1	8	377	430	23	0	312
Request188	3	7	349	416	11	0	312
Request189	1	8	366	443	24	0	312
Request190	1	6	321	401	10	0	312
Request191	1	6	391	502	11	0	312
Request192	1	8	348	516	27	0	312
Request193	2	9	366	433	7	0	312

Table C.6 continued from previous page

Request ID	Origin	Destination	Release Time	Due Time	Volume	Assigned Mode	Announcement Time
Request194	2	10	352	458	17	0	312
Request195	1	9	316	383	16	0	312
Request196	1	10	326	428	30	0	312
Request197	1	9	329	461	9	0	312
Request198	3	6	335	384	30	0	312
Request199	2	10	490	594	5	0	480
Request200	1	5	546	661	11	0	480