Reducing port downtime: A Deep Q-Network approach to repair sequence

A Port of Rotterdam case study

Tanja de Bruin

Delft University of Technology

TU Delft

Reducing port downtime: A Deep Q-Network approach to repair sequence

A Port of Rotterdam case study

Student Name	Student Number
Tanja de Bruin	6062687

Faculty	Civil Engineering & Geosciences
Thesis Supervisors	Patrick Stokkink
	Yongqiu Zhu
	Srijith Balakrishnan
Project Duration	March 2025 - September 2025

Contents

1
Summary

Maritime trade moves over 80% of global goods, making ports critical in global logistics. However, ports face growing risks from disruptions such as piracy, geopolitical events, and extreme weather linked to climate change. When ports are partially of complete disrupted, ships backlogs and can effected both local and global trade. Fast and efficient recovery is essential, requiring optimal repair sequences for disrupted waterways in a port.

Existing research has shown that approaches such as greedy algorithms have been applied in a variety of different network successfully. The same can be said about reinforcement learning, which shows promising results in network disruption recovery. This research investigates whether a reinforcement learning algorithm, specifically Deep Q-Network (DQN), can be implemented to optimize recovery in port waterway networks.

An agent-based simulation of the Port of Rotterdam was developed in AnyLogic, representing the operational functioning of the port. Disruptions in the network affect waterway sections, resulting in unreachable terminals. Ships will start queueing when no suitable terminals are available, leading to waiting times for ships. The DQN agent interacts with the simulation to exchange the state of the environment and the action that needs to be performed. With the aim of learning a policy that is able to find an optimal repair sequence, depending on the disrupted waterway sections and arriving ships.

The network of the port of Rotterdam consisted of 51 waterway sections and 54 terminals and has a tree like structure. Training was performed with scenarios consisting of 7 disrupted waterway sections each time. A total of 3000 episodes were run to train the DQN algorithm. With 102 unique disruption scenarios, out of the 583.506.504.000 different scenarios possible. Testing was performed on 20 new scenarios, each run 100 times to adjust for uncertainty generated by ship arrival. The average reward values of the DQN and GA are compared to decide the effectiveness of the DQN algorithm.

The results show that the DQN outperformed the Greedy algorithm in scenarios where disrupted waterway sections had high network priority. However, in low summed priority scenarios, the Greedy algorithm sometimes matched or exceeded the DQN results. This can be explained by a lack of useful feedback received for the DQN agent, in low summed priority scenarios.

The conclusion is that DQN can be effectively contribute to optimal repair sequencing in high impact disruption. The approach is adaptable for different port, or can be adjusted for different types of networks. For usage in the industry, the model will have to be extended and based on real life data.

2
Introduction

Maritime trade is the backbone of logistics these days, with shipping over 80% of all products worldwide. With more and more people relying on functional maritime transport, the pressure is on for maintaining a functioning global and local trade network. The crucial parts of the maritime network are the ports, as these are under threat of different types of disruptions.

In recent years the maritime trade has been disrupted by different causes, such as piracy (Shepard and Pratson, 2020), extreme weather events and geopolitical causes (X. Li et al., 2024). In the past years disruptions with regard to extreme weather events have increased, Chand et al. (2022) stated that this is due to climate change. The disruptions caused by extreme weather events can range from shut-down of the port for a few hours to recovery taking multiple months.

The (partial) closure of a port for an extended period can create backlog of ships, influencing the operations of close-by ports and impacts worldwide maritime trade. Demand for maritime transport does not decrease when a port becomes disrupted, leading to more backlog at the disrupted port and spill-over to surrounding ports (L. Li et al., 2022). To benefit local and global trade, recovering a port in the shortest time possible is the most beneficial but also extremely costly (Wan et al., 2021). Port authorities need to adhere to budget and operational constraints when recovering from a disrupting event. To adhere to the limitations, a disruption specific repair sequence is needed. The aim of the port authorities will be to be fully operational in the shortest period of time, while minimizing the lost revenue or costs and adhering to operational constraints.

Repair sequence optimization strategies have been proposed in different industries regarding networks with disrupted waterway sections. Although multiple studies have been reviewed, no article was found using reinforcement learning to optimize the repair sequence of waterway sections within a port. Different methods have been used to optimize repair sequences, the main difference between these methods is the focus of the objective. Some papers focus on repairing major waterway sections first providing a possibility of transport as soon as possible, working top down based on a priority value given to disrupted infrastructure paths (Wan et al., 2021). Other papers focus on a global objective such as cost or time effective repair of the complete network (Su et al., 2022). Both types of objectives can be beneficial depending on the goal or the one in charge. Companies themselves could prioritize their own infrastructure for economical gain, while port authorities might choose for the global optimum.

This research proposes an agent-based reinforcement learning model aiming to minimize the overall waiting time of vessels arriving in a disrupted port. To reach an optimal solution the repair sequence of disrupted waterway sections will be used to benefit the objective of the model. To test the proposed model a case study for the Port of Rotterdam is performed using the agent-based simulation software AnyLogic. The proposed reinforcement learning model

is benchmarked against a Greedy algorithm.

In the following chapter the research questions will be stated on which this study is built. In the chapter thereafter, Chapter 4, a literature review will be given on the relevant topics. Chapter 5 will elaborate the methods used in this research, including the reinforcement learning algorithm, the agent-based simulation and the benchmark Greedy algorithm. In Chapter 6 the case study will be introduces. Followed up by chapter 7, which explains how the training of the RL agent took place and shows the results of testing both algorithms. These results are concluded and discussed in Chapter 8.

3
Research questions

The objective of this research is to provide a strategy on repair sequence of disrupted waterways to minimize overall waiting time for ships in a port, effected by extreme weather events. This is achieved by answering the following main research question:

How can reinforcement learning algorithms be employed to optimize the repair sequence of disrupted waterways within a port to minimize overall waiting time for ships?

The main research question will be answered by answering the following sub-questions:

1. What is the current state of the art with regards to repair sequence strategies?
2. Which state-space, actions and rewards should be defined for the reinforcement learning model?
3. What evaluation measures should be used to measure effectiveness of the model?
4. How do reinforcement learning approaches compare to traditional rule-based models in network recovery?

4
Literature review

4.1. Port disruptions

Disruption in the global maritime network can have multiple causes, X. Li et al. (2024) make a distinction based on the source of a disruption. They define two categories: natural and man-made disruptions. Man-made disruptions are linked to geo-political conflicts, maritime piracy and technology failure. This type of disruption does not occur frequently and often does not need physical repair. In contrary, natural disruptions are linked to natural disasters and weather hazards, which are out of control and often result in physical damage.

Lam and Lassa (2017) state that natural disasters are mainly classified as low-probability and high-impact disruptive events, based on the intensity of the event the effect can be catastrophic. Izaguirre et al. (2020) discovered that of the disruptions caused by natural causes, tropical cyclones impose the most risk for ports and consequently the global maritime network. Verschuur, Koks, Li, and Hall (2023) assessed 3140 ports globally and found that over 86% is exposed to more than three natural hazards. The categories of natural disruptions used in the study are: tropical cyclones, earthquakes, river flooding, pluvial flooding and coastal flooding. It was found that over 80% of all ports are prone to a type of flooding, making it the most common disruption threat for ports. Especially for ports in Western and Northern Europe flooding is the leading natural hazard.

In the research of Gou and Lam (2018) it is shown that both tsunamis as well as typhoons can cause ships to run aground, drift ashore or sink in port waters, causing risk for other vessels and closing off or damaging waterways and terminals. Chang (2000) investigated the impact of the earthquake in the western part of Japan in 1995, hitting the port of Kobe. Due to the intensity of the earthquake, it resulted in a recovery period of 2 years and an estimated damage cost of 1 trillion Yen. From the 186 berths only 9 where operational after the earthquake, a disruption of around 95%. During the recovery period, ships were diverted to surrounding ports. Priority was given to the container terminals, these make up the biggest part of the port and where of economical importance. After full recovery, part of the before handled cargo had shifted to different ports, and the port of Kobe lowered to number 17 on the world ranking instead of 6. Large scale disruption of ports can have a major influence in the short term as well as the long term on the port itself and surrounding ports.

Wan et al. (2017) defined resilience for a transport system as the absorptive capacity to withstand disruptions, the capacity to adapt with the aim of keeping structure and function during disruption, and the capacity to recover to acceptable performance levels. In post-disaster networks, resilience is often used to determine the added value of a recovery strategy, although different papers use different definitions of resilience. C. Li et al. (2025) concluded that in the context of maritime systems the core principle in all definitions of resilience, is the ability of a network to recover to normal operations after a disruptive event.

4.2. Network recovery strategies

Proag (2014) states that resilience can be divided into two forms: the ability to advert a threat of disruption and the ability to recover from a disruption. In light of network recovery the second form of resilience is most important. The recoverability of a port depends on the intensity of the disrupting event and the resilience of the port. This makes that the recoverability is very port specific (Verschuur, Koks, and Hall, 2023).

Network recovery has been extensive researched in different types of networks. Different industries where network recovery is proposed are among others: electricity networks (S. Wang et al., 2004, Çağnan et al., 2006), water distribution networks (Paez et al., 2020, Fan et al., 2022), road networks (Zhang et al., 2017, Fan et al., 2023, Shen et al., 2022, Su et al., 2022) and global shipping (Wan et al., 2021).

Paez et al. (2020) concluded that when presented with a network recovery problem, the proposed solutions by experts could be fitted in three categories. Namely, general-purpose meta-heuristics, greedy algorithms and ranking-based prioritizations. Meta-heuristics has an advantage when developing plans beforehand, due to the computational costs the use in real time planning is limited. On the other hand, greedy algorithms have lower computational times and are able to adapt to new information, making them more useful in case of emergency. Ranking-based approaches uses expert knowledge and criteria to quickly recommend a plan of action (Paez et al., 2020).

Wan et al. (2021) researched network recovery strategies in the line shipping network where ports are used as nodes. Four different strategies were proposed and compared on overall resilience. The strategies used are: 1) random recovery strategy, 2) degree centrality-based recovery strategy, 3) closeness centrality-based recovery strategy and 4) betweenness centrality-based recovery strategy. It was found that for the Northwest Pacific region the betweenness centrality-based recovery strategy minimized the total shutdown time and reduces ship delays as much as possible. The closeness centrality recovery strategy on the other hand, results in the largest resilience index and the least recovery costs.

In the last years the research regarding network recovery shifted from mathematical modelling to machine learning algorithms. Zhang et al. (2017) used a semi-supervised learning method called active learning. This type of machine learning works with both labelled and unlabelled data and uses labels to determine the training data set. The concept of active learning is based on the labelling, if the labelling is done correctly, the training dataset can be smaller and training can be done more efficient. The active learning method was used to develop recovery strategies for road-bridge networks in a post-disaster scenario. The active learning approach was benchmarked against a betweenness-first strategy and a greedy algorithm, overall network resilience was the evaluation measure. Zhang et al. (2017) concluded that active learning outperforms the betweenness-first strategy and the greedy algorithm, and is able to find better recovery sequences.

Multiple papers make use of Q-learning, which is part of reinforcement learning and uses a model-free approach. Su et al. (2022) used a Deep Q-Learning (DQL) algorithm for repair crew scheduling and routing in a post-disaster road network. After experimenting with the algorithm, they concluded that DQL through learning and cooperation is able to improve schedul-

ing and routing of repair crews. Fan et al. (2022) introduced a Graph Convolutional Neural Network-integrated Deep Reinforcement learning (GCN-DRL) model, which is based on assigning Q-values to actions in the model. The model was tested on water distribution networks disrupted by earthquakes and benchmarked against four conventional decision making methods. Including a GA method, a diameter-based repair prioritization method and two greedy search-based strategies. They concluded that the GCN-DRL model gets more advantages with an increase in damaged links. It outperforms the conventional methods, both in terms of performance and computational efficiency. Fan et al. (2023) tested the GCN-DRL model on a road network under earthquake or flooding hazards. The model was benchmarked against a genetic algorithm and prioritization based on graph importance with betweenness centrality. The repair sequences generated by the GCN-DRL model consistently outperformed the other model. Both Fan et al. (2022) and Fan et al. (2023) concluded that the GCN-DRL developed better repair sequences in case of disrupted networks, respectively water distribution and road networks. An other advantage of GCN-DRL is that training takes place before a disruption, so in case of a distribution no additional training is necessary to develop a close to optimal repair sequence.

Sun and Zhang (2020) also used deep Q algorithms, they combined a Deep Q-Network (DQN) with Agent-Based Modelling (ABM). The actions defined by the Markov Decision Process can be performed by the different agents active in the network, yielding rewards that can train the RL model how to perform in different states. With this approach there is direct interaction between the actions proposed by the reinforcement learning algorithm and the environment. The agent can be programmed to have interactive behaviour, resulting in a situation where the action of one agent could influence the next action of a different agent. Sun and Zhang (2020) used this principle to find a recovery strategy for a network with different types of infrastructure, the different types of infrastructure were modelled as their own agent. These agents influence each other. The ABM approach proves itself useful, in comparison with 5 random strategies the agent-based Deep Q-Network always proposed a better solution with a shorter recovery time. ABM simulation is elaborated on in the following section.

4.3. Agent-Based Modelling

Agent-Based Modelling has been extensively used from the early 1970's, when the computational capacity of computers grew strong enough to run these types of models. ABM is not limited to modelling and simulating a predefined script, but agents can make autonomous decisions in real time (Huang et al., 2022). Autonomy, collaboration and reactivity are the main advantages of ABM for modelling behaviours within for example transport networks (F.-Y. Wang, 2005). Huang et al. (2022) state that ABM is unique in integrating heterogenous entities and observing the dynamics and behaviours between agents.

Gueli et al. (2023) used ABM for vessel routing in ports. They focused on the ships as an agent moving through the port, both approaching and leaving manoeuvrers. The model was implemented for a case study of the port of Augusta. Based on the predefined set of rules and observations of the environment, ships were able to navigate safely through the port.

Chen et al. (2004) divided the functionalities of an agent into four modules: 1) effector, sensor and communication module, 2) intention module, 3) motivation module and 4) cognition module. The first module focusses on observing and communicating with the environment. Module 2 balances the short and long term goals of the agent, based on the motivations in

module 3. The last module consists of the knowledge an agent has and will use to base their action on. This gives agents the ability to communicate with other agents as well as interact with the environment and adjust actions based on received input.

Huang et al. (2022) point out that one of the less researched aspects of ABM is linked to module 4, the learning abilities of agents. They state that combining ABM simulations with for example optimization algorithms, is a promising research direction.

4.4. Literature conclusion

It can be concluded that the current state of art concerning repair sequence strategies, is slowly shifting from rule based and optimization algorithms towards reinforcement learning. As the data about disruptions in port waterways is highly limited and real-life testing is not feasible, a simulation can capture the important behaviours and make it possible to test multiple different scenarios. As stated by Huang et al. (2022) Agent-Based Modelling in combination with optimization algorithms is a promising research field. That can benefit a wide variety of networks, including those in ports.

5
Methodology

To answer the remaining sub-questions and in the end the main research question, a variety of methods is needed. First of all the ABM simulation model needed to be built. This process took an extended period of time, due to unfamiliarity with the software as well as limited resources regarding previous work on this subject. The simulation model is built in step, making sure the modelled part functioned as expected, before extending the simulation to the next step. Throughout building and checking if everything worked as it should, a smaller network was used to decrease computational power and run time. To reflect the case study in the best way, data was gathered about the locations and types of terminals in the port. As well as the arrival rates at the port and the distribution of the ships that enter the port. The simulation model is explained in Section 5.1.

After building the simulation model, the reinforcement learning algorithm had to be designed. Based on the findings in Chapter 4, a Deep Q-Network was chosen. The method of Deep Q-Network (DQN) is based on the advances made in training deep neural networks, and was first described by Mnih et al. (2015). DQN is based on the combination of reinforcement learning and deep neural networks, and aims to mimic the way humans learn via rewards and penalties. This learning is done by taking actions and calculating the action-value with the aim to maximize the rewards by selecting the right actions. Mnih et al. (2015) used a deep convolutional neural network for calculating the action-value, via Equation 5.1.

$$Q^*(s,a) = max_\pi E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s, a_t = a, \pi] \tag{5.1}$$

Q^* indicates the network that is used for calculating the policy values. By calculating the maximum sum of rewards (r_t) and discounting every step (t) by γ, the maximum value of a policy ($\pi = P(a|s)$) at state s and action a is given. γ is used to nudge the learning behaviour, a low value for γ indicates a focus on short term optimization, while a γ with a higher value aims for the long term best result.

To deal with the unstable aspects that can come from correlation between states, and might have a big influence how a policy is trained, a replay buffer (D) is introduced. The replay buffer randomizes over the data, removing correlations between states. The target network is also used to stabilize training, it is updated periodically which limits the correlation with previous q-values. To evaluate the quality of the learning of the model, the loss function is used. Equation 5.2 compares the q-values of the policy network with the target network values.

$$L_i(\theta_i) = E_{(s,a,r,s')\sim U(D)}[(r + \gamma max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)^2] \tag{5.2}$$

θ_i denotes the parameters of the policy network, and θ_i^- denotes the parameters of the target network. For a transition (s, a, r, s') sampled form the replay buffer (D), the q-value for the policy network is given by $Q(s, a; \theta_i)$. for the target network $r + \gamma \max_{a'} Q(s', a'; \theta_i^-)$ is used to

calculate the q-value. The loss function than consists of the mean squared error between the policy and target q-values.

For training the ϵ-greedy policy was used, starting at 1 and decreasing till 0.1 in a linear way. To let the agent learn a policy in the beginning the exploration rate is high, allowing for random actions to be taken. When the model is progressing, the epsilon is decreasing so the agent needs to rely more on the learned policy to exploit the right action.

The used DQN algorithm and explanation of useful hyperparameters as well as the action and state space are discussed in Section 5.2

With both models operational on their own, the communication between the two had to be established. The details about the communication will be discussed in Section 5.3. Apart from additional code for communication, the DQN scrips was extended to store information about learning and results. This data was plotted for visual representation of performance.

To start with training the DQN agent, a selection of scenarios is needed. These scenarios are generated via a Python script, with the constraint of all sections being represented equally over all scenarios and no duplicate scenarios are allowed. Based on these scenarios the training process has taken place for the DQN algorithm, this process is elaborated in Chapter 7. Training on one scenarios is performed, in this way it can be seen whether the DQN is able to learn a good policy over time. Training over multiple new scenarios is performed, this policy will be used for testing and shows the ability to learn for the whole network.

The results from testing will be compared to a Greedy Algorithm (GA), which acts as a benchmark. As can be seen in Chapter 4, GA has been used as a benchmark in a variety of studies. The rule-based approach focusses on short term results, by choosing the action with the highest reward possible. Without taking into consideration the long term effects of an action, GA is a basic approach with low computational times and costs. Port authorities might be interesting in applying GA for disruption recovery sequences, as it is quick and will gain results in the short term. The same as with DQN, GA will adapt the choices based on the network discussed. Making it possible to compare results based on a case study, but assume effectiveness of results will be the same in different networks. DQN has to outperform GA up to a certain level before port authorities are willing to invest in a new repair sequence method, that is computationally heavier and costs more time to set up.

Both the DQN and GA are tested on the same scenarios, which are not used for training the DQN algorithm. The results of both algorithms are collected and used for analysis, on which it can be concluded which algorithm performs best.
The remained of this chapter explains functioning of the ABM simulation, the DQN algorithm, communication between the two and the greedy algorithm.

5.1. Simulation

For building the simulation the software tool AnyLogic is used. AnyLogic is a Java based simulation software, combining three methods for building simulations, namely discrete event, agent-based and system dynamics. For this model the agent-based component will be combined with statecharts to guide agent behaviour. The AnyLogic model will act as the environment and set the states and rewards needed for the reinforcement learning algorithm.

The simulation is made to perform a selected scenario, provide the state information and implement the received action. The simulation is build in different layers, the top layer, called main, provides the graphical overview of the state and communicates with the Deep Q-Network. The simulation is built in such a way that with adjusting the input tables and parameter changes, a wide variety of networks can be simulated.

Main

The Main layer is the place where the model will run when a simulation is started. This acts as the environment were the states will be taken from. It holds the connections to the different layers. The environment contains the network that the ships use to navigate to their assigned terminal. This network consists of paths and nodes, the nodes represent the terminals in the port. This network operates separate from the Section and Terminal, which are elaborated below.

The complete structure of the model is represented in Figure 5.1. It can be seen that all layers of the model depend on each other to get the information needed to run the simulation. At the start of the simulation the disrupted sections are marked as disrupted, and the terminals will be based on this set to either available or unavailable. This state is exported to the DQN algorithm and an repair action is received. The RepairAgent is in charge of performing the received action. After the repair is complete the sections and terminals will be updated again, resulting in a new state. While the repair is going on, ships are arriving in the port. When there is an available terminal, the ship will start sailing to the terminal. When no terminal is available, the ship will be placed in a queue and recheck the terminals until an available terminal is found.

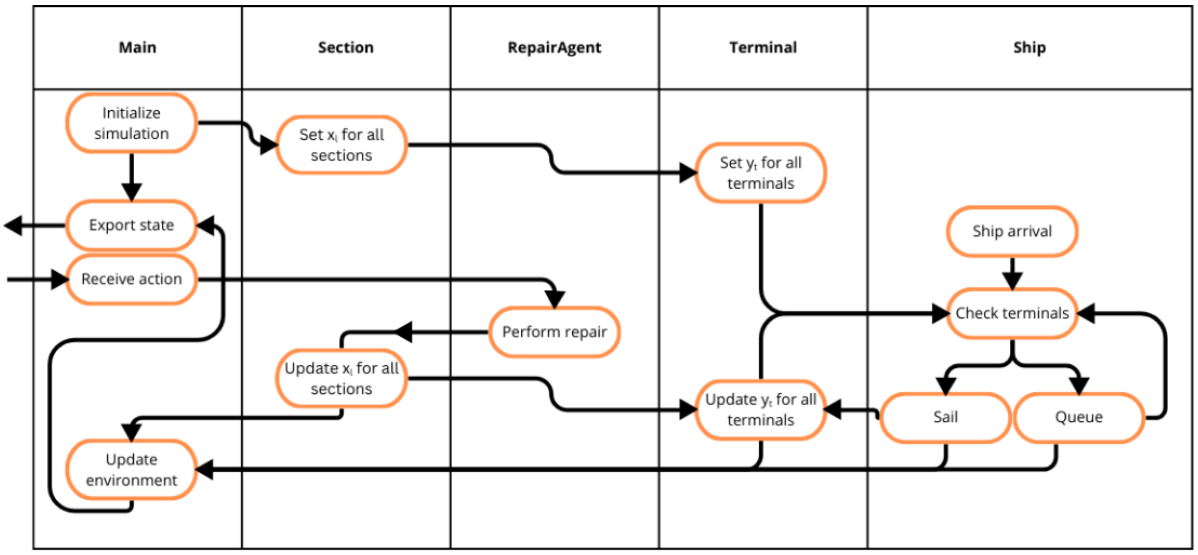

Figure 5.1: Influence chart AnyLogic agents

The mathematical notations for the main agent can be found in Table 5.1. The sets are used to built the complete environment, attributes of these sets will be elaborated in the corresponding sections. The used reward value can be easily adjusted depending on what needs to be learned.

Table 5.1: Attributes of Main

Set	Definition
S	Set of ships
T	Set of terminals
L	Set of waterway sections
I	Set of states
D	Set of cargo types
Variable	
Z_i	Reward value per state i
h_d	Amount of ships waiting of cargo type d

The cargo types set consists of D = {container, dry bulk, liquid bulk, RoRo, general}. For notation this will be transformed to D = {1,2,3,4,5}, each number represents a cargo type.

Waterway sections

The sections represent (sections of) waterways in a port, and have the characteristic to be either operational or disrupted. Which in the simulation is represented by either a green or gray line. The location and index of a section is stored in a database table included in AnyLogic. On start of the simulation these sections are visualized on the main layer. By replacing the database, different networks can be simulated. The attributes used for interacting with the environment and the RepairAgent are listed in Table 5.2.

Table 5.2: Attributes of Waterway sections

Parameter	Definition
r_l	Repair time $\forall l \in L$
Variable	
x_l	{0,1} 0 if a section is active, 1 if a section is inactive $\forall l \in L$

Whether a section is disrupted depends on the scenario that is chosen to run. The scenario can either be chosen from a list of different scenarios, or set specifically. The disruption is only initialized at the beginning of the simulation and no secondary disruptions are present.

Terminal

The terminal layer is as well as the Section layer based on a database table, which is included within the AnyLogic simulation. By replacing this database table, a different simulation can be run with the same logic. The data stored for all terminals is the type of cargo a terminals can handle and the capacity each terminal has. This information is needed to determine whether a terminal has spare capacity for ships. A terminal can only service ships that carry the same cargo type as the terminal is able to handle. In Table 5.3 the attributes of the terminals are listed.

Table 5.3: Attributes of Terminals

Parameter	Definition
p_{td}	Type of cargo the terminal can handle $\forall t \in T, d \in D$
c_t	Capacity of terminal, in number of ships $\forall t \in T$
Variable	
y_t	{0,1} 0 if a terminal is not reachable, 1 if a terminal is reachable $\forall t \in T$
q_t	The current occupied capacity $\forall t \in T$

The cargo type of a terminal is determined before hand and should not conflict with constraint 5.3.

$$\sum_{d \in D} p_{td} = 1 \qquad \forall t \in T \tag{5.3}$$

The capacity (c_t) is calculated by dividing the complete quay length of a terminal by the average length of a ship. The quay lengths are retrieved from the Harbour Master Port Map of the Port of Rotterdam (PoR, 2025). As average ship length 212 meters is used, this is based on the weighted average calculated from EMSA (2023). Whether there is still available place at a terminal can be calculated by taking the difference between the capacity (c_t) and the current occupied capacity (q_t). Equation 5.4 shows how this is written mathematically.

$$q_t \leq c_t \cdot y_t \qquad \forall t \in T \tag{5.4}$$

Ship

The Ship layer defines the behaviour performed by all individual ships. The behaviour is guided via a statechart, making it possible to perform different behaviour depending on circumstances. As the model is not based on real life data, Ships are only known from the moment they enter the port and need to be assigned to a terminal or wait for an available terminal. The behaviour of the ships is shown in Figure 5.2, and the attributes of the ships are shown in Table 5.4.

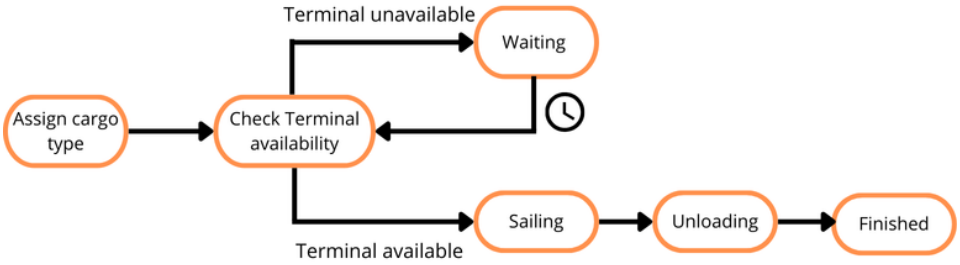

Figure 5.2: Ship state transition diagram

Table 5.4: Attributes of Ships

Parameter	Definition
v_{sd}	Type of cargo a ship carries $\forall s \in S, d \in D$
u_s	Time needed to unload $\forall s \in S$
n_s	Sailing speed $\forall s \in S$
m_s	Time before rechecking available terminals $\forall s \in S$
Variable	
w_s	Waiting time of a ship $\forall s \in S$
k_{std}	{0,1} 1 if ship s of type d is assigned to terminal t, 0 otherwise

On arrival in the simulation each ship gets one cargo type assigned (v_{sd}).

$$\sum_{d \in D} v_{sd} = 1 \qquad \forall s \in S \tag{5.5}$$

Based on the assigned cargo type the agent checks whether a terminal with the same cargo type is available (y_t).

$$v_{sd} \cdot p_{td} \cdot y_t = k_{std} \qquad \forall s \in S, t \in T, d \in D \tag{5.6}$$

If no terminal of the right cargo type is available at the moment, the ship is placed in a virtual queue and gets assigned the status waiting. From this moment the waiting time (w_s) starts being tracked. Ships only have the option to wait, the simulation does not allow to divert to a different port.

The summed waiting time of all ships is used to decide the reward value for the DQN algorithm, which is shared after each step. When arriving in the port all ships have an initial waiting time of 0. Waiting time is in fact not a reward but a penalty, this is why the waiting time is multiplied with -1, making it a negative reward, as shown in Equation 5.7.

$$\sum_{s \in S} w_s \cdot -1 = Z_i \qquad \forall i \in I \tag{5.7}$$

To monitor the amount of ships of a specific cargo type that are waiting, Equation 5.8 is used.

$$\sum_{s \in S} v_{sd} - \sum_{s \in S} \sum_{t \in T} k_{std} = h_d \qquad \forall d \in D \tag{5.8}$$

After a predefined time period (m_s) the ship will enter the Check terminal availability state and loop through the terminals again. When a suitable terminal is found the ship gets assigned to the terminal and starts occupying capacity (k_{std}). Equation 5.9 shows how the assignment of ships influences the occupied capacity at a terminal.

$$\sum_{s \in S} \sum_{d \in p_{td}} k_{std} = q_t \qquad \forall t \; inT \tag{5.9}$$

After getting assigned to the terminal the ship starts sailing to the terminal, all ships currently move with the same speed (n_s). Once a ship arrives at the terminal the unloading time (u_s) will start. When the unloading time is finished the ship has completed the whole process and will be removed from the occupied capacity variable, opening up new capacity at the terminal. The terminal is the end point of the ship. Because there is no additional value in simulating the return of a ship to sea.

In the case where multiple suitable terminals are available for a ship, the distance between a set point at the beginning of the port and the terminal will be calculated. The terminal with the shortest distance will be selected and assigned to the ship. There is no priority system between the ships, the ship that checks and finds a terminal first will be assigned.

RepairAgent

The RepairAgent is in charge of performing the repair action, received by the DQN algorithm, and assigns sections as active after repair. As shown in Figure 5.3 as long as there are disrupted sections, an action is expected to be performed. This action is denoted with a_l, as

can be seen in Table 5.5. After receiving the action the repair time r_l will be started, when this time has passed the next action can be received. When there are no more disrupted sections, the scenario is considered finished. The Repairagent receives one action at a time, so no simultaneous repair is taking place.

Table 5.5: Attributes of RepairAgent

Variable	Definition
a_l	Received action from the RL agent

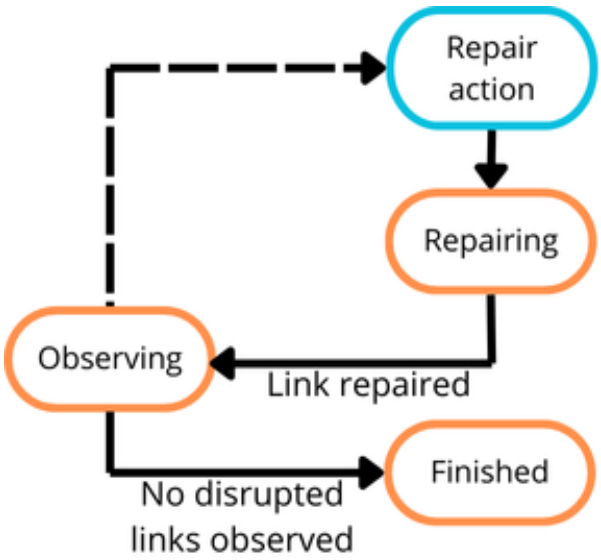

Figure 5.3: RepairAgent state transition diagram

5.2. Deep Q-Network algorithm

The chosen reinforcement learning method is Deep Q-Network (DQN), set with one hidden layer. The algorithm is written in Python and relies on the PyTorch library for reinforcement learning integration. The objective of the DQN algorithm is to minimize the overall waiting time of all vessels (Z_i). By making use of an efficient repair sequence, the overall waiting time should be optimized and approaching zero where possible.

At the start of an episode the state is received from the simulation and the DQN agent knows which sections need to be repaired. The disrupted waterway sections make up the action space for the DQN agent. In the following parts, the state space, action space and DQN algorithm will be explained.

Agent

The agent that will perform the action given by the DQN algorithm, is the RepairAgent explained in Section 5.1. The reinforcement learning agent is tasked with deciding which action needs to be taken. In the training process this is either based on a random exploration or exploitation of known q-values. When testing the DQN algorithm the decision will be solely based on learned q-values.

State space

The state of the environment is a combination of multiple parameters and variables, listed in Table 5.6. As input for the DQN algorithm the status of all the waterway sections in the environment is used, as well as the cumulated waiting time, known as the reward. Next to these two values, the amount of waiting ships per cargo type is given per state. This can be used to optimize the repair sequence, for example by prioritizing repairing sections that can be correlated to a drop in waiting ships of a specific type.

Table 5.6: State input

Variable	Definition
x_l	{0,1} 0 if a section is active, 1 if a section is inactive
Z_i	The reward value of state i
Values	
h_1	The amount of Container ships waiting
h_2	The amount of Dry bulk ships waiting
h_3	The amount of Liquid bulk ships waiting
h_4	The amount of RoRo ships waiting
h_5	The amount of General cargo ships waiting

Action space

The DQN algorithm has an action space equal to the amount of disrupted waterway sections present in the environment.

$$\sum_{l \in L} x_l = \text{size of action space} \tag{5.10}$$

Each disrupted section is considered as an action that can be taken. The set $A \in \{a_l, ..., a_{l+n}\}$ stores the indexes of disrupted sections, that can be used as actions. The set will update after repairing, leaving only disrupted waterway sections as possible actions. In this manner repair actions can only be undertaken on disrupted sections, and no time and resources are allocated to operational parts of the port. This ensured that the focus of learning is on the optimal repair sequence and not on learning whether a section is disrupted or not.

Reward function

The reward function should align with the objective of this research. The assigning of rewards and penalties is used to nudge the DQN algorithm into optimizing the environment. The reward of a state is given by the simulation as Z_i, for each episode the overall waiting time of all ships should be minimized, as shown in Equation 5.11. Because the reward value is negative, the algorithm should choose actions optimizing the objective in the direction of zero.

Depending on the scenario, some scenarios will independent of the repair sequence return a value of zero. This is due to the tree like structure of the network, some sections have little to no influence on the overall waiting time of ships. This can make it hard for the DQN agent to learn a policy, as it needs a rewards value after a state to determine whether an action was effective. If no feedback is received in the form of changing waiting times it can be hard to learn what sequence the waterway sections should be repaired.

$$\min \sum_{i \in I} Z_i \qquad \forall episode \in N \tag{5.11}$$

Discount factor

For this model the long-term performance is the most important, so a high discount factor (γ) will contribute to this. A high discount factor will be used to give the long-term reward priority, this way short-term rewards will be valued less and will help the agent to learn behaviour that optimizes in the long run.

Pseudo code

In Table 5.7 the notations used in the DQN algorithm pseudo code are explained. These notations will be used to give an overview of how the DQN algorithm functions and communicates with the simulation.

In the pseudo code it can be seen that in the first part the hyperparameters, policy network (Q_θ), target network ($Q_{\theta'}$), action set (A) and variables are assigned an initial value of zero. When starting a training episode, first the environment is reset to these basic values, to make sure there is no unintended influence from previous episodes on the episode that will start.

After receiving the state from the simulation, consisting of the reward, the status of the waterway sections and the amount of ships waiting per cargo type. The set of possible actions (A) is collected from the data. As long as there are still disrupted sections in the state, the DQN agent will provide an action for the Agent-Based Model. Either a random action with probability ϵ is chosen or the action with the best q-value is selected.

The chosen action is forwarded to the simulation. After applying the action the reward and new state are shared. The $total\_reward$ variable is updated and the next state is initialized. After a step the transition is stored in the replay buffer (D), the done flag is stored. When the replay buffer reaches the minimum memory (B), the model starts training based on a sample from the replay buffer.

For this sample the policy q-value and target q-value are computed via the equations shown in the pseudo code. Based on the target q-value and policy q-value the loss function is calculated, via a Mean Squared Error (MSE) loss function. After looping through all actions in multiple steps, the episode will be done when no more disrupted sections are present in the environment. On that moment the done flag will be set to true, and the episode will be finished. As long as the next episode is below N, the environment will be reset and the ϵ is updated. After a predefined amount of steps (C), the target network will be updated with the q-values from the policy network.

Table 5.7: DQN notations

Notation	Explanation
Q_θ	Policy network
$Q_{\theta'}$	Target network
N	Number of episodes
B	Batch size
γ	Discount factor
ϵ	Epsilon
ϵ_{decay}	Epsilon decay factor
ϵ_{min}	Minimum epsilon value
A	Action set
D	Replay buffer
C	Target network update frequency

Algorithm 1 DQN pseudo code

Initialize: Hyperparameters
Initialize: Policy network Q_θ, target network $Q_{\theta'} \leftarrow Q_\theta$
Initialize: Replay buffer $D \leftarrow \emptyset$
Initialize: Action set A
Initialize: Training step counter $t \leftarrow 0$
Initialize: $total\_reward \leftarrow 0$
for $episode = 1$ to N **do**
    Reset environment
    $done \leftarrow False$
    Get initial state s
    Collect A
    while not $done$ **do**
        With Probability ϵ select random action a
        Otherwise $a \leftarrow \arg\max_{a \in A} Q_\theta(s, a)$
        Send action a to simulation
        Receive reward r and s' from simulation
        $total\_reward \leftarrow total\_reward + z_i$
        $s \leftarrow s'$
        remove a from A
        if A is empty **then**
            $done \leftarrow True$
        end if
        Store $(s, a, r, s', done)$ in D
        if $|D| \geq B$ **then**
            Sample minibatch $(s, a, r, s', done)$ of size B from D
            Compute q-value:

$$y_i^{pred} \leftarrow Q_\theta(s_i, a_i) \quad \forall i \in B$$

            Compute target q-value:

$$y_i \leftarrow z_i + \gamma \cdot \max_a Q_{\theta'}(s_i', a) \quad \forall i \in B$$

            Compute loss:

$$loss \leftarrow \frac{1}{B} \sum_{i=1}^{B} (y_i^{pred} - y_i)^2$$

        end if
    end while
    $t \leftarrow t + 1$
    Update $\epsilon \leftarrow max(\epsilon_{min}, \epsilon \cdot \epsilon_{decay})$
    if Training step t is divisible by C **then**
        Update $Q_{\theta'} \leftarrow Q_\theta$
    end if
end for

5.3. Interaction between simulation and DQN algorithm

In Figure 5.4 the communication between the Python based DQN agent and the Anylogic based ABM simulation is shown. The communication about the state and action is done via Json files. The state is given by the numbers listed in Table 5.6 and stored as a string, on importing the state into the DQN algorithm the terminal and ships types are converted from categories to numbers. After receiving the state, the decision for the action is made and exported in another Json file. AnyLogic will perform the received action and returns the new state after updating the environment. When no disrupted sections are present in a newly received state file, the episode is finished. The DQN algorithm will run an AutoHotKey script, closing the AnyLogic simulation window and starting a new simulation.

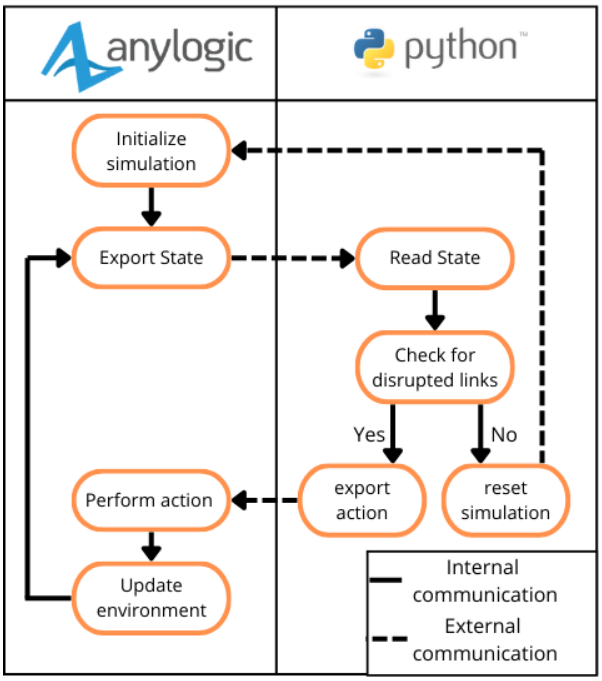

Figure 5.4: Interaction between Anylogic and Python

5.4. Greedy algorithm

As mentioned by Paez et al. (2020) Greedy algorithms have low computational time and are able to adapt to new information, making them more useful in case of emergency. Multiple studies benchmarked their proposed method against a greedy algorithm (Zhang et al., 2017, Fan et al., 2022).

A greedy algorithm has a focus on short-term optimization and tries to reach local optima instead of a global optimum. Local optimums are reached by choosing the action that looks the most beneficial in the moment, without considering the long term effects of making the choice.

To apply a Greedy algorithm to the waterway network in a port, the waterway sections need to be ranked. This allows the Greedy algorithm to prioritize one section over the other, with the aim of reaching an optimal repair sequence. As the aim is to minimize the overall waiting time, sections that are needed for reaching multiple terminals should be ranked higher than sections that only provide access to a single terminal. This indicates that all waterway sections have a certain priority in getting repaired. By assigning a so called priority value to all

sections, the GA can make a decision which section needs to be repaired first. The priority value is based on the amount of terminals that get unreachable due to a disrupted waterway section. In case a waterway section needs to be used to reach 7 different terminals, the priority value attached to that specific section will be equal to 7. In that manner all section of the waterway can be ranked and the GA can distinguish which sections need to be repaired first, to find the local optimum. The priority values used in this research will be explained in Chapter 6.

The Greedy algorithm is written in Python and makes use of the ABM simulation in AnyLogic. The GA communicates in the same way as the DQN, via Json files for receiving disrupted links and resetting the environment with the AutoHotKey script. Table 5.8 denotes the information received by the Greedy algorithm.

Table 5.8: State input GA

Variable	Definition
x_l	{0,1} 0 if a section is active, 1 if a section is inactive
Z_i	The reward value of state i

Based on the state information and an attached Excel file containing the priority values for each waterway section, the GA performs the repair sequence. Below the pseudo code of the Greedy algorithm is given. At the start the Excel file with the priority values needs to be loaded. The set A stores the available actions and is first initialized as an empty set. Next the reward value is set to zero, and the environment is reset. After receiving the state and collecting the available actions, the action is chosen based on the priority values. The chosen action is send to the simulation and the next state is received afterwards. After calculating the reward value and removing the performed action for the set A, the next action will be decided until no more disruptions are present in the simulation.

Algorithm 2 GA pseudo code

Load: Priority values from excel
Initialize: Action set A
Initialize: $total\_reward \leftarrow 0$
for $episode = 1$ to N **do**
    Reset environment
    Get initial state s
    Collect A
    while do$A$ not empty
        Select action with highest priority value
        Send action a to simulation
        Receive reward r and s' from simulation
        $total\_reward \leftarrow total\_reward + r_i$
        remove a from A
    end while
end for

6
Case study

Maritime trade is one of the biggest global transport markets, making it the backbone of logistics. For smooth operation of maritime transport, ports are essential. When ports get completely or partially disrupted this has a high impact on the global transport sector. When managed correctly overall delay of ships can be limited by moving around terminal assignment and prioritizing repair based on the cargo types most effected. Port authorities play a big role in both repairing of infrastructure as well as navigating ships to alternative terminals.

With chances of extreme weather disruption increasing, and dependency on the maritime transport market growing. The maritime sector would benefit from a structured method to calculate repair sequences based on real-time disruptions, with an aim to reduce backlog at ports. To test whether the model proposed in Chapter 5 can contribute to this objective, a case study is performed.

The case study performed is based on the port of Rotterdam. The port of Rotterdam is selected for this case study, based on 5 reasons:

- It is one of the biggest ports in the world and the busiest port in Europe
- A big variety of cargo types are handled within the port
- Openly accessible data about terminals and historical data on handled ships
- The port of Rotterdam has a port authority, in charge of all port infrastructure, including waterways
- Verschuur, Koks, and Hall (2023) identified that over time the Port of Rotterdam will become prone to flooding

As a world leading port with increasing risk of disruptions due to extreme weather events. The port of Rotterdam can be used as an example for increasing resilience in a port and impacting repair sequencing in other ports.
While the port of Rotterdam exists out of numerous separate operated terminals, the waterway infrastructure falls under the responsibilities of the port authority. This makes it possible to execute coordinated repair and aim for solutions that are best for all partners in the port.

The case study will be explained further in the same fashion as the simulation model is introduced. The Agent-Based simulation is a simplification of the port of Rotterdam, holding the important information needed for the Deep Q-Network to learn a repair sequence policy.

6.1. Environment

The network used in the case study is based on the Rotterdam Harbour map (PoR, 2025). It contains all waterway sections and terminals present in the Port of Rotterdam. The network consists of 51 waterway sections and 54 terminals. The complete network is shown in Figure 6.1.

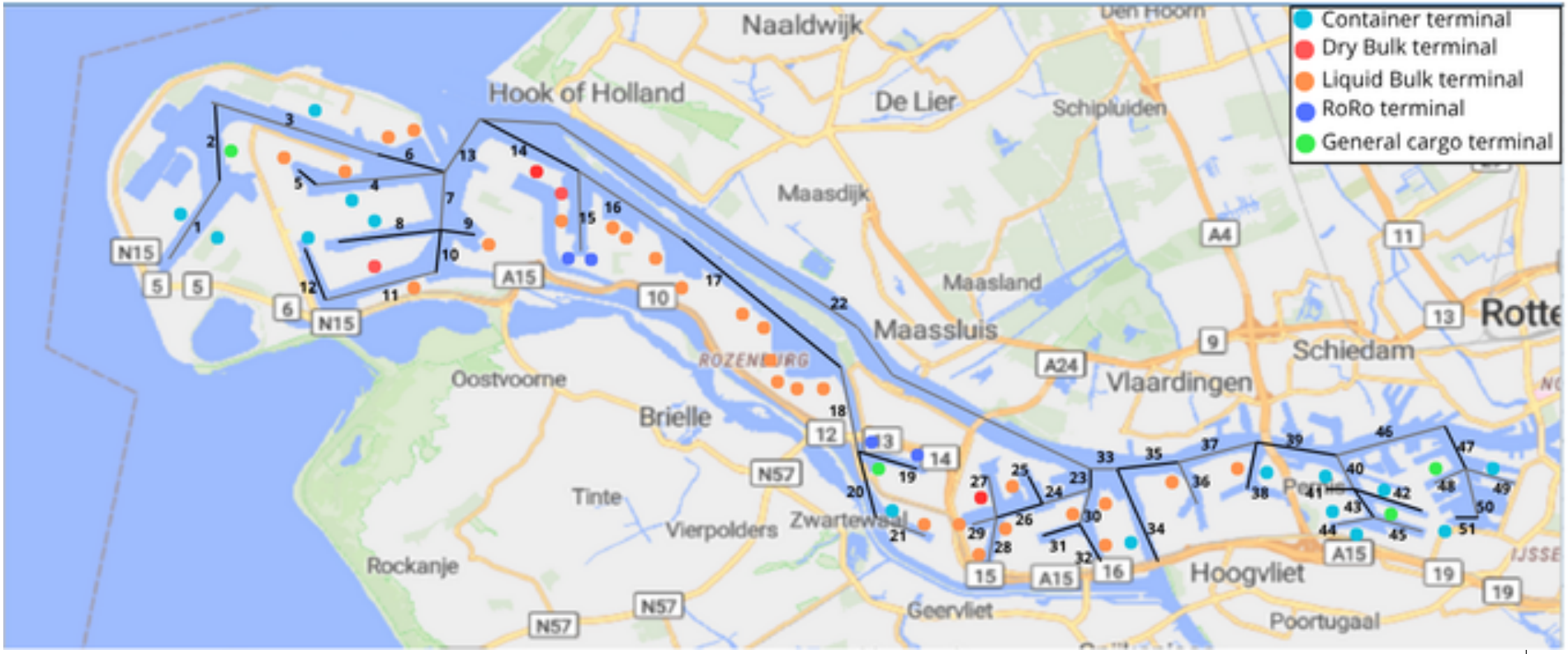

Figure 6.1: Port of Rotterdam network

6.2. Waterway sections

The waterways are dived into 51 sections, the corresponding index for each section can be found in Figure 6.1. All sections operate individually, so each of the sections can get disrupted effecting the surrounding terminals. In the simulation the repair time (r_l) per section is equal for all sections, and is set at 8 hours.

6.3. Terminals

In total 54 terminals are included in the simulation, each terminal is able to handle one type of cargo. This is denoted with the parameter p_{td}, which can be found in Appendix A for all terminals. Table 6.1 displays the subdivision of the terminals per cargo type and the combined amount of quay places denoted as capacity (c_t).

Table 6.1: Subdivision terminals per type

Type of terminal	Amount of terminals	Combined capacity
Container	15	44
Dry Bulk	5	30
Liquid Bulk	26	113
RoRo	4	10
General	4	18

In Figure 6.1 it can be seen that for some cargo types the terminals able to handle these types are clustered around a limited amount of waterway sections. In case of a disruption affecting the few crucial waterway sections leading to for example the RoRo terminals, backlog can be generated quickly leading to high waiting time for ships.

6.4. Ships

Based on data by Stateline (CBS, 2025), in 2024 around 80 ships per day and a total of 29.200 ships per year were registered at the port of Rotterdam. Stateline made a division of the different cargo types, in Table 6.2 the relation between the different ships can be seen.

Table 6.2: Distribution of ship types

Ship type	Amount of ships in 2024	Percentage in 2024	Average amount per day
Container	7000	26.7%	21
Dry bulk	1000	3.8%	3.04
Liquid bulk	8600	32.8%	26
RoRo	4600	17.6%	14
General	4900	18.7%	15
Other	100	0.4%	0.32

Based on the values stated above, the decision is made to exclude the ships being part of the type other from the case study. These are probably ships not carrying cargo, but have their own berth or company in the port.

There are no speed limitations in the Port of Rotterdam, ships can all sail the same set speed, regardless of the terminal they are heading to (PoR, 2025). The sailing speed (n_s) is set to 5 km/hour and applies for all ships. Unloading time (u_s) is also set the same for all vessels, and

takes 5 hours independent of the cargo type. m_s is set at 30 minutes, which is the time spent waiting until a new check for an available terminal.

This research is focussed on the repair sequence of infrastructure links and uses the case study of a port as an example. To not complicate the research, details about the ships, such as draft or specific lengths are not considered.

6.5. Case study specific parameters

In Table 6.3 the parameter values used for the case study can be found. r_l is the time needed for repair, currently all waterway sections have the same repair time which is set at 8 hours. p_{td} lists the type of cargo each terminal can handle. Each terminal can handle only one type of cargo, for the case study the terminals and corresponding cargo type are listed in Appendix A. The same applies for the capacity of each terminal (c_t). The unloading time for a ship (u_s) is uniform across all ships and is set at 5 hours. n_s denotes the sailing speed for all ships, set at 5 km per hour. The time between checking if a terminal is available (m_s), for ships in a queue, is set at 30 minutes.

Table 6.3: Case study parameter values

Parameter	Value
r_l	8 hours
p_{td}	Can be found in Appendix A
c_t	Can be found in Appendix A
u_s	5 hours
n_s	5 km per hour
m_s	30 minutes

6.6. Priority values

The priority value is determined per waterway section and is based on the amount of terminals affected by a disruption of that specific sections. Figure 6.2 and 6.3 show two different disrupted waterway sections. Based on the place of the section in the the network, the priority value differs. As can be seen, when section 18 is disrupted (Figure 6.2) 5 terminals become unreachable. This means that the priority value assigned to this waterway section is equal to 5. The same logic applies for section 13, shown in Figure 6.3, 14 terminals become unreachable when this section will be disrupted, leading to a priority value of 14. The priority value for each waterway section can be found in Appendix B.

Figure 6.2: Affected terminals with section 18 disrupted

Figure 6.3: Affected terminals with section 13 disrupted

In case multiple waterway sections are disrupted the severity of the disruptions can be denoted in an overall priority level by adding the priority values of the individual disrupted sections. In Figure 6.4 it can be seen that in case both waterway sections 13 and 18 are disrupted, a total of 19 terminals become unreachable. So the effects of the disruption can be given the summed priority value of 19.

Figure 6.4: Affected terminals with sections 13 and 18 disrupted

Figure 6.5 shows the priority value per waterway section. All these values are calculated based on the logic explained above.

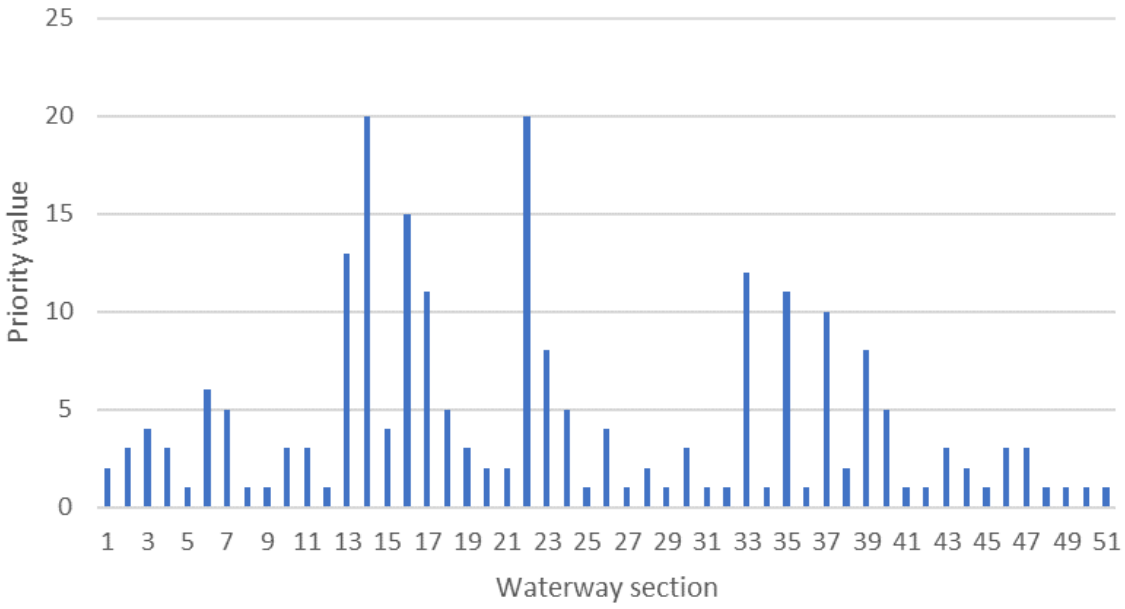

Figure 6.5: Priority value of all waterway sections

7
Training and testing

Based on the case study introduced in Chapter 6 the training and testing is performed. This chapter will first explain the training process of the DQN algorithm. After training the DQN the policy needs to be tested and benchmarked against the GA. Based on the comparison the added value of the DQN can be decided. To see whether the trained policy can be generalized to other scenarios, two different levels of disruption are tested and discussed.

7.1. DQN training

The Deep Q-Network is working with the hyperparameters in Table 7.1, which are based on the hyperparameters used in the paper of Mnih et al. (2015). The learning rate is determined after exploring a range, the value that accommodated the learning best, is selected.

Table 7.1: Hyperparameters of DQN

Hyperparameter	Value
Minibatch size	32
Replay memory size	10000
Discount factor	0.99
Learning rate	0.0001
Initial exploration	1
Final exploration	0.01

The training is done over 3000 episodes, each episode consists of a scenarios with 7 disrupted waterway sections. The complete network considered in the case study consists of 51 waterway sections, resulting in a total of 583,506,504,000 different scenarios with 7 disrupted links. The value of the Deep Q-Network algorithm is that based on a selection of scenarios, it can be trained to perform well on all scenarios.

To train the DQN a total of 102 scenarios, consisting of 7 sections are used. The 51 sections of the waterways had to be represented an equal amount of times in the complete set of training scenarios, to ensure stable learning. This was set as a rule in the Python script used to generate the scenarios. Furthermore, no rules were specified and the scenarios were randomly selected. The scenarios used for training can be found in Appendix C. At the start of each episode one of the scenarios is selected and set up in the simulation. The DQN will come across the same scenario multiple times, meaning in these episodes the same waterway sections are disrupted. But the episodes are not exactly the same, this is due to the arrival of the ships. For example, a disruption of section 15 will lead to a higher overall waiting time, when a lot of RoRo ships arrive. Because section 15 leads to 2 of the 4 RoRo terminals available in the network. When instead of RoRo ships, a lot of container ships arrive, the overall waiting time might be less affected from the same disruption. This could influence the

DQN to consider a different repair sequence, making each episode a new learning opportunity.

Training the model on 3000 episodes with scenarios of 7 disrupted waterway sections, takes about 20 hours. The power of DQN is that training can be done prior to a disruption happening, this makes is possible to have an extensively trained policy on the shelf for when a disruption happens. The DQN learns a policy based on the correlation found between the input variables, the performed action and the received reward value. The used input about the states can be found in Table 7.2.

Table 7.2: State input

Variable	
x_l	Whether a link is disrupted or not
Z_i	Reward value per state i
Values	
h_1	The amount of Container ships waiting
h_2	The amount of Dry bulk ships waiting
h_3	The amount of Liquid bulk ships waiting
h_4	The amount of RoRo ships waiting
h_5	The amount of General cargo ships waiting

Figure 7.1 shows how often each scenario is run during training. By seeing each scenario multiple times, the DQN policy should be able to learn for these scenarios and be able to apply the knowledge on scenarios it has not seen before.

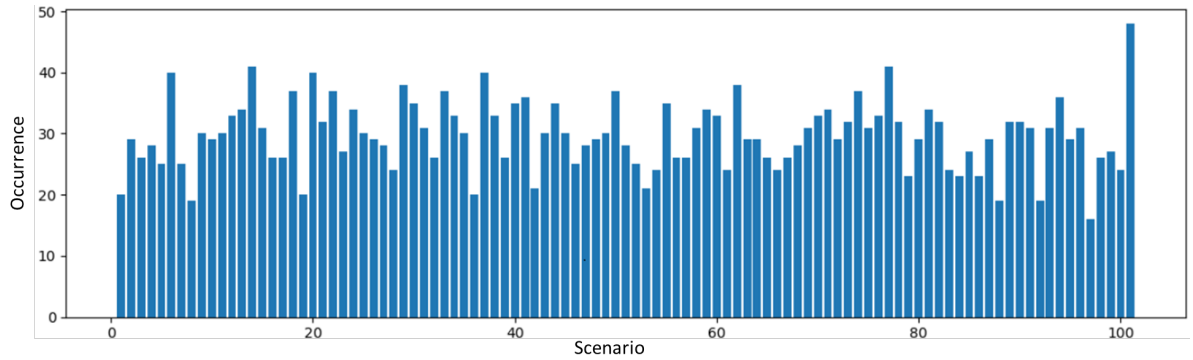

Figure 7.1: Frequency of simulated scenarios

How well the target network and policy network are aligned can be shown by the loss function. For this research the mean squared error is calculated between the q-value of the policy network and the target network q-value and averaged for the minibatch size.
In Figure 7.2 the loss values per episode are shown. As can be seen, the loss value decreases over time, indicating that the target network and policy network get more aligned. The first few values are equal to zero, this is due to the fact that training only starts when the minimum value for the replay buffer of 32 is reached which is after 5 episodes.

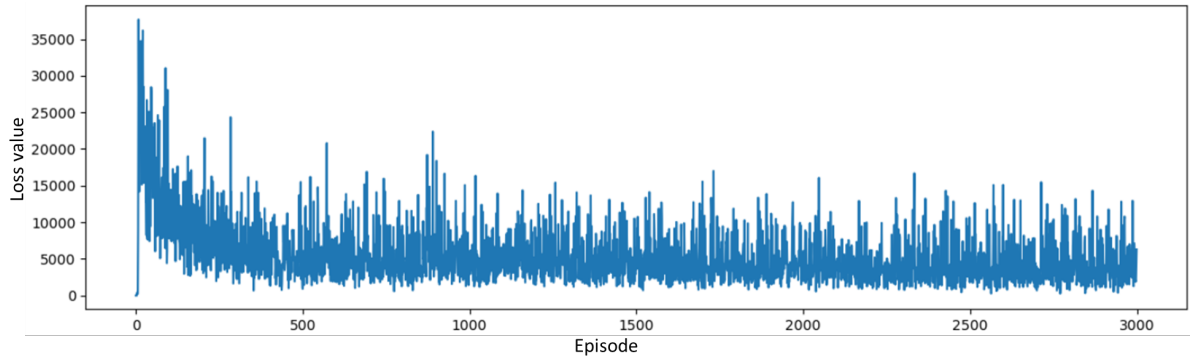

Figure 7.2: Loss value per episode

When the DQN algorithm is trained on a single scenario over 3000 episodes, Figure 7.3 shows that the loss values decreases as well. The chosen scenarios has a summed priority value of 53, assuring that the DQN algorithm has enough feedback to learn a repair sequence. Due to the uncertainty of the ship arrival, the DQN still needs time to adjust the policy and target network with each other.

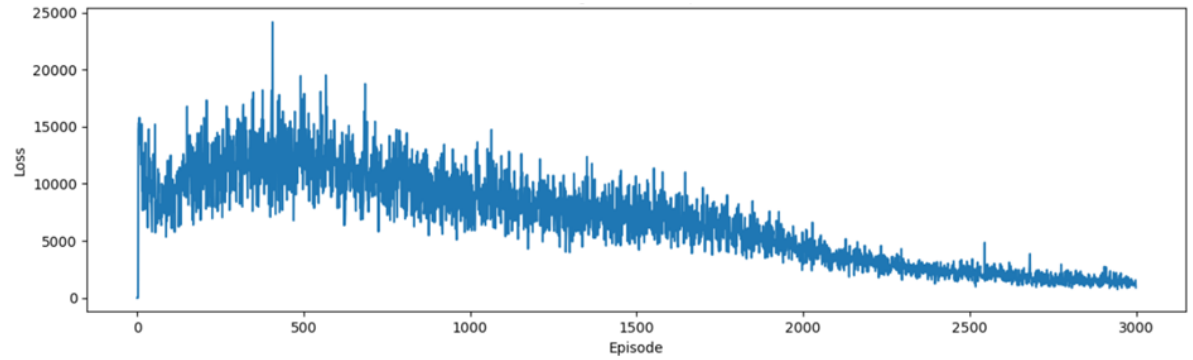

Figure 7.3: Loss values per episode training with one scenario

When testing both of the algorithms on this scenario, the DQN algorithm was able to outperform the Greedy algorithm. This indicates that the DQN is able to learn a good policy.

The learned policy was also checked for effectiveness by running scenarios that is had trained on. In all cases it outperformed a random repair sequence, so it can be concluded that the DQN learns a more optimal repair sequence than randomly repairing waterway sections. Whether the learned policy is more effective compared to a common rule-based approach is studied in the following part.

7.2. Testing

To test the performance of the DQN and benchmark it against the GA, both algorithms need to perform repairs on the same scenarios to be able to compare the results. To see whether the DQN learned a policy that is effective for all scenarios, the scenarios for testing are unique from the scenarios used for training. A total of 20 scenarios is chosen for testing. The scenarios can be found in Appendix D. For each scenario, based on the disrupted waterway sections, the summed priority value is calculated.
These summed priority values are visualized in Figure 7.4, where the difference between the scenarios can be seen clearly.

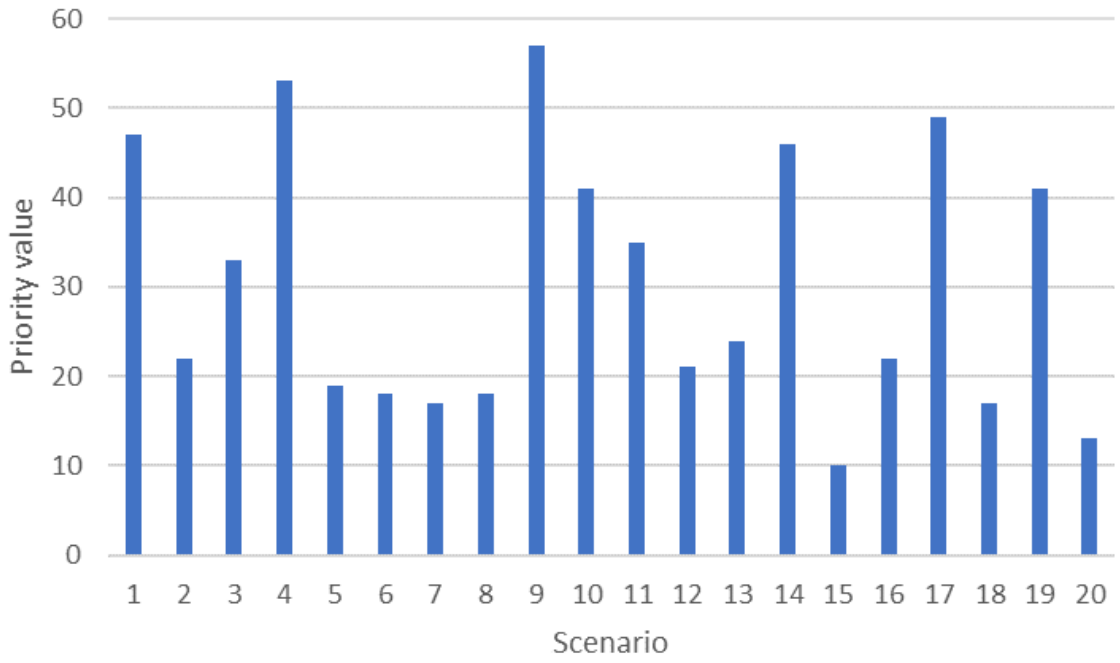

Figure 7.4: Summed priority values of test scenarios

Each of the 20 scenarios is run 100 time by both the DQN as well as the Greedy algorithm. This done to compensate for the uncertainty caused by the ship arrival. The average of the 100 reward values is taken and used for comparing the two algorithms. The averaged reward values per scenarios for both algorithms are listed in Table 7.3.

Table 7.3: Reward values DQN and GA

Scenario	DQN	GA	Summed priority value
1	-0.52	-1.67	47
2	-976.99	-18.03	22
3	0	-19.91	33
4	0	0	53
5	-348.83	-20.17	19
6	-83.65	-559.49	18
7	-0.28	-0.2	17
8	-90.55	-2.67	18
9	-0.53	-0.35	57
10	-0.62	-0.59	41
11	-0.6	–0.55	35
12	-21.59	-2.94	21
13	-0.3	0	24
14	-0.3	-0.2	46
15	0	-0.03	10
16	0	0	22
17	0	0	49
18	-120.12	-66.41	17
19	0	0	41
20	-344.84	-1.19	13

It can be seen that with some scenarios the difference between the two outcomes is large. While with other scenarios it can be seen that the outcomes are equal or close to equal. The results are listed as equal when they are within a margin of 1. The results listed above are visualized in Figure 7.5.

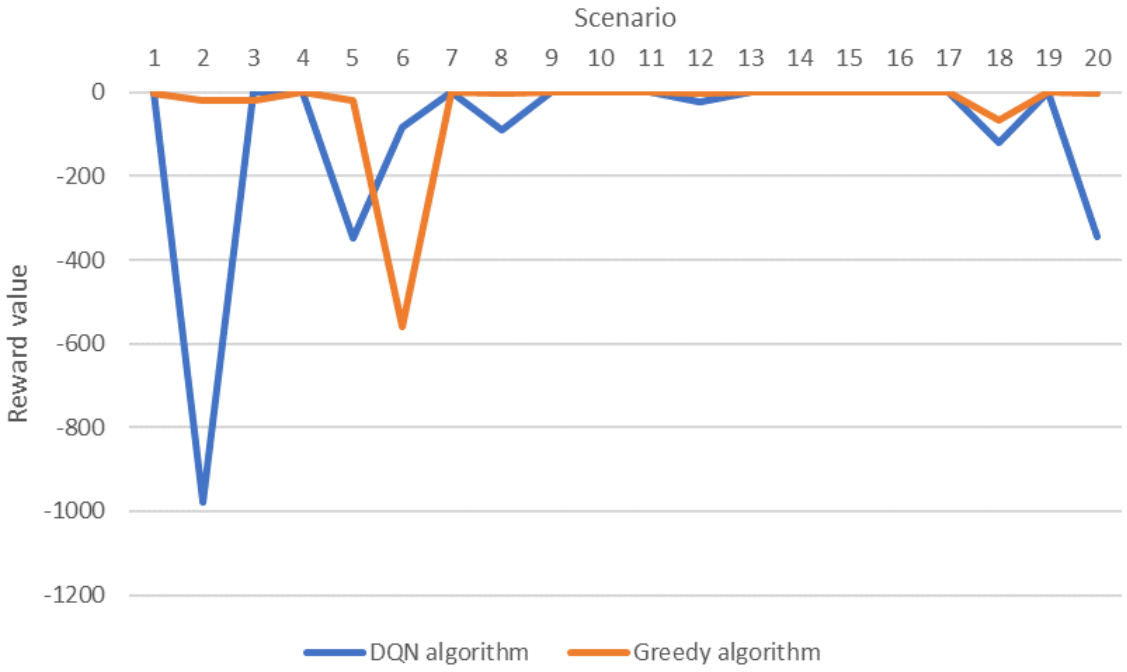

Figure 7.5: Reward values DQN and Greedy algorithm compared

In 6 scenarios the Greedy algorithm outperforms the DQN, When comparing the summed priority values of these 6 scenarios it turns out all scenarios have a relative low summed priority value. In Figure 7.6 these scenarios are highlighted in red next to the other scenarios.

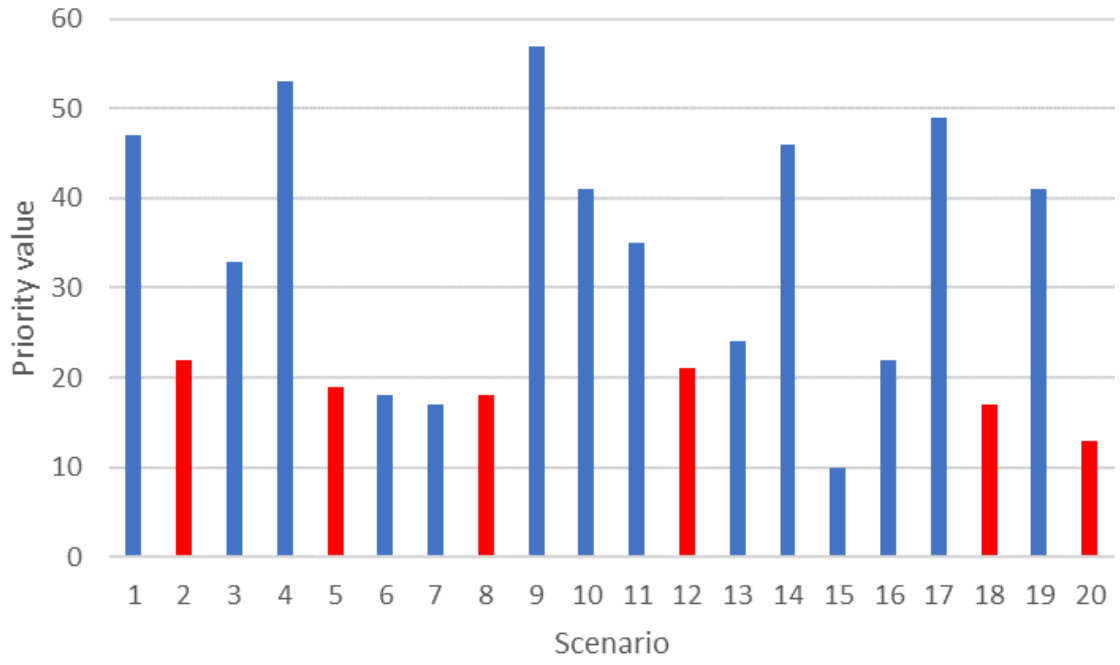

Figure 7.6: Priority values of test scenarios

The reason why these scenarios can not be solved more efficiently by the DQN algorithm, is

most likely because the agent was not able to receive enough feedback. In order to learn a repair policy, the waiting time of the ships is needed. When a scenario consists of waterway sections with low priority, the ships might not be affected, resulting in a reward value of 0. Independent of the sequence in which the waterway sections are repaired, the value will remain the same. This gives the DQN agent no indication of how to efficiently solve these kind of scenarios. When testing with the trained policy, it will have more difficulty with scenarios with low summed priority values.

A low summed priority value indicates that the DQN agent might struggle with finding the optimal repair sequence due to not enough feedback on low priority waterway sections. However a low summed priority value does not automatically mean that GA will perform better. In the test scenarios in can be seen that scenario 6 and 8 have the same summed priority value, difference is that with scenario 6 DQN performs equal to GA. This can be due to the distribution of the priority values within the scenarios. Figure 7.7 and 7.8 show the priority values in red. When looking at the priority values present in both scenarios, scenario 6 has 2 waterway sections with higher priority values, while scenario 8 has mainly low values. But in both scenarios the priority values add up to the same summed priority value. But the DQN agent probably had learn more about the higher priority value in scenario 6, making it possible to match the GA.

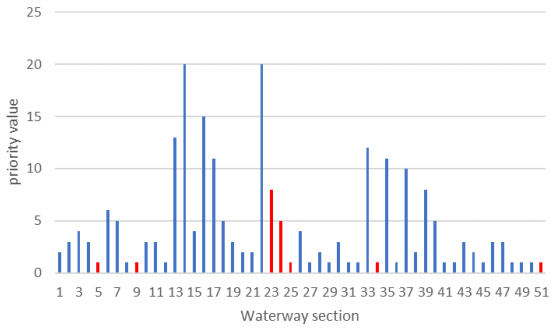

Figure 7.7: Priority values present in scenario 6

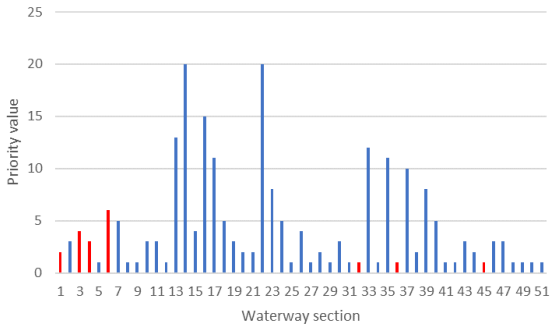

Figure 7.8: Priority values present in scenario 8

To see to what extend it can be concluded that the Deep Q-Network outperforms the Greedy algorithm with increasing summed priority values per scenario, a correlation plot is made. For this plot an additional few scenarios are run, to make sure all summed priority values are represented in the plot. The minimum summed priority value for a scenario is 7, this is made up from 7 waterway sections all with an priority value of 1. The maximum summed priority value is 102, this can be reached when taken the 7 waterway sections with the highest priority values.

Figure 7.9 shows the correlation between the summed priority value and the performance of the DQN algorithm in comparison with the GA. A negative value on the x-axis indicates that the GA outperforms the DQN, while a positive value indicates the DQN outperforming the GA. Based on the trend line it can be seen that with increasing the summed priority value for a scenario, the DQN increasingly outperforms the GA.

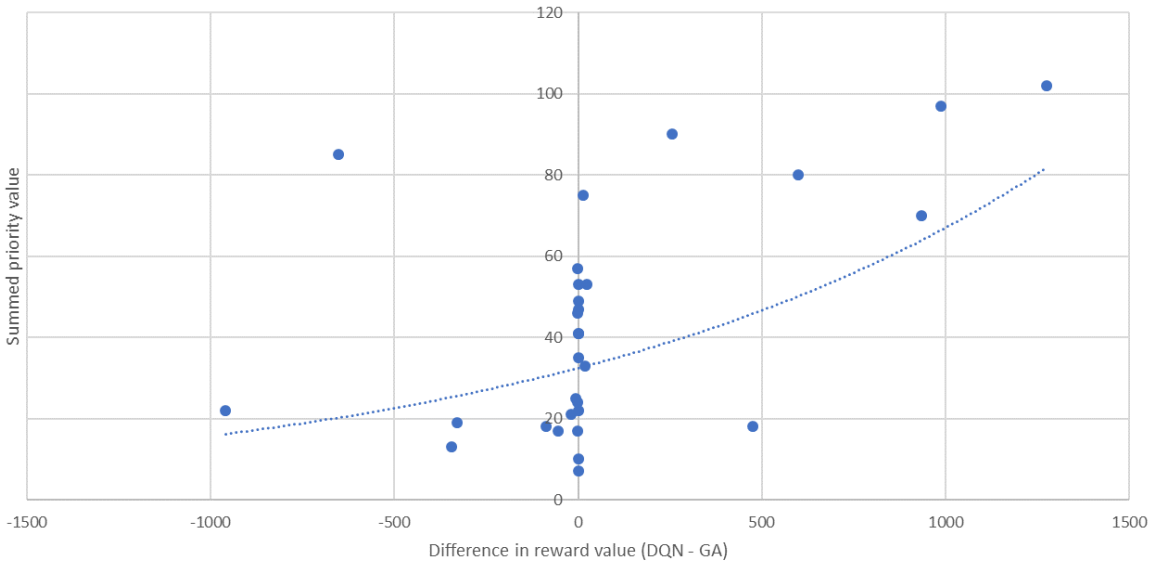

Figure 7.9: Correlation between summed priority value and DQN performance

7.3. Extended testing

To see whether the policy trained on 7 disrupted waterway sections, is also useful for disruptions with a different amount of disrupted waterway sections. The policy is tested on two different levels of disruption, scenarios with 5 disrupted waterway sections, and scenarios with 9 disrupted waterway sections. The testing is done in the same fashion as with the 7 waterway section disruptions. A total of 10 scenarios is run per test, each scenario is run 100 times.

5 disrupted waterway sections

The scenarios used are randomly generated via Python and are all unique. The scenarios along with the summed priority values can be found in Appendix E.
Each scenario is run a 100 times by both the DQN algorithm as well as the Greedy algorithm, the average of the reward value of these 100 run is used for comparison. The average reward values for all scenarios by both DQN and GA are shown in Figure 7.10, it can be seen that in 3 scenarios the Greedy algorithm outperforms the DQN.

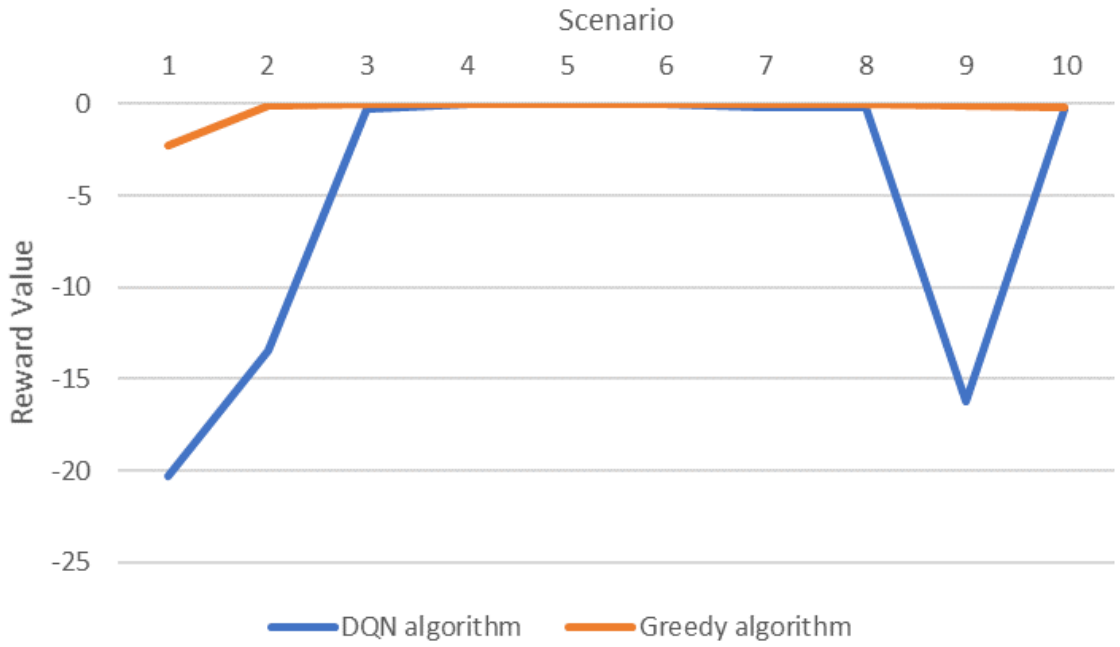

Figure 7.10: Reward values DQN and GA with 5 disrupted waterway sections

The scenarios in which GA outperforms DQN are marked red in Figure 7.11. It can be seen that the summed priority value of these scenarios are on the lower side. Scenarios 4, 5 and 6 also have low summed priority values, these however are so low that both algorithm are able to perform a repair sequence resulting in a reward value of 0.

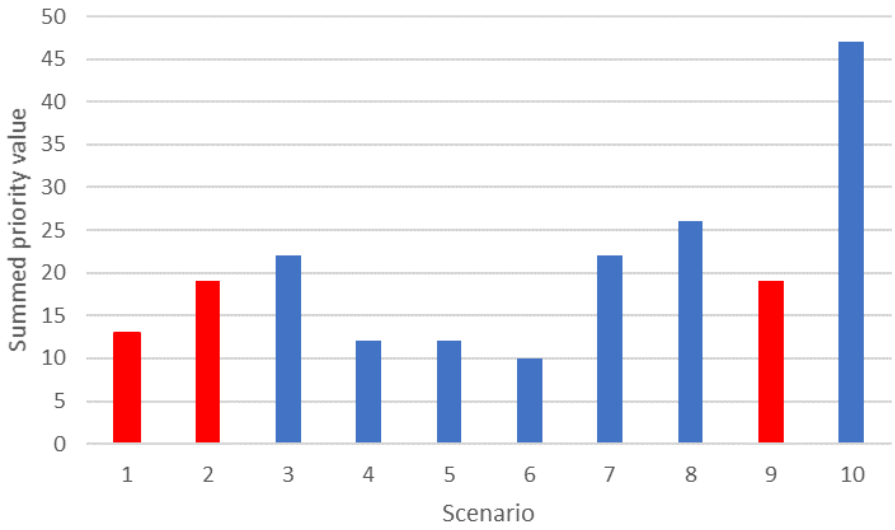

Figure 7.11: Summed priority values

For scenarios with 5 disrupted waterway sections, the policy trained on 7 disrupted waterway sections, performs in the same fashion as with testing on 7 disrupted waterway sections. So in case of scenarios with higher summed priority values, DQN performs equal to GA, but with the tested scenarios the DQN never outperformed the GA. This could be to the limited range

of summed priority values available when testing with only 5 disrupted waterway sections.

9 disrupted waterway sections

The scenarios used for testing with disruptions consisting of 9 disrupted waterway sections can be found in Appendix E.
Each of these scenarios is run a 100 times, and the average reward value is used for comparing the two algorithms. The average reward values for each scenario per algorithm can be seen in Figure 7.12, it can be seen that in 3 scenarios the GA outperforms the DQN.

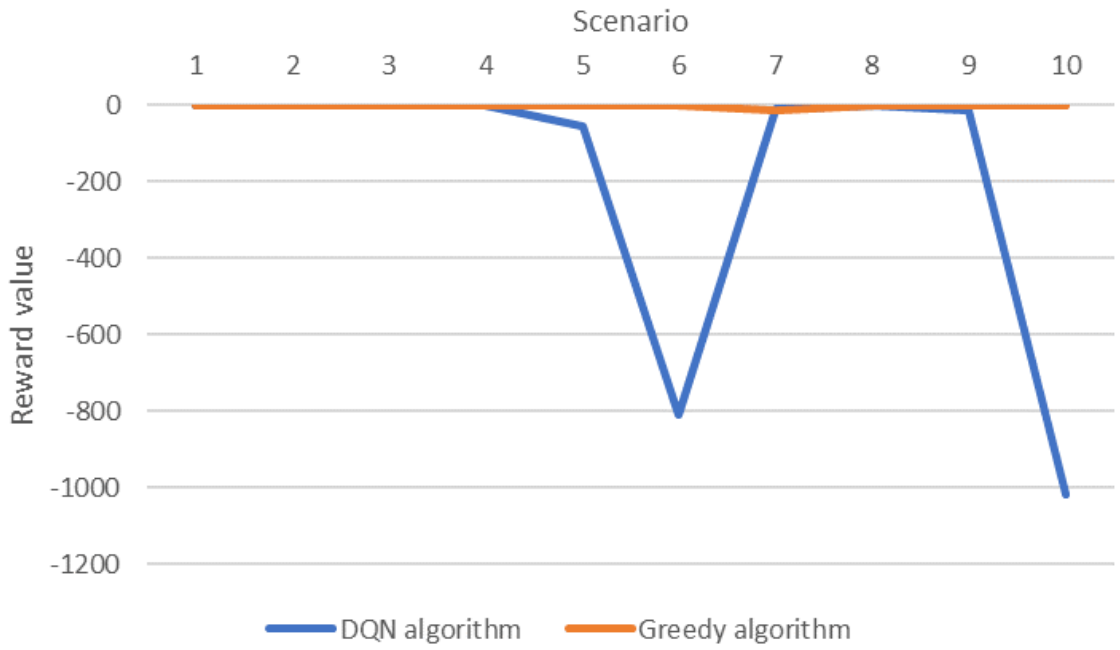

Figure 7.12: Reward values DQN and GA with 9 disrupted waterway sections

The scenarios in which GA outperforms DQN are marked in red in Figure 7.13. It can be seen that the summed priority value of these scenarios are on the lower side. Scenarios 1,2,3 and 8 also have low summed priority values, however these scenarios do not have enough influence on the port operations.

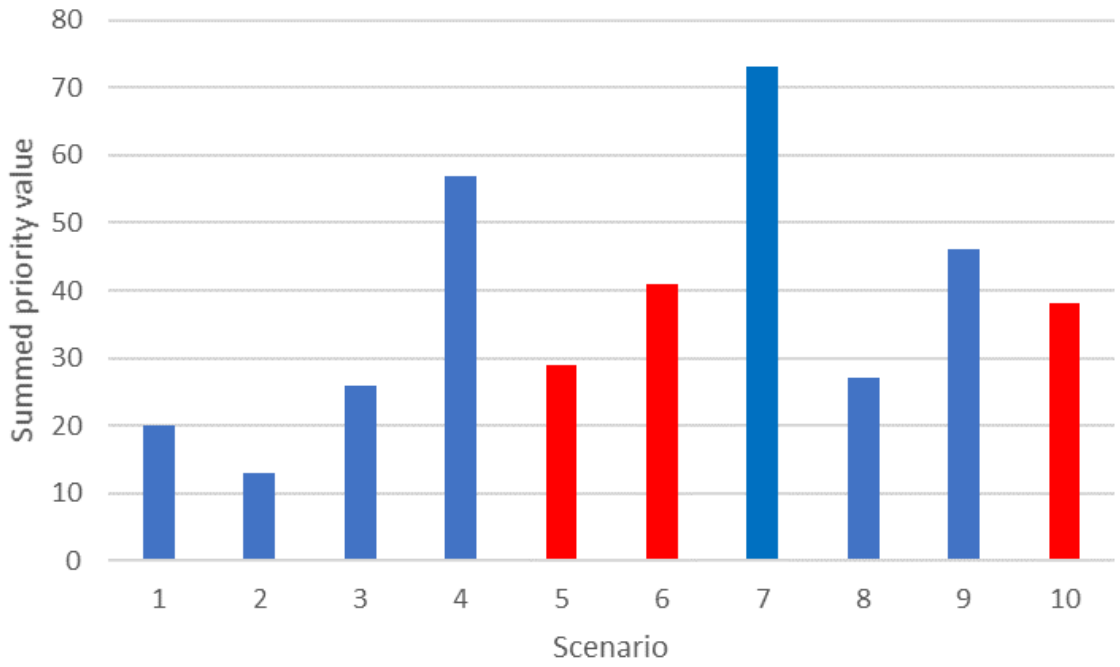

Figure 7.13: Summed priority values per scenario

For scenarios with 9 disrupted waterway sections, the policy trained on 7 disrupter waterway sections, seems to perform in the same fashion as with testing on 7 disrupted waterway sections. Again scenarios with higher summed priority values, perform equal to GA. The DQN never outperforms the GA in the case of 9 disrupted waterway sections.

In both cases of 5 or 9 disrupted waterway sections, the DQN algorithm seems to perform in a similar fashion. However in both cases, the DQN is only able to equal the GA, and not to outperform the GA. This suggest that the trained policy can be applied to different levels of disruption without making big mistakes. Although for performing the optimal repair sequence for a different disruption level, it is suggested to train the DQN on scenarios with different levels of disrupted waterway sections.

8
Conclusion and Discussion

The objective of this research was to explore whether reinforcement learning could optimise the repair sequence of disrupted waterway sections within a port, to minimise the overall waiting time for ships. No prior research on this topic was found, while the risk of port disruptions is increasing (Verschuur, Koks, Li, and Hall, 2023). With maritime trade relying on operational ports and more than 80% of goods being shipped through ports, recovering from disruptions in the most efficient way possible is important for port authorities. This research relies on a case study based on the port of Rotterdam, however the model is built to accommodate different ports or networks as well.

8.1. State of the art for repair sequence strategies

The current state of the art in repair sequence strategies is moving from traditional rule-based and optimisation approaches to data-driven and learning-based models. Earlier work relied mainly on metaheuristics and rule-based approaches (Paez et al., 2020). These approaches are useful for pre-disaster planning, but the high computational costs limit the real-time use. Greedy algorithms, on the other hand, offer adaptability and speed, making them more suitable for disruption recovery.

In recent years, research has increasingly shifted towards machine learning approaches, especially reinforcement learning. Zhang et al. (2017) used active learning to develop recovery strategies for a road-bridge network. The trained policy was benchmarked against a betweenness-first strategy and a Greedy algorithm, outperforming both. Su et al. (2022) and Fan et al. (2022) both use a deep reinforcement learning approach, which limits the amount of training needed to develop a recovery policy. The added value of training a policy beforehand is that once a disruption happens, the disruption only needs to be given to the trained algorithm, resulting in a repair sequence. Although both studies do not involve waterway networks within a port, the results of these approaches are promising and recover network more efficient in comparison the used benchmarks.

Sun and Zhang (2020) applied a deep learning algorithm to an Agent Based Modelling simulation. In this manner individual agent behaviour can be modelled and used to train the DQN network on. ABM simulations have been recognized as useful simulations to learn about human behaviour as well to test programmed behaviour (Huang et al., 2022). Recent literature highlights the potential of combining ABM with optimization or reinforcement learning algorithms to enhance disruption recovery strategies.

This suggest that reinforcement learning in combination with Agent-Based Modelling, can be the new direction for finding optimal repair sequence strategies. This approach has been proven useful in a variety of networks, but is not yet implemented in the context of port waterways, although it could be largely beneficial for port authorities to have an efficient recovery

sequence strategy. Optimizing the repair sequence of disrupted waterway sections, can lead to lower downtimes in the port, and reduced costs for repairing.

8.2. State-space, actions and rewards

The aim of this research is to minimize overall waiting time for ships, that arrive in a disrupted port. To let the DQN algorithm learn a policy that is able to come up with an optimal repair sequence, it needs information about the state. The state is taken from the ABM simulation in AnyLogic and exported via a Json file. As reward value, the combined waiting time in an episode is used. Because the waiting time needs to be minimized, a value of -1 is assigned to each time unit a ship has waited. The reward value together with the state values are exported to the Python based DQN algorithm. Based on the received state the action space can be established. To focus the DQN algorithm on learning an optimal repair sequence and not on being able to recognize disrupted waterway sections, only disrupted waterway sections are part of the action space.

Next to to the status of all waterway sections and the reward value, the state space consists of the cumulative amount of ships waiting per cargo type. This could direct the DQN agent in adjusting the repair sequence based on which ships are waiting, making the repair sequence both scenario specific and also unique for the ships that are waiting.

In the future the state space could be extended to include more information about the environment. This research used the basic information needed to train, it is expected that with adding more relevant information, the DQN algorithm will get even better at optimizing the repair sequence.

8.3. Evaluation measures

To keep track of the training process of the DQN algorithm, the loss function is used to see whether the policy network and target network align. It can be seen that when training over 3000 episodes the loss function declines over time, suggesting that the policy and target network align. This however does not immediately imply that a good policy is learned.

To test the effectiveness of both the Deep Q-Network as well as the Greedy algorithm, the reward value is used. To make sure the reward values could be compared with each other, both algorithms recovered the same 20 scenarios a 100 times. These 20 scenarios were unique from the scenarios used to train the DQN. The average reward value of these 100 episodes was taken to compensate for uncertainty from random ship arrival.

By using a the Greedy algorithm as a benchmark the results from the DQN can be put into perspective. The DQN can be called effective when it at least equals the results generated by the GA. To test the efficiency of the DQN algorithm in comparison with established methods, the DQN should be further test against different methods.

8.4. Comparing the Deep Q-Network with the Greedy algorithm

For comparing the two algorithms, a total of 20 scenarios is run within the ABM simulation. Each scenario consisted of 7 disrupted waterway sections, within a network of 51 waterway sections. The reward value is used to compare the two algorithms with each other. Of the 20 scenarios, there were six scenarios in which the Greedy algorithm outperformed the Deep Q-Network algorithm. It can be concluded that the severity of the disruption, denoted by the

summed priority value, has an effect on the quality of the repair sequence performed by the DQN. The higher the summed priority value, the better the DQN seems to equal or even outperform the GA.

Apart from the summed priority value of a scenario, the distribution of the priority values over the disrupted waterway section in the scenario seems to influence the ability of the DQN to perform.

The difficulty of the DQN for performing in scenarios with low summed priority values is most likely due to a lack of feedback when training. The DQN algorithm relies on the state input to receive feedback on the performed action. If a disrupted waterway section has a low priority value, it could be the case that by repairing this specific section, no change is seen. And so no feedback is given to learn whether the action was a good one or not.

The correlation between summed priority levels and the performance of the DQN algorithm, is shown to be positive. By increasing the summed priority value, the reliability of the DQN to outperforming the GA increases as well.
By testing the DQN policy on different levels of disruption, namely 5 and 9 disrupted waterway sections, it can be seen that the policy behaves the same. However for applying the DQN algorithm on bigger network and more severe disruptions it is suggested to train on more severe disruptions. This would make the policy better to generalize to different levels of disruption.

8.5. Implementing reinforcement learning algorithms for repair optimization in ports

This research used an ABM simulation in combination with a Deep Q-Network algorithm, with the aim of reducing overall waiting time for ships arriving at a disrupted port, by optimizing the repair sequence. The proposed model is benchmarked against a Greedy Algorithm and tested for generalization. It can be concluded that the trained DQN behaves similar in different levels of disruptions.

The DQN algorithm however, has trouble with disruptions with a low summed priority value, being disruptions with a small influence on the port operations. While it can be seen that with disruptions with a high influence, the DQN increasingly outperforms the Greedy algorithm. To overcome this issue, the DQN should receive better feedback while training. This could be done by extending the information provided by the state input. Improving the reward value used to evaluate actions, can benefit the ability to learn about waterway sections with lower influence on the port operations. For example including the capacity of affected terminals could give the DQN algorithm a better idea of the impact a waterway sections has on the port operations.

Investing in a well tailored DQN algorithm based on port specific operations and network structure, would be beneficial for port authorities. By training a policy before a disruption happens, valuable time can be saved when repair can be started. Instead of identifying which waterway sections are disrupted and then starting with planning a repair strategy, the disrupted waterway section can be given to the DQN algorithm resulting in a quick optimal repair sequence in minutes. In this manner repair can start earlier, leading to less backlog and lost revenue. By training the DQN algorithm and testing it within the simulation, crucial waterway section can be identified and port authorities can make resilience predictions in advance, instead of using historical data.

Employing reinforcement learning algorithms to optimize repair sequence strategies in disrupted port, with the aim of minimizing waiting time for ship, is a promising research direction. The current model provides a proof of concept, but is not yet suited for real-life usage. Future research could focus on extending the proposed model with the aim of reflecting port operations in a more realistic way.

Several assumptions were made in the development of the model which can be improve to increase realistic operations. The model assumes that repair can start immediately after a disruption happens. In reality, the effected waterway sections need to be identified and conditions have to settle down before repair can be started. The model assumes that all equipment and material needed for a repair are immediately available. In reality it could be the case that equipment or material needed for the repair are limited available or first need to arrive at the port before repair can start. Another simplification involves the simulation of the ships arriving. Ships arriving in the model are randomly assigned a cargo type, without this being based on actual arrival of ships in the port. By making use of historic arrival datasets that are specific for the port, the DQN agent should be able to predict even better repair sequences and optionally could learn seasonal differences.

An extension would be to allow for multiple links to be repaired at the same time. In practice, multiple repairs crews will be available, giving port authorities the possibility to repair simultaneously. This extension would require to integrate operational constraints concerning resource availability. By integrating operational constraints, it would be interesting to assign monetary values in the simulation. This could be integrated into the reward value, with the possibility of nudging the DQN algorithm into a cost effective repair sequence.

In conclusion, this research shows the Deep Q-Network algorithms can effectively contribute to optimal repair sequences in disrupted port waterways, in the case of influential disruptions. With further development and usage of real-life data and operation constraints, DQN algorithms can be a valuable tool for port authorities. With improving the model, the usage of the Deep Q-Network algorithm should become useful in a wider range of scenarios within a disrupted port.

References

Çağnan, Z., Davidson, R. A., & Guikema, S. D. (2006). Post-Earthquake Restoration Planning for Los Angeles Electric Power. *Earthquake Spectra*, *22*(3), 589–608. https://doi.org/10.1193/1.2222400

CBS. (2025). *Zeevaart; aantal schepen, type schip*. CBS. https://opendata.cbs.nl/statline/portal.html?_la=nl&_catalog=CBS&tableId=85602NED&_theme=434

Chand, S. S., Walsh, K. J. E., Camargo, S. J., Kossin, J. P., Tory, K. J., Wehner, M. F., Chan, J. C. L., Klotzbach, P. J., Dowdy, A. J., Bell, S. S., Ramsay, H. A., & Murakami, H. (2022). Declining tropical cyclone frequency under global warming. *Nature Climate Change*, *12*(7), 655–661. https://doi.org/10.1038/s41558-022-01388-4

Chang, S. E. (2000). Disasters and transport systems: Loss, recovery and competition at the port of kobe after the 1995 earthquake. *Journal of Transport Geography*, *8*(1), 53–65. https://doi.org/https://doi.org/10.1016/S0966-6923(99)00023-X

Chen, B., Cheng, H. H., & Palen, J. (2004, September). *Agent-based real-time computing and its applications in traffic detection and management systems* (Vol. Volume 4: 24th Computers and Information in Engineering Conference). https://doi.org/10.1115/DETC2004-57707

EMSA. (2023). *The 2022 world merchant fleet statistics from equasis*. EMSA. https://www.emsa.europa.eu/equasis-statistics/download/7697/472/23.html

Fan, X., Zhang, X., Wang, X., & Yu, X. (2023). A deep reinforcement learning model for resilient road network recovery under earthquake or flooding hazards. *Journal of Infrastructure Preservation and Resilience*, *4*(1). https://doi.org/10.1186/s43065-023-00072-x

Fan, X., Zhang, X., & Yu, X. ((2022). A graph convolution network-deep reinforcement learning model for resilient water distribution network repair decisions. *Computer-Aided Civil and Infrastructure Engineering*, *37*(12), 1547–1565. https://doi.org/https://doi.org/10.1111/mice.12813

Gou, X., & Lam, J. S. L. (2018). Risk analysis of marine cargoes and major port disruptions. *Maritime Economics & Logistics*, *21*(4), 497–523. https://doi.org/10.1057/s41278-018-0110-3

Gueli, R., Amatore, A., Accardo, E. S., Raccuglia, M., Giandolfo, R. A., & Spampinato, E. (2023). Agent-based model for realistic vessel route simulation in port areas. *2023 IEEE International Workshop on Metrology for the Sea; Learning to Measure Sea Health Parameters (MetroSea)*, 380–384. https://api.semanticscholar.org/CorpusID:265257617

Huang, J., Cui, Y., Zhang, L., Tong, W., Shi, Y., & Liu, Z. (2022). An overview of agent-based models for transport simulation and analysis. *Journal of Advanced Transportation*, *2022*(1), 1252534. https://doi.org/https://doi.org/10.1155/2022/1252534

Izaguirre, C., Losada, I. J., Camus, P., Vigh, J. L., & Stenek, V. (2020). Climate change risk to global port operations. *Nature Climate Change*, *11*(1), 14–20. https://doi.org/10.1038/s41558-020-00937-z

Lam, J. S. L., & Lassa, J. A. (2017). Risk assessment framework for exposure of cargo and ports to natural hazards and climate extremes. *Maritime Policy & Management*, *44*(1), 1–15. https://doi.org/10.1080/03088839.2016.1245877

Li, C., Yang, X., & Yang, D. (2025). Port vulnerability to natural disasters: An integrated view from hinterland to seaside. *Transportation Research Part D: Transport and Environment*, *139*, 104563. https://doi.org/https://doi.org/10.1016/j.trd.2024.104563

Li, L., Xiong, X., & Yuan, H. (2022). Ships' response strategies to port disruptions caused by hurricanes. *Ocean & Coastal Management*, *227*, 106275. https://doi.org/https://doi.org/10.1016/j.ocecoaman.2022.106275

Li, X., Chua, J. Y., & Yuen, K. F. (2024). A review on maritime disruption management: Categories, impacts, and strategies. *Transport Policy*, *154*, 40–47. https://doi.org/https://doi.org/10.1016/j.tranpol.2024.05.013

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Paez, D., Filion, Y., Castro-Gama, M., Quintiliani, C., Santopietro, S., Sweetapple, C., Meng, F., Farmani, R., Fu, G., Butler, D., Zhang, Q., Zheng, F., Diao, K., Ulanicki, B., Huang, Y., Deuerlein, J., Gilbert, D., Abraham, E., Piller, O., … Walski, T. (2020). Battle of postdisaster response and restoration. *Journal of Water Resources Planning and Management*, *146*(8), 04020067. https://doi.org/10.1061/(ASCE)WR.1943-5452.0001239

PoR. (2025). *Harbour master port map*. Port of Rotterdam. https://experience.arcgis.com/experience/aef5cad4ef6a437499c2ba6a7606c07b/page/Harbour-Master-Port-Map

Proag, V. (2014). The concept of vulnerability and resilience [4th International Conference on Building Resilience, Incorporating the 3rd Annual Conference of the ANDROID Disaster Resilience Network, 8th – 11th September 2014, Salford Quays, United Kingdom]. *Procedia Economics and Finance*, *18*, 369–376. https://doi.org/https://doi.org/10.1016/S2212-5671(14)00952-6

Shen, Y., Duan, L., Zhu, Q., Su, Z., & Zhang, G. (2022). Multiagent q-learning for multicrew dynamic scheduling and routing in road network restoration, 1217–1222. https://doi.org/10.1109/CCDC55256.2022.10033659

Shepard, J. U., & Pratson, L. F. (2020). Maritime piracy in the strait of hormuz and implications of energy export security. *Energy Policy*, *140*, 111379. https://doi.org/https://doi.org/10.1016/j.enpol.2020.111379

Su, P. Z., DUAN, L., ZHANG, G., CHANG, J., & SHEN, Y. (2022). Multicrew scheduling and routing in road network restoration based on deep q-learning. *AAAI-22 Workshop on Machine Learning for Operations Research (ML4OR)*. https://openreview.net/forum?id=QAKb7o4IAEQ

Sun, J., & Zhang, Z. (2020). A post-disaster resource allocation framework for improving resilience of interdependent infrastructure networks. *Transportation Research Part D: Transport and Environment*, *85*, 102455. https://doi.org/https://doi.org/10.1016/j.trd.2020.102455

Verschuur, J., Koks, E. E., & Hall, J. W. (2023). Systemic risks from climate-related disruptions at ports. *Nature Climate Change*, *13*(8), 804–806. https://doi.org/10.1038/s41558-023-01754-w

Verschuur, J., Koks, E. E., Li, S., & Hall, J. W. (2023). Multi-hazard risk to global port infrastructure and resulting trade and logistics losses. *Communications Earth & Environment*, *4*(1). https://doi.org/10.1038/s43247-022-00656-7

Wan, C., Tao, J., Yang, Z., & Zhang, D. (2021). Evaluating recovery strategies for the disruptions in liner shipping networks: a resilience approach. *The International Journal of Logistics Management*, *33*(2), 389–409. https://doi.org/10.1108/ijlm-05-2021-0263

Wan, C., Yang, Z., Zhang, D., Yan, X., & Fan, S. (2017). Resilience in transportation systems: a systematic review and future directions. *Transport Reviews*, *38*(4), 479–498. https://doi.org/10.1080/01441647.2017.1383532

Wang, F.-Y. (2005). Agent-based control for networked traffic management systems. *IEEE Intelligent Systems*, *20*(5), 92–96. https://doi.org/10.1109/MIS.2005.80

Wang, S., Sarker, B. R., Mann, L., & Triantaphyllou, E. (2004). Resource planning and a depot location model for electric power restoration. *European Journal of Operational Research*, *155*(1), 22–43. https://doi.org/https://doi.org/10.1016/S0377-2217(02)00803-2

Zhang, W., Wang, N., & Nicholson, C. (2017). Resilience-based post-disaster recovery strategies for road-bridge networks. *Structure and Infrastructure Engineering*, *13*(11), 1404–1413. https://doi.org/10.1080/15732479.2016.1271813

A

Terminal information

Terminal index	Name	Type	Capacity
1	RWG_terminal	Container	6
2	APM_terminal_mv2	Contianer	4
3	SIF	General	3
4	Hutchison_ECT_euromax	Container	6
5	Gate_terminal	Liquid bulk	4
6	MOT_terminal	Liquid bulk	4
7	LyondelBasel	Liquid bulk	2
8	Neste	Liquid bulk	2
9	Hutchison_ECT_Delta	Container	5
10	Hutchison_ECT_Delta_2	Container	4
11	Rotterdam_container_terminal	Container	1
12	Europees_Massagoed_Overslagbedrijf	Dry bulk	10
13	Impala_terminal	Liquid bulk	5
14	BP_raffinaderij	Liquid bulk	8
15	Ertsoverslagbedrijf_Europoort	Dry bulk	5
16	ADM_europoort	Dry bulk	3
17	European_Bulk_service	Dry bulk	3
18	Stena_line	RoRo	2
19	PO_Ferries	RoRo	3
20	Shell_europoort_terminal	Liquid bulk	6
21	TEAM_terminal	Liquid bulk	6
22	Standard_storage	Liquid bulk	2
23	Rotterdam_raffinaderij	Liquid bulk	6
24	Exxon_mobile	Liquid bulk	1
25	Lutra_shipping	Liquid bulk	5
26	Euro_tank_terminal	Liquid bulk	3
27	Lyondell_chemie	Liquid bulk	2
28	Maatschap_Europoort_terminal	Liquid bulk	5
29	VoPak_terminal_europoort	Liquid bulk	12
30	Cobelfret_Ferries	RoRo	4
31	Rotterdam_car_terminal	RoRo	1
32	CLdN_distriport	General	3
33	Steinweg_handelsveem	Container	1
34	Lyondell_chemie_BV	Liquid bulk	1
35	Liquin_botlek	Liquid bulk	2
36	HES_bulk_terminal_maasdelta	Dry bulk	9
37	HES_tank_terminal_botlek	Liquid bulk	4

38	Liquin_botlek_2	Liquid bulk	3
39	Tepsa_netherlands	Liquid bulk	3
40	Liquin_botlek_3	Liquid bulk	6
41	Change _terminal_botlek	Liquid bulk	4
42	LBC_rotterdam	Liquid bulk	2
43	Cetem_containers	Container	2
44	Chance_terminal_vondelingplaat	Liquid bulk	5
45	Change_terminal_nieuwe_maas	Liquid bulk	10
46	Combi_terminal_twente_rotterdam	Container	1
47	Mainport_container_services	Container	1
48	Matrans_rotterdam_terminal	Container	3
49	Rotterdam_short_sea_terminal	Container	6
50	Steinweg_beatrixkade	General	7
51	United_waalhaven_terminals	Container	1
52	Steinweg_waalhaven	General	5
53	United_waalhaven_terminals_2	Container	1
54	Steinweg_waalhaven_2	Container	2

B

Greedy algorithm values

Link	Priority value
1	2
2	3
3	4
4	3
5	1
6	6
7	5
8	1
9	1
1	3
11	3
12	1
13	13
14	20
15	4
16	15
17	11
18	5
19	3
20	2
21	2
22	20
23	8
24	5
25	1
26	4
27	1
28	2
29	1
30	3
31	1
32	1
33	12
34	1
35	11
36	1

37	10
38	2
39	8
40	5
41	1
42	1
43	3
44	2
45	1
46	3
47	3
48	1
49	1
50	1
51	1

C

Training dataset

scenario id	link 1	link 2	link 3	link 4	link 5	link 6	link 7
1	6	15	24	38	39	41	42
2	8	17	21	24	29	37	40
3	9	11	18	22	27	30	49
4	17	28	37	38	40	47	48
5	9	14	15	20	34	40	46
6	8	10	20	29	36	37	40
7	1	5	16	26	39	45	50
8	6	9	15	20	27	39	46
9	2	16	18	30	40	46	50
10	5	6	32	34	39	42	43
11	14	33	41	44	48	50	51
12	12	18	20	28	32	46	50
13	2	5	18	22	35	39	48
14	4	7	24	31	36	37	51
15	22	23	41	42	43	45	47
16	15	17	23	35	47	49	51
17	11	12	17	18	19	44	50
18	3	5	6	41	43	47	50
19	4	23	25	27	34	44	46
20	20	24	25	26	27	45	48
21	2	8	12	14	31	41	49
22	18	21	23	25	30	42	46
23	2	23	28	35	45	47	48
24	4	6	24	32	35	47	50
25	7	19	22	30	35	42	49
26	1	11	12	17	24	27	51
27	1	6	10	16	19	20	24
28	8	14	25	26	38	48	49
29	3	10	13	16	19	41	43
30	1	5	17	26	29	41	42
31	5	7	9	11	22	25	39
32	2	19	26	29	31	37	46
33	15	16	25	39	43	45	45
34	2	7	10	22	28	33	37
35	3	16	20	27	31	38	51
36	5	9	12	18	20	43	47

37	4	28	30	33	35	36	38
38	4	11	16	23	30	37	40
39	6	7	8	14	29	37	39
40	22	32	35	41	46	47	51
41	1	4	8	24	39	43	48
42	1	20	23	29	39	44	46
43	3	8	36	38	38	40	51
44	6	8	12	17	19	22	32
45	11	13	27	32	38	40	43
46	3	25	29	34	35	36	45
47	10	17	22	24	33	34	45
48	1	10	29	30	33	36	39
49	5	11	12	13	31	39	48
50	20	25	28	32	34	38	46
51	1	2	4	7	21	30	35
52	3	4	9	32	34	37	38
53	9	14	18	19	25	45	50
54	3	6	9	13	19	40	41
55	1	4	17	23	32	42	49
56	3	10	12	13	21	30	49
57	7	21	32	33	39	45	48
58	1	7	29	33	37	39	43
59	1	2	3	5	8	18	40
60	3	10	22	28	40	44	48
61	2	7	20	21	23	36	43
62	7	16	27	28	29	38	42
63	1	8	13	14	18	28	43
64	12	17	22	27	28	36	49
65	10	14	15	16	30	42	45
66	13	21	26	27	28	30	47
67	5	10	19	24	33	47	51
68	3	16	21	30	46	49	50
69	14	21	29	31	34	38	43
70	2	6	13	14	21	28	35
71	3	4	20	35	42	44	47
72	7	13	14	30	32	42	50
73	3	18	19	28	41	49	51
74	3	4	7	13	16	31	36
75	3	5	12	21	26	31	34
76	6	11	17	26	41	42	51
77	7	12	24	25	30	33	48
78	6	15	23	24	25	32	37
79	10	13	20	27	34	37	46
80	12	15	26	31	33	35	40
81	2	6	13	15	18	41	44
82	7	14	16	22	27	42	46
83	9	11	20	26	27	31	47
84	11	13	14	23	33	34	48
85	12	17	18	25	29	40	49

86	4	7	15	26	43	46	48
87	14	29	36	44	47	50	51
88	2	10	15	17	21	37	45
89	2	8	12	17	33	41	49
90	1	4	11	31	40	42	45
91	16	31	42	45	46	50	51
92	9	14	17	21	33	38	44
93	8	9	11	26	27	36	44
94	10	20	25	32	35	36	50
95	9	22	32	33	44	47	48
96	4	10	18	23	24	37	49
97	1	5	6	19	34	38	42
98	16	23	30	34	36	41	50
99	2	13	15	18	34	36	44
100	11	24	33	34	37	47	51
101	8	16	19	21	23	24	26
102	4	18	22	26	26	49	51

D

Test dataset

scenario id	link 1	link 2	link 3	link 4	link 5	link 6	link 7
1	11	13	22	25	40	44	47
2	7	11	24	26	38	42	44
3	1	14	31	38	40	44	51
4	4	16	20	22	32	35	42
5	6	7	9	21	32	47	48
6	5	9	23	24	25	34	51
7	9	21	26	30	41	43	47
8	1	3	4	6	32	36	45
9	7	14	16	18	19	23	36
10	1	4	6	8	14	32	39
11	6	22	25	40	42	49	51
12	20	27	35	44	45	47	51
13	1	6	11	12	34	36	37
14	4	16	23	24	35	43	49
15	20	34	45	46	49	50	51
16	1	4	12	15	31	37	51
17	3	6	13	22	38	42	43
18	2	3	15	20	21	29	46
19	5	12	13	14	38	42	46
20	5	28	38	43	47	48	49

E
Extended testing

Scenarios with 5 disrupted waterway sections

Table E.1: Scenarios 5 waterway sections disrupted

Scenario	Disruption 1	Disruption 2	Disruption 3	Disruption 4	Disruption 5	Summed priority value
1	2	12	19	43	46	13
2	5	19	32	35	46	19
3	11	29	35	40	44	22
4	1	15	38	47	50	12
5	1	24	38	42	44	12
6	5	15	36	38	44	10
7	6	20	27	31	33	22
8	6	10	35	39	40	26
9	12	19	35	47	50	19
10	22	35	36	37	40	47

Scenarios with 9 disrupted waterways sections

Table E.2: Scenarios 9 waterway sections disrupted

Scenario	Disruption 1	Disruption 2	Disruption 3	Disruption 4	Disruption 5	Disruption 6	Disruption 7	Disruption 8	Disruption 9	Summed priority value
1	1	21	23	28	36	44	48	49	50	20
2	1	5	8	12	25	26	41	49	51	13
3	5	7	9	23	26	28	34	43	51	26
4	6	11	22	23	25	27	35	38	40	57
5	4	5	11	19	24	32	35	36	42	29
6	4	5	14	19	23	25	30	45	51	41
7	4	14	17	22	25	26	29	33	42	73
8	3	9	17	20	24	27	48	49	50	27
9	7	16	18	19	24	30	36	39	41	46
10	1	2	5	7	8	14	19	34	44	38