UNIVERSITY OF TWENTE.

Master Thesis

Intelligent Decision Integration in Logistics: A Deep Q-Learning Approach to Refuelling, Cleaning, and Resting Operations in a Transportation Company

by

Gijs Renskers

Industrial Engineering and Management
Specialization Production and Logistics Management
Orientation Supply Chain and Transportation Management
Faculty of Behavioural, Management and Social Sciences

Examination committee
Dr. A. Asadi
Dr. B. Alves Beirigo
University of Twente

External supervision
Luca Overmars
Nijhof Wassink

2025

Abstract

This thesis investigates whether refuelling, cleaning and resting decisions on fixed multi-day routes can be improved by an integrated, data-driven decision aid for Nijhof–Wassink. The problem is modelled as a Markov Decision Process and addressed with a Deep Q-Network trained on historical three-day routes enriched with station and cost information. The resulting policy is evaluated against a heuristic, the current decision method and a Mixed-Integer Linear Programming benchmark. On the constructed test routes, the DQN reduces expected operational costs by 19.33% compared to the current decision approach and by 3.73% compared to the heuristic, with an average optimality gap of 13.40% relative to the MILP solutions. Scenario and sensitivity analyses indicate that the policy remains effective under changes in station availability and cost parameters, suggesting that the approach is a promising basis for an integrated decision-support tool in bulk logistics.

Preface

By reading this, you had the courage to dive into the realm of reinforcement learning and specifically Deep Q-Learning. When I started this thesis project, those terms felt just as abstract to me as they might sound now. Over time, the abstract concepts of Deep Q-learning turned from vague ideas into practical tools for improving decisions in the operations of Nijhof-Wassink.

This thesis marks the end of my Master's in Industrial Engineering and Management at the University of Twente. I could not have hoped for a better opportunity to develop relevant skills in reinforcement learning, but also in broader areas such as change management and collaboration between academia and industry.

A special word of thanks goes out to my supervisor at Nijhof–Wassink, Luca Overmars. You placed great trust in me throughout the process and gave me the freedom to explore the ideas I wanted to pursue. Our many sparring sessions provided valuable insights and helped me to approach the problem from different perspectives.

I would also like to express my sincere gratitude to my university supervisors. In particular, I wish to thank my first supervisor, Amin Asadi, for his continuous guidance, constructive criticism and careful supervision throughout the project. I am likewise grateful to my second supervisor, Breno Alves Beirigo, for his valuable suggestions and critical remarks, which helped to refine the work in the last phase of the project. Finally, I would like to extend my thanks to Abbas Maleki, whose regular meetings and engagement with the topic encouraged me to explore alternative ideas and approaches.

Finally, I want to thank my family and friends for their support, encouragement and their occasional reminders that there is more to life than Deep Q-learning. Their patience and understanding made the thesis period a lot more enjoyable.

Management Summary

Nijhof-Wassink's operating costs are dominated by personnel, refuelling and cleaning. Currently, decisions regarding cleaning, refuelling and resting are made independently, whereas integration of these actions could decrease the number of stops, avoid long detours and combine certain actions. This thesis therefore asks: "Can an integrated optimisation model improve cost efficiency and operational performance by deciding refuelling, cleaning, and resting jointly over a fixed multi-day route?"

Methodology

The problem is modelled as a Markov Decision Process (MDP) and solved with Deep Q-learning. A parametric Q-network is used to evaluate one state-action pair at a time, which allows training with a variable action set. Training uses full-episode Monte-Carlo returns in combination with a stratified replay buffer to avoid bias towards many small refuels. A Mixed-Integer Linear Program (MILP), a heuristic approach and the current decision method are used as benchmarks to evaluate the performance of the DQN model. The DQN model is trained on historical three-day routes in combination with station data, including prices and cleaning times.

Results

The evaluation shows that the DQN outperformed the current decision-making approach by 19.33% and the heuristic approach by an average of 3.73%. Comparison with the MILP approach revealed that there is a 13.40% optimality gap. Furthermore, the DQN approach uses fewer refuelling stops per route while still completing the routes, indicating a more efficient refuelling strategy. Scenario analysis shows that the policy remains effective when the number of refuelling stations is reduced by 50%. Sensitivity analysis shows that the model adapts smoothly to parameter changes suggesting that the state representation and training setup generalise beyond the exact environment of Nijhof-Wassink.

Contributions

This research provides an integrated decision support tool that combines refuelling, cleaning and resting choices without handcrafted rules which offers measurable cost improvements while remaining robust in a changing environment. Theoretically, this research provides a structured formulation for the Fixed Route Vehicle Refuelling, cleaning and resting problem (FRV-RCRP) and a scalable DQN design for large, non-stationary action and state spaces.

Recommendations

- **Long horizon validation:** Run a longer-horizon evaluation to compare the DQN with the current approach.
- **Route-calculation:** Integrate a route-calculation API to get accurate driving times and fuel consumption before implementation.
- **Scenario mapping:** Map all key operational scenarios and adjust exploration so each scenario is represented during training.
- **Graph transformer:** Implement a graph-transformer to capture truck-to-station and station-to-station geometry for better upfront combinatorial choices.
- **Robustness:** Test the model under uncertainties such as stochastic travel times, daily price shifts and station queues.
- **Method extension:** Experiment with various reinforcement learning techniques, such as PPO, to determine if model performance improves.

Conclusion

This thesis presents an integrated Deep Q-Network (DQN) decision aid for addressing the Fixed-Route Vehicle Refuelling, Cleaning, and Resting Problem. The DQN model learns key trade-offs between price, detour, and service without relying on hand-crafted rules. It outperforms the current approach by an average of 19.33% and a heuristic method by 3.73% across various test routes. The agent successfully reduces both the frequency of stops and the amount of fuel used per stop, while still maintaining effectiveness under scenario stress tests. Due to its region-agnostic state space formulation, the DQN model can be applied in different operational regions. Overall, these results position the approach as a credible and scalable candidate for pilot deployment and continuous improvement.

Contents

Acronyms

API	Application Programming Interface
CI	Confidence Interval
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
ETA	Estimated Time of Arrival
FRVRP	Fixed-Route Vehicle Refuelling Problem
FRV-RCRP	Fixed-Route Vehicle Refuelling, Cleaning and Resting Problem
GPS	Global Positioning System
KPI	Key Performance Indicator
LP	Linear Programming
MDP	Markov Decision Process
MILP	Mixed-Integer Linear Programming
MIP	Mixed-Integer Programming
NN	Neural Network
OSM	OpenStreetMap
OSRM	Open Source Routing Machine
ORS	OpenRouteService
RL	Reinforcement Learning
SD	Standard Deviation
SOTA	State of the Art
TSP	Travelling Salesperson Problem
VRP	Vehicle Routing Problem
VRTDSP	Combined Vehicle Routing and Truck Driver Scheduling Problem

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Company Description

Nijhof-Wassink Group is a leading logistics company specialised in transporting bulk goods from and to worldwide customers. Nijhof-Wassink strongly focuses on efficiency and reliability, aiming for safe and timely delivery of bulk materials. The group can be divided into two business Units (BVs): Wemmers and Nijhof-Wassink. Wemmers specialises in transporting human food and ensuring hygiene and safety standards. Nijhof-Wassink is further divided into three sectors: Feed, Chemicals and Fuel. The feed department transports animal feed, and the fuel department is responsible for transporting diesel and petrol. The chemical department is divided into two divisions, namely, DBL (Dry Bulk Logistics) and LBL(Liquid Bulk Logistics). DBL handles industrial powders such as plastic pallets, whereas LBL focuses on disinfectants and other liquid chemicals. Nijhof-Wassink operates a fleet of around 700 trucks and 900 trailers by a team of 700 highly skilled drivers. Nijhof-Wassink is primarily active in Europe, with most of the transport in Benelux, Germany, and Eastern Europe. Next to transportation activities, Nijhof-Wassink is active in the tank-cleaning services and the warehousing sector.

1.2 Research Motivation

In recent decades, massive trade globalisation has occurred. Global trade volume reached 28.5 trillion dollars in 2021, tenfold the amount in 1980, indicating an enormous expansion of global supply chain networks (UN Trade and Development, 2022). The transportation industry plays a crucial role in these supply chains, transporting goods globally. For bulk transportation companies, fuel, cleaning and resting decisions are key in optimising operational processes. Fuel costs alone can account for up to 50% of total operating expenses in road transport , making strategic refuelling decisions essential (MINISTERIO DE INDUSTRIA et al., 2006). Additionally, cleaning locations must be selected to comply with regulations while minimising the waiting time and cleaning costs. Strict driving and rest regulations increase the complexity of these decisions as they restrict the planning of the activities. Due to the significant differences in fuel and cleaning prices across Europe and the low profit margins in the bulk transportation industry, minor decision-making inefficiencies can severely decrease the profitability of bulk transportation companies (Gkatzoglou et al., 2024).

1.3 Problem Statement

1.3.1 Problem Identification

This section discusses the daily operational planning process at Nijhof-Wassink. After extensive interviews with employees in different departments of Nijhof-Wassink, four main decision categories have been established:

1. **Daily planning:** The daily planning entails assigning drivers, trucks and trailers to loading and unloading locations. Consequently, the driver chooses the route between the assigned locations.

2. **Refuelling :** Truck drivers must determine the location and timing of refuelling. Drivers may choose the refuelling location while trying to select the most cost-effective options.

3. **Cleaning:** Trailers must be cleaned to avoid contamination of freight due to residue left inside the tankers. The cleaning or swapping of trailers happens between the delivery and pickup of cargo. At the beginning of the day, each driver receives their route, including the pickup, delivery and cleaning locations.

4. **Resting:** Drivers must take regular breaks due to regulations and ensure safe driving behaviour. The driver can choose the resting location based on personal preferences while still adhering to the existing rules

The daily planning is currently done by the planning department and is deemed to be fixed. In contrast, no integrated decision-making procedure for refuelling, truck cleaning, and driver resting is implemented in the company's logistics. Therefore, drivers lack insight into the optimal choices, which could lead to short-sighted and sub-optimal decisions. Each of the three required actions is decided on independently, whereas a combined decision-making approach can reduce costs and improve efficiency. Thus, this research aims to determine whether an integrated solution approach can enhance the efficiency of operational decisions and decrease the total operational cost.

1.3.2 Problem Cluster

After extensive analysis, a problem cluster has been constructed, where the cause-and-effect relations of all problems are visualised. The problem cluster is depicted in Figure 8.1. The core problem is the lack of an integrated decision-making approach for refuelling, cleaning, and driver resting, which leads to sub-optimal decisions in these three factors. These inefficiencies lead to increased operational costs through several direct and indirect consequences.

Refuelling inefficiencies

Drivers may choose the refuelling location themselves, but due to the lack of insight into the consequences of their decisions down the road, refuelling locations are not optimised. Additionally, drivers lack visibility into gas stations along their route. The

result is that trucks may be refuelled at gas stations with high fuel prices, which increases fuel costs. Furthermore, truck drivers make too many stops, leading to high labour costs due to increased man-hours. An extra 15 minutes is charged for every refuelling stop that is made. Additionally, drivers must re-accelerate after stopping the truck, which consumes a significant amount of fuel, especially when the truck is fully loaded. Lastly, truck drivers make long detours to arrive at the chosen gas stations, further increasing labour costs and fuel consumption.

Cleaning inefficiencies

Unlike the refuelling locations, the cleaning locations are determined by Nijhof-Wassink's planning department. There are two options for determining these cleaning locations.

1. **Swap facility:** At this type of facility, the tanker can immediately be exchanged for a clean one. This allows the driver to continue the planned route. The original tanker is cleaned at a later moment
2. **On-cite cleaning facility:** The driver must remain at the station during the cleaning process and, if possible, assist with the cleaning process.

Not optimising the cleaning location can lead to trucks being cleaned at expensive cleaning companies, leading to increased operational costs. Furthermore, not using the swap facility as much as possible adds waiting time, which increases labour costs and further adds to the operational costs.

Resting times inefficiencies

Truck drivers are allowed to work a certain number of hours daily, which serves as a hard upper limit. Not adhering to these rules can lead to high fines. However, drivers are flexible in choosing their resting times earlier if this improves efficiency. For example, resting close to a gas station with low fuel prices can lower the length of detours and lower fuel costs to improve cost-effectiveness. The resting locations are often based on drivers' preferences about available facilities at the stops.

1.3.3 Relevance Of The Research

This thesis examines the joint decision-making process involved in cleaning, refueling, and resting. Financial data from Nijhof-Wassink indicate that the majority of total costs stem from personnel, refuelling, and cleaning expenses. Additionally, the company's profit margins are quite low. As a result, it is crucial to make decisions regarding cleaning, refuelling, and resting as efficiently as possible. By integrating these three actions, it may be possible to enhance efficiency and thereby reduce operational costs.

From an academic perspective, this study addresses a gap in existing research, which typically focuses on optimising only one or two of these decisions simultaneously, whereas this study aims to optimise all three. By quantifying decision performance and comparing it with other benchmark solution methods, this study provides a robust evidence for the proposed solution approach. A detailed explanation, along with examples, can be found in Section 8.2.

1.3.4 Research Goals

Several research goals have been established to guide the study and ensure valuable insights and information are obtained. The research questions in Section 1.3.5 are formulated based on this framework of goals.

1. Analyse and map the current decision-making processes and challenges at Nijhof-Wassink.
2. Explore existing literature's modelling techniques and model-solving approaches for integrating refuelling, cleaning and resting decisions and their advantages and disadvantages.
3. Examining different performance metrics and benchmark solutions to evaluate the performance of the chosen model.
4. Explore the possible added value of implementing or further researching the integration of refuelling, cleaning and resting decisions in the daily workflow of Nijhof-Wassink.
5. Evaluate and understand the proposed model's limitations and provide recommendations for its implementation and future research to Nijhof-Wassink.
6. Validate the constructed model, proposed solution and experimentation results.

1.3.5 Research Questions

To guide this research, a main research question and several sub-questions have been constructed based on the problem cluster and research goals. The problem-solving approach for the following research questions is discussed in 8.3.

Main Research Question

"How can the integration of refuelling, cleaning, and resting decisions in a single optimisation model improve cost efficiency and operational performance for Nijhof-Wassink Group's truck fleet?"

Sub-questions

Business Understanding:

1. What criteria, constraints and methods are currently used by Nijhof-Wassink to make refuelling, cleaning, and resting decisions?
2. What inefficiencies occur in the current decision-making process?

Literature Review:

3. What approaches for modelling integrated decision-making problems in fleet management exist in the literature?
4. What approaches for solving the chosen modelling technique exist in the literature?

Model Engineering:

5. What modelling approach best suits the problem based on the theoretical aspects of integrated decision-making in fleet management?
6. What solution method is most suitable for optimising the chosen model effectively?

Data Collection and Preparation:

7. What specific data must be incorporated into the model in order to effectively address the integrated decision-making problem?
8. What pre-processing steps are required to prepare the data?

Quality Assurance and Deployment:

9. What performance metrics should be used to evaluate the effectiveness of the selected models and solution approaches?
10. How can the different performance metrics be measured and validated in a real-world logistics setting reflecting the operational constraints and decision-making process of Nijhof-Wassink?

11. How do different modelling and solution approaches compare regarding the selected performance metrics, and which solution approach best suits the decision-making process of Nijhof-Wassink?

Monitoring and Maintenance:

12. What are the recommendations and limitations for implementing the proposed solution?

13. How can the chosen model be adapted so that it can be applied to the multiple operation areas in which Nijhof-Wassink is active?

1.3.6 Scope and Limitations

This research will focus solely on the DBL and LBL sectors as they share many characteristics and constraints and have more data available. The other sectors differ in operational processes and have limited data available. Additionally, LBL and DBL make up a large part of Nijhof-Wassinks' driving routes, and thus, a large part of the operational costs originates from these two sectors. The absence of generalizability in other sectors and the large operational cost share of LBL and DBL led to the research of these two sectors. This means that the proposed models and used data may need adaptations before being implemented in the other sectors of Nijhof-Wassink.

This research focuses on optimising refuelling, cleaning, and resting decisions in the Netherlands, Belgium and Germany. Nijhof-Wassink has several customers in other European countries, such as Spain, Italy, Switzerland and other European countries. However, these countries will not be included in the research due to the small number of orders and the absence of reliable data on gas and cleaning stations.

To realise a feasible model, some assumptions must be made. Examples of assumptions are average driving speed, weather conditions, driver preferences and driving costs per hour. Assumptions are kept to a minimum to ensure that the findings of this research are still applicable to the Nijhof-Wassink case study. The exact list of assumptions is presented in Section 4.2.2.

This study will focus on the short-term decision-making of refuelling, cleaning and resting locations and timing. The proposed model will function as operational advice for planners and drivers regarding real-time or near-real-time decision support.

1.3.7 Assessment of Validity and Reliability

Validity

Face Validity: Experts at Nijhof-Wassink will be asked for their opinions on the chosen solution and the corresponding KPI values. Experts from pre-planning, live-planning, finance, application managers, and several drivers will be consulted to view the solution from multiple perspectives.

Content Validity: All input data, constraints, and decision rules are based on Nijhof-Wassink's business process. Current decision rules and constraints are extracted based on interviews with experts. It is assumed that these experts have the company's best interests at heart, and thus, these claims are valid. The data is from a database used daily to make driving routes, cleaning, and fueling decisions. While errors may exist in the data, the daily use in operations indicates that the data is valid enough to use in the constructed model.

Construct Validity: The KPI values from the solved model will be compared to KPIs flowing from the current decision-making process to see if values are in the same order of magnitude. Detecting significant differences between the two sets of KPIs can indicate an error has been made in the model. Most sequential decision-making problems are sensitive to errors as they are based on multiple assumptions and constraints.

Reliability

The reliability of the research will be ensured by completing multiple experiments with the same input data and model settings. However, data science and especially machine learning techniques do not always yield consistent results, making it hard to establish the reliability of such research. To ensure the reliability of this research, the experiments are repeated and the variability of the outcomes is analysed which provides insight into the stability of the conclusions. Furthermore, the results will be presented to multiple employees working in different departments to mitigate observer bias.

Chapter 2

Context Analysis

This Chapter explains the current operational processes of the Dry Bulk and Liquid Bulk departments at Nijhof-Wassink. First, Section 2.1 explains the current transportation network and fleet management infrastructure. Subsequently, Section 2.2 explains the current operational planning process per decision category. Then, Section 2.3.1 provides a detailed cost breakdown, highlighting which components are decision-sensitive and therefore most impactful. Section 2.4 gives a summary of this chapter. The research questions that are answered in this Chapter are: *What criteria, constraints and methods are currently used by Nijhof-Wassink to make refuelling, cleaning, and resting decisions?* and *What inefficiencies occur in the current decision-making process?*

2.1 Transportation Network and Fleet Management

2.1.1 Routes and service areas

The DBL and LBL sectors of Nijhof-Wassink are active in large parts of Europe. Most customers have recurring loading and unloading requests at Nijhof-Wassink, for which a new price is agreed upon each time they place an order. In addition to fulfilling transportation requests from regular customers, Nijhof-Wassink also handles SPOT transportations. Customers can submit SPOT requests on a marketplace website, allowing Nijhof-Wassink to select and execute specific transportations, thereby reducing empty kilometres driven and keeping costs low. Data analysis of historical routes from January 1, 2024, to January 1, 2025, recorded 324 loading locations and 901 unloading locations. These include the regular and SPOT requests. The loading and unloading locations are referred to as events in this research, as these are the locations the trucks must visit. The route sections in between the events are referred to as segments.

2.1.2 Technology and Data Infrastructure

Several connected systems manage the fleet of Nijhof-Wassink. This Section explains the different systems and their functions, which are visualised in Figure 2.1. The interaction between the systems and the connected planning processes are explained in Section 2.2.1.

- **Navitrans:** Nijhof-Wassink utilises a transport management system for managing order requests. All orders, along with their respective loading and unload-

ing locations and time windows, are recorded in this system. Additionally, all invoices are generated and stored, and all pricing agreements are maintained.

- **Greencom:** This system tracks the GPS location and fuel level of the trucks and regularly sends updates to the planning software ORD.
- **ORD:** This is the planning software used by Nijhof-Wassink. Drivers are assigned to a truck and trailer, which are then assigned to specific loading and unloading locations. ORD also tracks the location of the trucks via the Greencom device.
- **Boardcomputer:** This device is a tablet mounted inside the truck and serves multiple purposes. Truck drivers can view their action list and provide updates on the status of their current tasks to the Nijhof-Wassink planning department. For example, a driver can indicate when they have started loading or unloading at a customer's location or when they are taking a break. Additionally, the driver can capture the Bill of Lading (BOL) using the camera on the board computer and send it to Navitrans and ORD for further processing.
- **Tachometer**: This device tracks the driving and resting times of every individual driver to ensure compliance with existing regulations.

2.2 Operational Processes

2.2.1 Planning of Transportations

Nijhof-Wassink utilises multiple software systems to manage daily operations and collect data on critical processes, thereby enhancing decision-making and supporting fleet analytics. Figure 2.1 depicts the system architecture by visualising the operational flow regarding the planning of a truck. When a customer places an order, information on the loading and unloading addresses, as well as the order execution date, is saved in Navitrans by the order intake department of Nijhof-Wassink. After registering the transportation request in Navitrans, the planning department creates a planning which entails assigning a truck, trailer and driver to several consecutive actions that span a maximum of three days. Behind this three-day horizon, Nijhof-Wassink has not received enough transportation requests to create an efficient planning. The planned actions typically involve loading and unloading locations, as well as cleaning advice after an unloading location. The loading and unloading locations are mandatory, whereas the cleaning locations are just a recommendation that the driver is not necessarily required to follow. During transport, the truck sends frequent updates to ORD with information on its current GPS location and fuel level. ORD then automatically calculates the estimated arrival time to the next location on the action list. The planning department then analyses if the rest of the planning is still feasible or if rescheduling is necessary. The planning can become infeasible if the driver is not on time at the next location due to the assigned time window constraints at the customer. Many customers of Nijhof-Wassink utilise time windows for loading and unloading to manage congestion at the loading and unloading docks. The planning department can reassign the next location to a different driver, ensuring the next time window is

met and Nijhof-Wassink avoids fines. The change in planning is communicated to the drivers through the board computer.

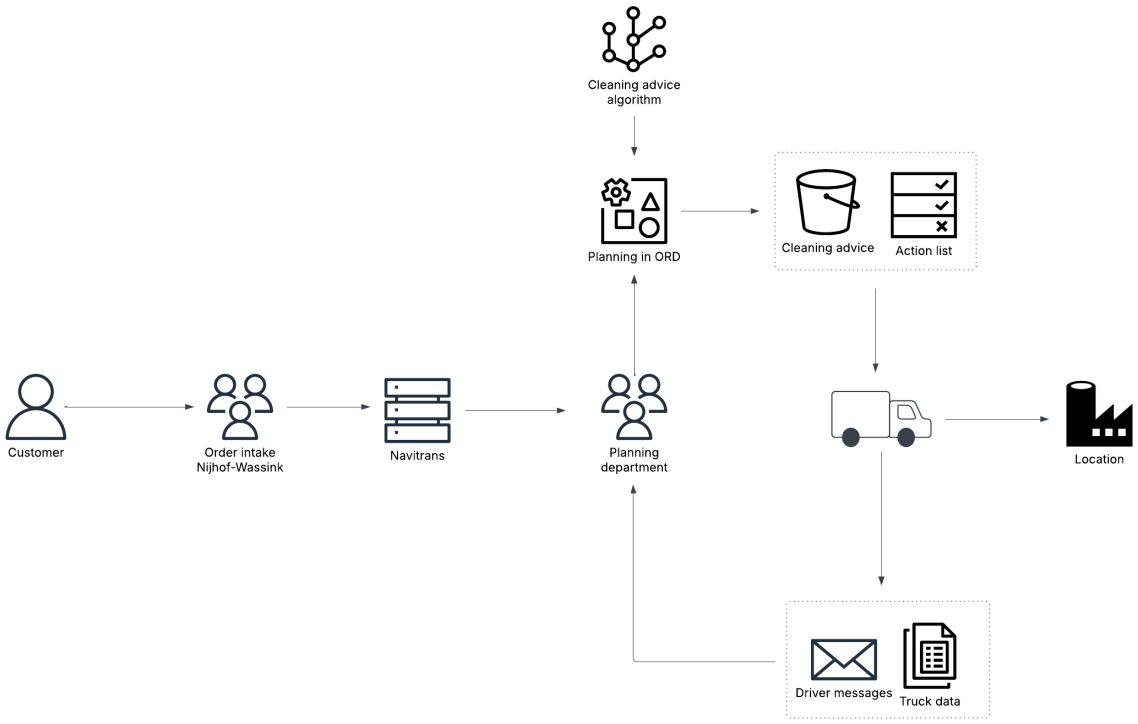


Figure 2.1: System architecture

2.2.2 Cleaning Process

After unloading a product, trailers need to be cleaned at a cleaning station. Cleaning can be done at various cleaning stations throughout Europe. However, not all materials can be cleaned at every station due to significant differences in cleaning difficulty among products. Generally, dry bulk products are easier to clean than liquid bulk products. This is primarily due to two factors: first, regulations for cleaning liquid bulk are stricter than for dry bulk; and second, some liquid bulk materials can be sticky, making them more challenging to remove from the containers. Commodity codes are assigned to various products to indicate the difficulty and duration of the cleaning process. The four commodity codes and corresponding cleaning hours are presented in Table 2.1. Utilising a swapping station always takes one hour in total; however, the original cleaning costs are incurred because the trailer still needs to be cleaned. There are significant price differences between the different cleaning stations, which are visualised in Figure 2.2. It can be seen that cleaning prices vary between €50 and €750. Choosing the right cleaning station is therefore of utmost importance for ensuring cost efficiency and maintaining competitive operational performance.

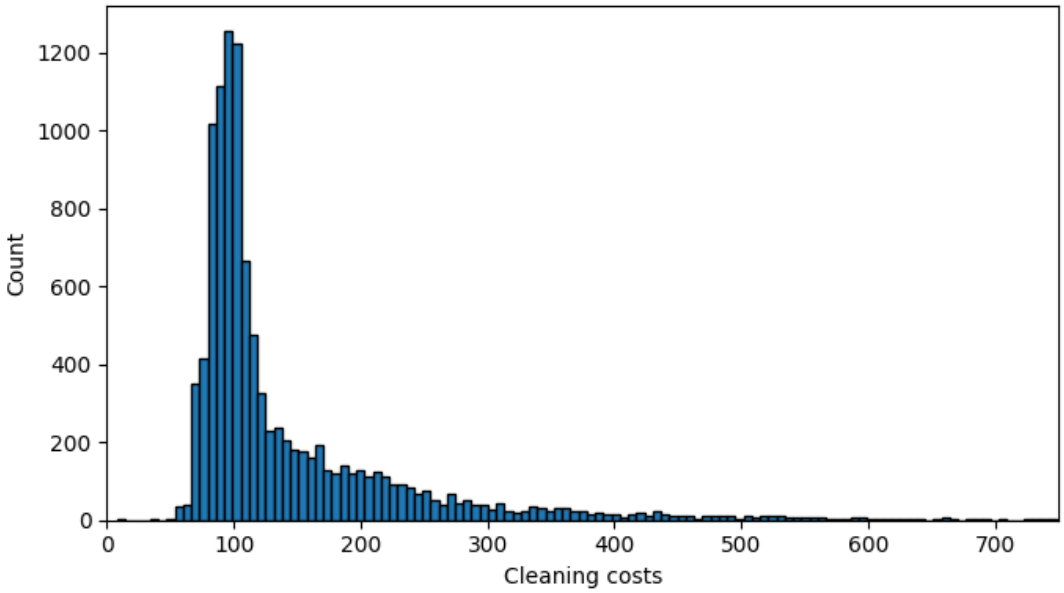


Figure 2.2: Cleaning costs distribution

Comodity code	Cleaning duration (hours)
EASY	1.5
MEDIUM	2
HEAVY	3
UNKNOWN	2

Table 2.1: Cleaning durations

Initially, the planning department chooses the cleaning location based on their expertise and implicit knowledge rather than using explicit rules. For liquid bulk, the location choice is relatively easy, as not many companies can handle cleaning liquid products. This leaves little choice, and the cheapest station is often chosen from the feasible options. In contrast, the choice for dry bulk cleaning stations is more complex due to the greater number of viable cleaning stations caused by the ease of the cleaning process. One challenge is determining whether it is worth travelling a greater distance to access a cleaning process that offers a lower price. Therefore, an additional check in the form of a cleaning station selection algorithm has been implemented by the application management department of Nijhof-Wassink. At the start of the day, the feasible cleaning stations for every truck in a segment are retrieved. Then, a cost calculation is made for reaching every individual cleaning station that is present within X kilometres of the route between the unloading point and the next loading point. This number is divided by the actual cleaning costs paid at that station to analyse whether the savings in cleaning costs exceed the extra costs incurred for the detour. The most economical option is then selected and communicated to the drivers as advice. The main limitation of this method is that it evaluates cleaning stops in isolation from the other two decisions. As a result, It may overlook opportunities to save costs by combining cleaning actions with refuelling at a cheaper gas station.

2.2.3 Refuelling Process

In contrast to cleaning decisions, no advice is given on the refuelling location. Currently, drivers choose their own refuelling location, often based on the base fuel price represented at the gas stations and their expertise throughout the years. The base fuel price excludes any discounts or tax refunds specific to that gas station, leading to decision difficulties among truck drivers and suboptimal refuelling behaviour. In addition to the lack of real-time fuel price information, drivers may also face other challenging situations. For instance, if a truck is low on fuel and only high-priced stations are nearby, while cheaper stations are further away but currently inaccessible, the driver encounters an operational dilemma.

1. Take a small refuel at the expensive station to ensure reachability of the cheaper one down the route. Then, make a second stop to refuel at the cheaper station fully, utilising the lower fuel price but incurring extra detour and stopping costs.
2. Fully refuel now at the nearby expensive station to avoid detour and stop costs of this second fuel stop, but incur high fuel costs.

Another dilemma arises when considering a detour to a cheaper gas station. In general, there are two options:

1. Take a detour to refuel at a lower price, increasing the total distance travelled, consuming additional fuel, and potentially raising toll costs. However, the savings on gas prices may offset the increase in other expenses.
2. Wait for a gas station on the route with a potentially higher fuel price, but avoid the extra distance and delays.

In both dilemmas, the chosen action in the current stage has consequences for possible actions in later stages, making optimal decision-making even more difficult. These two situations do not describe the entire decision space. In practice, many other configurations may arise, for example in combination with the other two action types: cleaning and resting. These two are presented to illustrate the difficulty of the decision-making process.

Price agreements

There are significant fuel price differences across European fuel stations. Two main price calculation methods are used in the Netherlands and the rest of Europe. The fuel prices in the Netherlands are determined by the Platts prices, which serve as a benchmark for trading crude oil, gas, and metals. Nijhof-Wassink receives a new Platts price at the start of each week. The fuel price paid by Nijhof-Wassink is calculated by adding a certain premium based on the region in which the gas station is located. The premium that is paid on top of the Platts price is the same for most gas stations, namely €4.75/100L. The other gas stations have premiums ranging between €3.75/100L and €22/100L.

Regarding the fuel prices in the rest of Europe, Nijhof-Wassink has a price agreement with BP and its subsidiary companies. European countries are divided into regions on

which a rebate is based. The fuel prices per region are calculated by subtracting the rebate from the base price. Belgium is unique in that it offers both a rebate and a tax refund, resulting in fuel prices that are generally the lowest. The average fuel prices from 2023-01 until 2025-01 of the countries in the scope of this research are visualised in Figure 2.3. It is evident that there are significant differences in fuel prices across various countries.

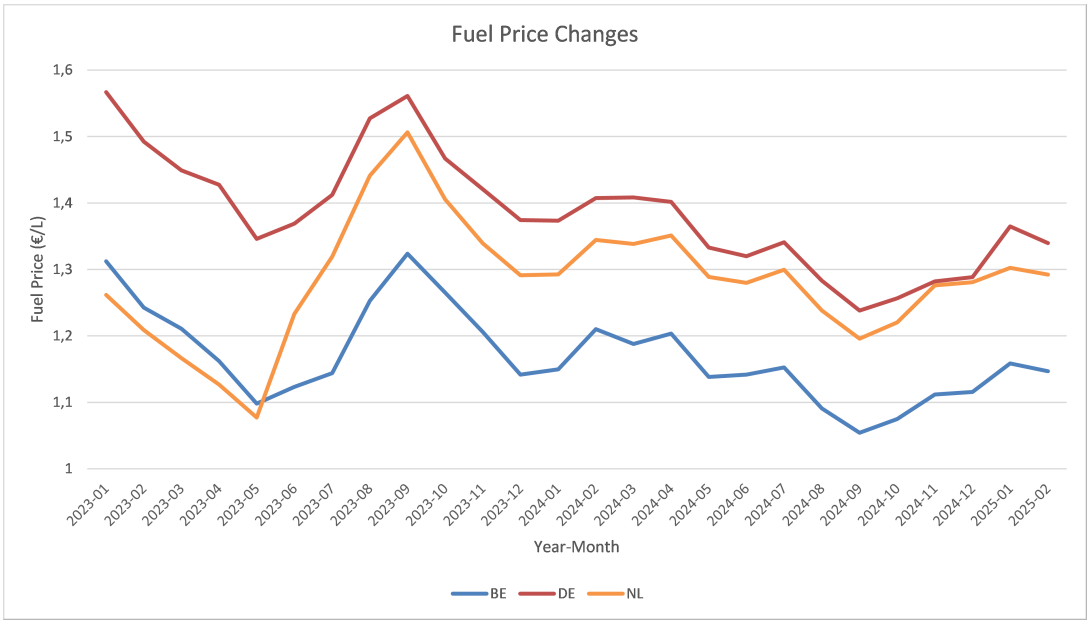


Figure 2.3: Fuel prices NL, BE and DE

In addition to the variations in fuel prices between countries, there are also significant differences in prices within individual countries. Figure 2.4 shows the fuel price spread within the different countries on 2024-12-15.

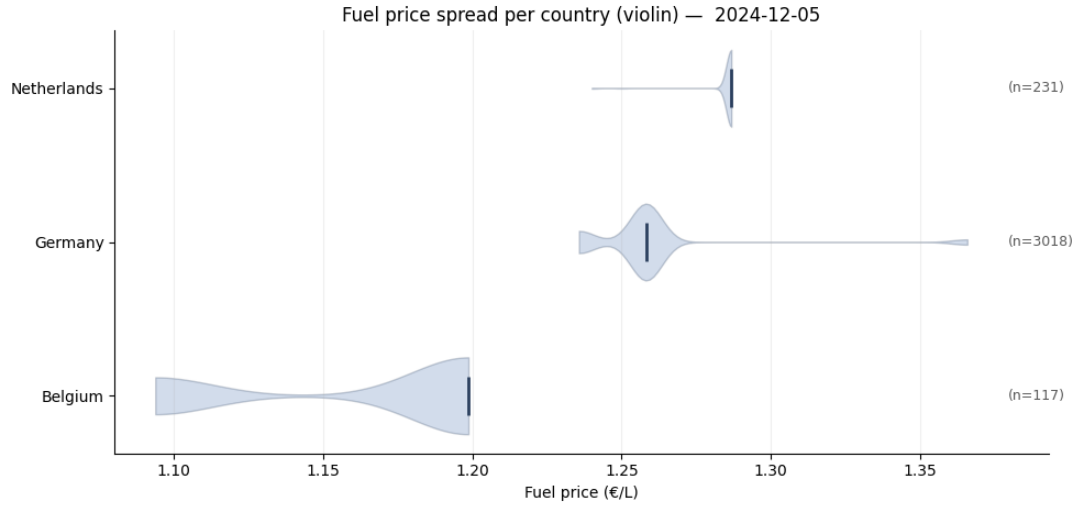


Figure 2.4: Fuel price spread

Refuelling behaviour

The fuel tanks of the trucks used by Nijhof-Wassink have a capacity of 550 litres. Approximately 1,500 kilometres can be travelled with a full tank, depending on external factors such as weather conditions, traffic, driving style, and road conditions. The average amount refuelled by drivers in the DBL and LBL sectors is approximately 284 litres, which seems suboptimal given the truck's 550-litre tank capacity. Refuelling when the fuel level is lower would be more beneficial, as it allows the driver to avoid frequent stops. The histogram in Figure 2.5 shows that the mode of the refuelling amount is between 330 and 360 litres.

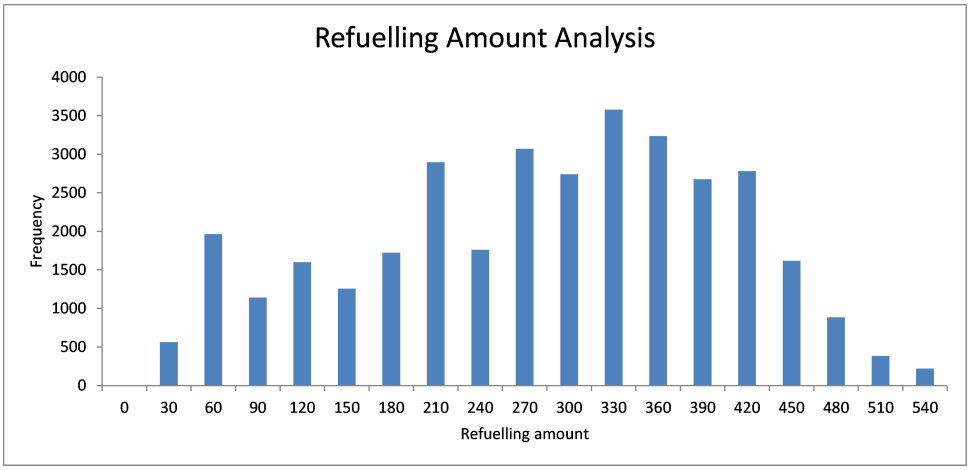


Figure 2.5: Refuelling amounts histogram

2.2.4 Resting Process

Truck drivers must adhere to the driving and rest regulations of the European Union. This research focuses solely on decisions related to nighttime rest, and therefore, only regulations affecting these decisions are presented. The following rules apply to Nijhof-Wassink's drivers.

- Daily driving shall not exceed 9 hours, with an exemption of twice a week when it can be extended to 10 hours
- Total weekly driving time may not exceed 56 hours, and the total fortnightly driving time may not exceed 90 hours
- Daily rest period shall be at least 11 hours, with an exception of going down to 9 hours maximum three times a week. Daily rest can be split into 3 hours of rest followed by 9 hours of rest to make a total of 12 hours of daily rest.

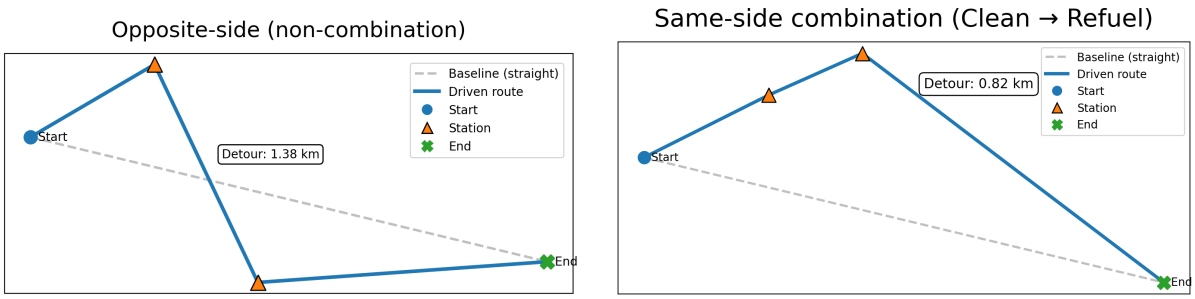
The driving times are recorded by a tachometer installed inside the truck's cabin. Drivers cannot alter the recorded data, which is automatically sent to the hour-controlling department of Nijhof-Wassink. The police conduct random checks both on the road and at the Nijhof-Wassink office to ensure compliance with driving and rest regulations. Any violations of driving regulations may result in significant fines. Therefore, it is crucial to adhere to these rules.

2.2.5 Feasibility of Combining Actions

In the context of route execution, a combination action is defined as a deliberately paired sequence of cleaning refuelling or resting stations.

- **Cross-station combinations:** Directly driving between a cleaning and resting station or between a cleaning and refuelling station on the same side of the route segment.
- **Co-location combinations:** Performing a resting and refuelling action at the same station.

The same-side restriction for the Cross-station combination is essential, only then the vehicle does incur one off-route detour rather than paying two separate detours for two independent stops on opposite sides of the route. This difference is visualized in Figure 2.6



(a) Combination on opposite side

(b) Combination on same side

Figure 2.6: Action combination example

Although Nijhof-Wassink does not actively combine the three different actions to achieve higher operational efficiency, there is clear potential to do so. Gas and cleaning stations are often in close proximity, allowing for combination actions which reduce detour distances, waiting times, and overall operational costs. Figure 2.7 illustrates the spatial distribution of refuelling and cleaning stations that are located within a 10-kilometre radius of each other, highlighting regions where such combinations are feasible.

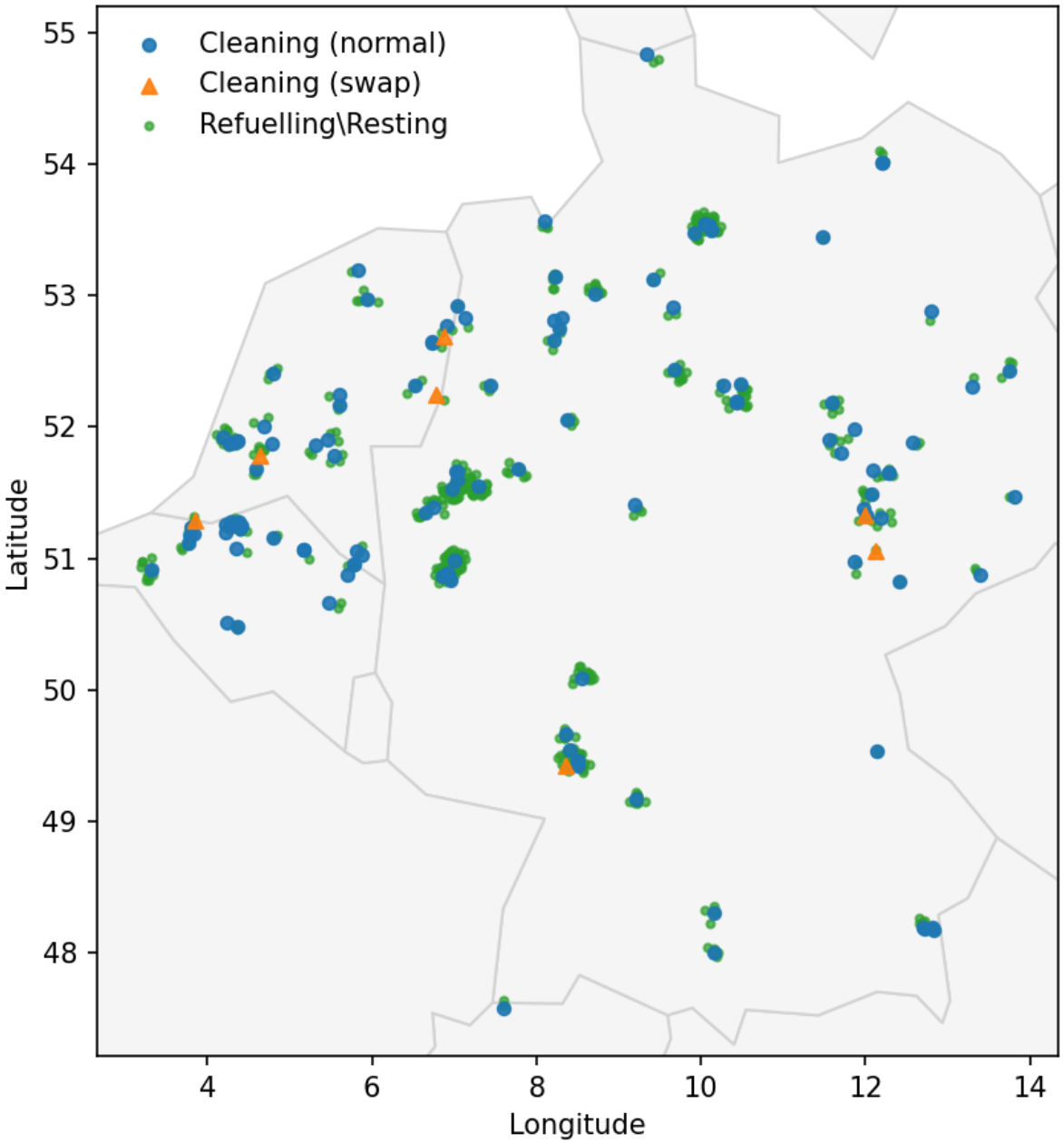


Figure 2.7: Station proximity map

2.3 Critical Decision Factors

2.3.1 Cost structure

To gain insight into the factors that drive operational costs, a cost breakdown is visualised in Figure 2.8. There are four main cost components: fuel, cleaning, toll, and labour costs. The green boxes represent costs that can be influenced by adjusting the refuelling, cleaning, and resting locations, as well as the amount of fuel purchased per refuelling action. In contrast, the red boxes represent costs that are unaffected by these decisions.

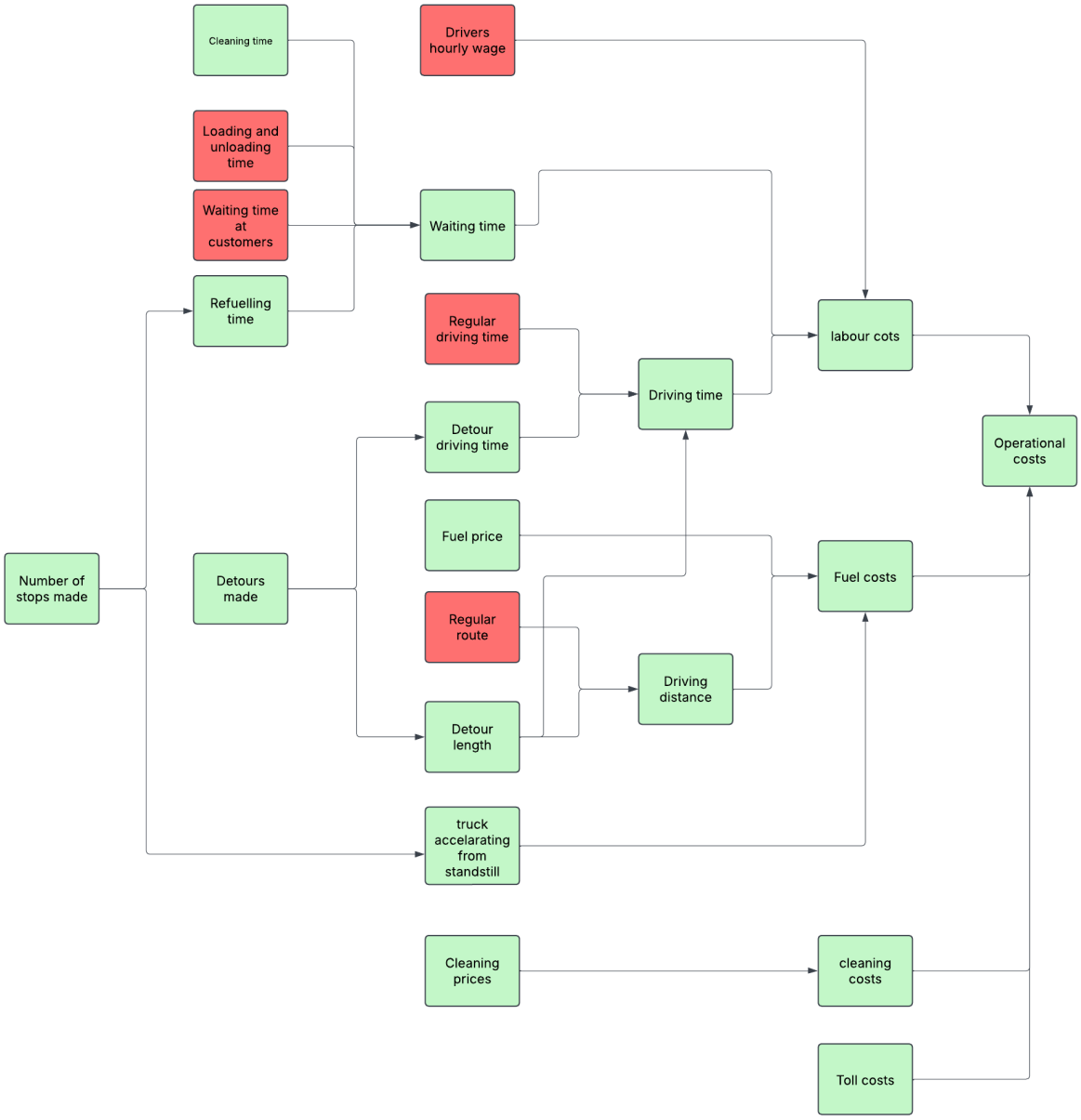


Figure 2.8: Cost Breakdown Nijhof-Wassink

To charge customers with well-considered costs and analyse current operational processes, some variable values have been generalised per department. A standard cost per kilometre and hour for the two sectors has been established by Nijhof-Wassink. The driving costs are set to €X per hour and €X per kilometre. The price per hour is calculated based on the hourly wage of a driver, truck depreciation, taxes, insurance, and overhead costs, and is used to determine the costs of driving and waiting time. The price per kilometre is calculated using an average maintenance cost. It is then multiplied by the total kilometres driven in a segment to determine the driving costs for that lane.

The second component is European road tolls. However, these are excluded from the scope of this research due to limited data availability.

Cleaning costs incurred between unloading and loading should also be included in the operational expenses. These costs vary depending on the material being cleaned and the specific cleaning station utilised. Even if the same material and cleaning station are involved, different prices may be charged by the cleaning company due to external factors such as temperature and humidity. For instance, when birch bark is transported in low temperatures, its viscosity increases, making the cleaning process more challenging. This can lead to higher cleaning costs imposed by the cleaning company. An average cleaning cost per material and cleaning company is calculated. This cost list is not included in this research due to the numerous combinations of cleaning stations and products.

2.3.2 Cost-Sensitive Decision Levers

Section 2.3.1 points out that there are are several important cost factors that influence the total operational costs of the transporation fleet of Nijhof-Wassink. Alongside that, the review on the current planning logic highlights decisoin points where choices can significantly affect cost. Together, these observations point to the followning cost-sensitive areas that are used later for validation of the decision policy:

- **Fuel price selection:** The difference in fuel price arccos different countries stresses the importance of the ability of the decision policy to exploit station price differences.
- **Detour management:** Driving costs per hour and kilometre make it essential to weigh added distance and time against potential savings from cheaper stations.
- **Bridging refuels:** The identified The ability to detect bridge-refuels and act on them.
- **Action Combination:** Recognizing stop and detour overhead costs motivates combining refuel, rest and cleaning actions at the same station or at nearby facilities when possible, to improve efficiency.
- **Stopping frequency:** Reducing stopping frequency to avoid repeated fixed stop overheads and cumulative detour distance.

2.4 Summary

This Section describes the business context of the DBL and LBL sectors of Nijhof-Wassink, which are the focus of this research. The transportation network, including service areas and specific loading and unloading, was outlined. Subsequently, the current technological infrastructure, including Navitrans, ORD, Greencom, the board computer and tachometer, was mapped. The primary operational processes regarding cleaning, refuelling, and resting were discussed, outlining decision difficulties and price differences. The selection of refuelling, cleaning and resting currently heavily relies on the expertise of employees and truck drivers, which is not always a guarantee for the best decision outcomes. Therefore, a structured planning approach is needed to ensure robust decision-making. Finally, an analysis of the cost structure of Nijhof-Wassink was conducted. It was determined that labour, fuel, and cleaning costs are the primary components, indicating that the most significant potential savings could be achieved in these areas.

Chapter 3

Literature Research

This Chapter discusses the theoretical context of the problem faced in this research. This literature research is split into seven sections. Section 3.1 aims to discover the problem landscape of the problem faced at Nijhof-Wassink by analysing its characteristics and comparing them to existing literature. Due to the limited research on combining the three decision factors into a single decision optimisation model, each will be researched individually. Section 3.2 describes the Fixed Vehicle Routing Problem and modelling approaches, as this type of problem is most related to the problem faced at Nijhof-Wassink. Subsequently, the broader theoretical landscape is discussed in Section 3.3. In Section 3.4, the application of the sequential modelling approach, specifically Markov Decision Processes, is explained, along with its potential applications in transportation problems. Section 3.5 dives deeper into different solving approaches. Section 3.6 classifies the problem in this research within its own group of problems. 3.7 gives a summary of this Chapter. This Chapter aims to answer the following research questions: *What approaches for modelling integrated decision-making problems in fleet management exist in the literature?* and *What approaches for solving the chosen modelling technique exist in the literature?*

3.1 Problem Landscape and Background

3.1.1 Refuelling Optimization

Fuel costs can generally be decreased in three ways. The first approach indirectly lowers fuel costs by minimising the distance trucks travel during a given period. The second approach aims to minimise the sum of the fuel costs spent during a certain route or period. A third approach combines the two approaches that aim to reduce both travelled kilometres and spent fuel costs.

Problems that aim to minimise the travelled distance are often referred to as Vehicle Routing Problems (VRP) or Travelling Salesman Problems (TSP) in the existing literature. A TSP involves minimising the travelled distance of a single vehicle which must visit a set of locations, after which it returns to the starting point (Dantzig et al., 1954). TSPs are NP-hard problems, meaning no polynomial-time algorithm exists to solve all instances of a TSP. The solving time of such problems exponentially increases with the number of locations, making most real-life problems complicated. Variants of the TSP are symmetric Travelling Salesman Problems (STSP), where the distance between cities

is the same in both directions or asymmetric Travelling Salesman Problems (ATSP), where the distances between cities depend on the travel direction. Padberg and Rinaldi (1991) solved large instances of the STSP to optimality using a branch-and-cut algorithm. Choi et al. (2003) used a genetic algorithm to solve the ATSP to near optimality or, in some cases, optimality.

A VRP is an extension of the TSP, which considers multiple vehicles that must visit numerous locations (Dantzig and Ramser, 1959). The aim is to create a set of routes that minimises the travelled distance. Many variations based on the nature of a problem exist in the literature. Examples include Vehicle Routing Problems With Time Windows (VRPTW) and its variations, such as those with a constraint on the maximum number of vehicles to be used (m-VRPTW). Lau et al. (2003) provided a computable upper bound as a benchmark and proposed a tabular search to solve the m-VRPTW.

In the Fixed Route Vehicle Refuelling Problem (FRVRP), the route points are predetermined, and the main objective is to minimise fuel expenses by optimising the refuelling strategy for a truck. The route for the trucks is already determined beforehand thus, decisions regarding the refuelling location and refuelling amount must be made. The assumption that fuel prices differ between gas stations makes these decisions crucial for cost efficiency. This type of problem is closest to our problem formulation, as the Nijhof-Wassink planning department has already fixed the routes, with the main difference being that the FRVRP problem does not include decisions regarding cleaning and resting. The FRVRP models are researched more extensively in Section 3.1.3.

A study by Khuller et al. (2011) introduces four different instances of a combined problem involving the TSP and FRVRP. Optimisation heuristics are used to solve the four different instances. An addition to the combination of the TSP and FRVRP is the Travelling Salesman Problem with Time Windows and Refuelling (TSPTWR) (López-Ibáñez et al., 2013). In current transportation networks, time windows are frequently encountered; thus, a time-restricted approach can improve real-world applicability. Suzuki (2012) created a decision support system to optimise the route and fueling decisions of a truck, including time windows that are present at the visiting locations. The problem is divided into two sub-problems, each solved using a combinatorial simulated annealing algorithm. The author initially proposes to solve the two sub-problems sequentially. However, this has the drawback of ending up in local optima instead of the global optimum. An alternative approach stores the M best feasible tours from the simulated annealing algorithm in the first problem, which are all considered in the second solving phase to increase the chance of ending up in a global optimum.

3.1.2 Resting Optimization

Assigning routes to truck drivers is a common problem in the transportation industry. Drive and rest regulation for truck drivers adds complexity to creating an efficient schedule that adheres to the rules. Goel and Irnich (2016) explains that many studies on the previously mentioned VRPTW problems do not include these drive and rest regulations, which causes infeasible solutions when implemented in real-world applications. The Combined Vehicle Routing and Truck Driver Scheduling problem

(VRTDSP) includes drive and rest regulations in the classical VRP problem to better align with real-world scenarios. Goel and Irnich (2016) proposed a branch-and-price algorithm for the VRTDSP problem. Xu et al. (2001) formulated the problem as a set partitioning formulation and used standard column generation to solve a linear relaxation. The master problem was then solved with an LP solver. Heuristics solved the resulting sub-problems. Other approaches by Archetti and Savelsbergh (2009) and Goel and Kok (2010) demonstrated that the problem can be solved in exponential time, proportional to the number of customers. Thus, no exact solutions can be found for large instances of the problem. Goel and Irnich (2016) are the first to apply an exact method for solving the VRTDSP problem. Several experiments with different numbers of customers were executed. A branch-and-price algorithm is used to solve instances with 25 customers and several instances with 50 to 100 customers. The number of cases per customer varied based on customer locations, demand, and model constraints. The VRTDSP (Vehicle Routing Decision Support Problem) involves creating complete routes that include all visiting points. This is different from the problem addressed in this research, where the routes are already predefined. Therefore, we cannot classify this research within the VRTDSP category.

Melo et al. (2016) aimed to schedule rest periods and breaks for a fixed route. A Mixed-Integer Linear Programming (MILP) model with two distinct objective functions was employed to address the problem, and a lexicographic approach was applied to the two objective functions to solve the MILP. Bernhardt et al. (2017) builds upon Melo et al. (2016) by also including refuelling decisions in addition to the resting and driving regulations. This includes the timing and location of refuelling and the amount per refuelling action on a fixed route. The lexicographic solution method is applied to three separate sub-models to solve the problem. The resting problem in this research can be best categorised as such.

3.1.3 Cleaning Optimization

Cleaning has received comparatively little explicit attention in road transportation optimisation. However, some related works are discussed. A study by Gifford et al. (2018) optimises multiple facets of a bulk tanker dispatch problem. Challenges regarding washing locations, tanker trailer assignments, route optimisation and driver assignments arise from chemical compatibility constraints and driver scheduling rules. To address these complexities, a multiphase optimisation framework is constructed to allocate resources efficiently. The research uses mathematical programming, column generation and heuristics to find solutions. Lahyani et al. (2014) studies the multi-compartment vehicle routing problem in which compartments may require cleaning before carrying a different product. A Mixed Integer Linear Program (MILP) is formulated and solved using the branch-and-cut method. This MILP takes into account cleaning times and costs while optimizing routing and assignment decisions. The paper presents a template for representing cleaning triggers, scheduling, and costs within road transport, making it relevant to this thesis.

3.2 Fixed Route Vehicle Refuelling Problems

The cleaning, refuelling and resting optimisation problem faced at Nijhof-Wassink can best be classified as an FRVRP problem, as the planning department already determines the truck routes. The FRVRP solely focuses on refuelling decisions. However, cleaning and resting decisions share similar characteristics with refuelling choices as they involve possible locations with corresponding prices. This Section will thus elaborate more on the different variants and solution approaches of FRVRPs. Four overarching variants of the FRVRP can be distinguished based on characteristics of the modelled routes and associated constraints.

On-route, no minimum refuel requirement

The most basic version of the problem is called the Simple Fixed Vehicle Refuelling problem (SFRVRP) and was introduced by Hong Lin et al. (2007). This problem assumes that all refuelling stations are located on the route and that no deviations from the original route are allowed. Furthermore, no minimum refuelling quantity exists. This problem aims to minimise the total cost of a single truck driving from starting point s to ending point t while always respecting the fuel tank capacity. This version of the problem is polynomially solvable. Hong Lin et al. (2007) showed that the SFRVRP can be solved in linear time by modelling the problem as an inventory-capacitated lot sizing problem.

Off-route, no minimum refuel requirement

This variant extends the simple FRVRP, with the main difference that it allows the truck to deviate from the fixed route. In the literature, this problem is commonly known as the "gas station problem". The objective is to minimise fuel costs in combination with detour costs. Although the stations are not directly on the route, the problem remains polynomially solvable. Khuller et al. (2011) proposed several solving algorithms where a detour cost is incurred when the truck leaves the main route path. Papadopoulos and Christofides (2017) later provided faster solutions for this version of the problem. Lin (2008) found algorithms with complexity $O(n^3)$ for the general variant and $O(n^3 log(k))$ in case the number of refuelling stops is bounded by k.

On-route, minimum refuel requirement

A third variant of the problem involves refuelling decisions on a fixed route, including a constraint on the minimum amount if a refuelling action is performed. The minimum fuel requirement breaks the continuity property of the previous two problem variants, which makes the problem non-convex. The non-convexity causes the problem to become NP-hard. Suzuki (2014a) mentions the NP-hardness of this problem instance, while Kim and Pardalos (1999) demonstrated the same fixed-charged structure in the broader Fixed-Charge Network Flow problem (FCNF), which is well known to be NP-hard in the literature.

Off-route, minimum refuel requirement

This variation is commonly known as the "complex" or "general" FRVRP. In this variant, refuelling actions can be conducted at stations located off-route, necessitating a detour to reach the station. Again, the minimum refuel requirement makes the problem NP-hard. Several attempts have been made to decrease the computational time of the complex FRVRP. Suzuki (2014b) used a variable reduction technique to reduce CPU time by 75%. Schulz and Suzuki (2022) used a set of bounds and inequalities and managed to solve the complex FRVRP to optimality, reducing the CPU time drastically. An example of an applied bound is the dominance criteria, where stops are eliminated based on a specific set of rules that ensure a stop is always inferior to the rest of the stops in a set. This significantly reduces the number of stops and improves computational time. Schulz and Suzuki (2022) managed to reduce the average computation time by 78% and for large instances of 2000 a reduction of 95% was achieved. This approach enables the resolution of fueling issues for large-scale transportation companies.

3.3 Wider Theoretical Landscape and Practical Context

This section discusses the problem in a wider context and highlights the gap between theoretical optimisation models and their practical applicability. While these studies are not directly utilized in this research, they help in understanding the usability of the proposed solution and provide a foundation for future research proposals.
Lin (2014) researched the trade-off between driving distance and fuel prices with a multi-objective mixed integer linear program. The author also distinguishes between arbitrage-free models, where the truck can visit gas stations once, and models without the arbitrage constraints, where trucks can visit gas stations multiple times. Lifting the arbitrage-free constraints in this multi-objective problem exponentially grows the number of constraints with the number of locations, thus making the problem NP-hard.

Many of the fuel optimisers used in industry are built on variants of the complex FRVRP. However, transportation companies are hesitant to implement such optimisers for two reasons. Finding an optimal solution for large instances of the complex FRVRP takes considerable time. This creates a problem as trucking companies have to optimise the decisions every day for multiple trucks. Furthermore, fuel optimisers have difficulty finding feasible solutions regarding the exceedance of tank capacity. (Schulz and Suzuki, 2023)

A concern raised by Suzuki (2009) is the potential employee satisfaction issues associated with imposing refuelling decisions on truck drivers. Two main problems are determined. The first is that the low compliance rates of truck drivers make the fuel optimisers less profitable, as drivers ignore advice regarding the refuelling location. The advice on the amount of fuel to be purchased is less of an issue for truck drivers. A second problem is the increase in the (already high) driver turnover rates in the transportation industry due to dissatisfaction among the truck drivers. Truck drivers often have their favourite resting and refuelling stops because of the available facili-

ties. Acquiring and hiring new truck drivers is an expensive process that could diminish the cost efficiency gained from utilising a fuel optimiser. Suzuki (2009) uses two methods to advise drivers on the refuelling amount and location, without confiscating their decision freedom. Based on the resting location chosen by the driver, advice is given to either refuel before or after midnight, assuming that the fuel price changes at that exact moment. Then, a min-max approach is used to determine the amount to refuel. The advantage of such a model is that advice is provided in real time, and thus changes in the driver's route do not affect the usability of the advice. In fuel optimisers used in industry, refuelling advice is based on the optimal route. A change in the planned route due to traffic or other external factors means that the advice is no longer applicable. A disadvantage of the approach by Suzuki (2009) is the limited capability to evaluate stops with possible lower fuel prices other than the location chosen by the driver.

Suzuki et al. (2014) and Suzuki and Lan (2018) both emphasise the fact that fuel usage is not always a linear function during a route and is based on factors such as fuel tank levels and slope steepness. The former article investigates the benefit of retaining space in the tank when refuelling to improve a truck's fuel economy. The latter paper examines the profitability of avoiding refuelling before hills or mountains to reduce fuel consumption and enhance cost-efficiency.

Studies not directly related to the problem faced at Nijhof-Wassink are the fixed-route vehicle charging problem (FRVCP) by Montoya (2016), a shortest-route charging problem for plug-in hybrid vehicles by Arslan et al. (2015) and studies on the optimal location for gas stations by Capar et al. (2013) and Hwang et al. (2017).

3.4 Modelling Approaches

Mixed-Integer Programming (MIP)

Mixed-integer programming is a class of mathematical optimisation models that can model the problem at hand and find an exact solution. While MIPs can effectively solve smaller problems, the complexity of the current problem is significantly greater. The computational complexity of the problem increases with the number of decision possibilities. Due to the large number of decision possibilities in this research, an MIP approach is disregarded as the modelling approach. Furthermore, standard MIPs do not include the modelling of possible uncertainties

Even if a relatively good decision is achieved, a small disruption in the planning of a truck could make the solution infeasible, necessitating a complete recalculation of the entire solution. This is because a different truck with a different fuel level might be needed to pick up and deliver a load due to unforeseen circumstances.

Stochastic Programming

Stochastic programming extends the Mixed Integer Programming (MIP) approach by incorporating uncertainty through scenario trees. Each decision configuration is as-

sessed across all possible scenarios, resulting in an expected quality evaluation for specific decisions. However, the computational complexity increases significantly, as each decision configuration from the MIP must be evaluated against multiple scenarios to account for uncertainty. As a result, the computational complexity is even higher than finding an exact solution to an MIP.

3.4.1 Markov Decision Processes

A modelling technique often used in sequential decision-making problems is the Markov Decision Process (MDP) (Puterman, 1990). MDPs represent an environment with states, actions and rewards. An agent finds itself in a certain state, and in every state, a set of actions is available from which the agent must choose one. The selected action leads to a transition into a new state; a reward is incurred afterwards. The state transition can be either deterministic or probabilistic, depending on the problem's characteristics. A transition probability function defines the probability of moving to a specific state s' given the current state s and action a. The reward that is incurred is determined by the reward function. A distinction can be made between finite-horizon and infinite-horizon MDPs. Infinite MDPs have a fixed number of time steps over which the rewards should be maximised. These are often used in problems with clear deadlines, such as route planning. Infinite-horizon MDPs have no fixed endpoint and, thus, are often used in continuous operation systems such as traffic control. An MDP aims to find a policy that maximises the cumulative reward over time (Puterman 1990).

Markov Decision Processes (MDPs) have been utilised to model and address various issues in operations research. In inventory management, MDPs have been used to determine optimal pricing levels for perishable products, as demonstrated by Moshtagh et al. (2025). Additionally, MDPs have been employed to optimise purchase quantities and timing, as shown in a study by Cuartas and Aguilar (2022). They have also proven to be effective in solving production scheduling challenges, including flexible job-shop scheduling, as noted by Zhu et al. (2024). In the transportation industry, Kessels (2024) applied MDPs in conjunction with deep reinforcement learning to create a dynamic decision-making model for efficient route planning and truck scheduling. Furthermore, Kruit (2024) investigated the usefulness of combining MDPs with reinforcement learning for refuelling decisions at a trucking company. This research is most relevant to our work and will be discussed later.

3.5 Solving Approaches

Existing literature discusses various approaches to solving MDPs. The following sections elaborate on these approaches and their applications. Since this research focuses on truck transportation, only relevant papers are reviewed.

3.5.1 Dynamic Programming

Dynamic programming (DP) is a recursive approach used for solving MDPs. In this method, a value function is created for each state, reflecting the expected reward from

choosing the best action available in that state. There are two main approaches within dynamic programming: value iteration and policy iteration (Rust, 2008). Value iteration involves iteratively updating the value function, from which the optimal policy is eventually derived. In contrast, policy iteration improves the policy during the updates, ensuring that the current best policy is always utilised throughout the process.

Dynamic Programming (DP) is typically effective for smaller problems. However, many real-world issues face what is known as the "curse of dimensionality", which can make them too complex to solve effectively using DP. According to Powell (2011), there are three primary curses of dimensionality: the state space, outcome space, and action space. The challenge encountered at Nijhof-Wassink involves numerous states and actions, which means that traditional DP is not a viable solution approach and will therefore not be explored in more detail.

Approximate Dynamic Programming (ADP) can address the curse of dimensionality, which traditional Dynamic Programming methods encounter, by avoiding the need to enumerate the entire state space of the problem (Powell, 2012). Traditional ADP typically utilises a value function with simple hand-crafted features and linear architectures, which can be effective if the state and action space are large and the mapping from state features to the value function is roughly linear or smoothly additive. However, ADP becomes less efficient when the features have a non-linear relationship with the objective function. In this research, the value of a certain state exhibits non-linear, threshold-type effects, such as doing a small top-up refuel to make a downstream station reachable. The linear $V(s)$ approximation tends to blur or smooth out these action-dependent links, possibly leading to a value function approximation error. Therefore, ADP is not selected as a solution method for the proposed MDP. van Heeswijk et al. (2015) applied ADP to solve the dispatch problem in urban freight distribution. Monte Carlo Sampling was used to approximate the value functions.

3.5.2 Reinforcement Learning

Reinforcement learning learns from sampled interactions with the environment, avoiding the need to explicitly enumerate value functions over the full state–action space, and thereby sidestepping the curse of dimensionality (Sutton and Barto, 2014). A distinction can be made between value-based and policy-based reinforcement learning models. Value-based methods aim to approximate the value function of selecting action a in state s. Policy-based methods directly learn the optimal policy without maintaining separate value functions for each state-action pair. Within both categories, a distinction can be made between off-policy and on-policy methods. On-policy methods update the current policy that is used to collect experience. Off-policy methods learn from a different policy than the one used to make decisions. Furthermore, a distinction is made between deep reinforcement learning methods and non-deep reinforcement learning methods. Instead of tracking the Q-value of every state-action pair, an approximation function is used to estimate this Q-value. In problems with high-dimensional or continuous state spaces, this offers more efficient learning (Farazi et al. 2020). Several Reinforcement learning methods and their applications are discussed in the following sections.

Value-Based Reinforcement Learning

Q-learning and SARSA are both examples of value-based reinforcement learning methods (Sutton and Barto, 2014). There has been limited research on the application of reinforcement learning techniques to the Fuel Routing Vehicle Refuelling Problem (FRVRP). The study conducted by Kruit (2024) is noteworthy as it compares SARSA and Q-learning for optimising refuelling locations and amounts for a transportation company. Given the stochastic nature of fuel prices, a regression model was utilised to predict daily fuel costs. Implementing Q-learning for the refuelling issue resulted in an 11% reduction in costs, potentially saving up to €990,000 per year. However, a limitation of this research is that it focused solely on 21 routes within the Nijhof-Wassink network. Additionally, the constructed model solely allows refuellings up to the fuel tank capacity, which can miss economically necessary trade-offs such as buying just enough to reach a cheaper station.

Deep Q-learning extends Q-learning by using a neural network to approximate the Q-values. Instead of maintaining a Q-table with all Q-values for each separate state-action pair, the neural network approximates the action-value function $Q(s, a)$ with the use of the state values (Tan et al., 2017). Several researchers have applied Deep Q-learning to problems in the transportation domain. For instance, Wang et al. (2025) used Deep Q-learning for a large-scale dynamic logistics UAV routing problem where the aim was to minimise order timeouts and flight costs.

Policy-Based Reinforcement Learning

Policy-based methods have not been applied to address issues related to truck refuelling, cleaning, and resting. Therefore, this Section analyses several other relevant applications. Zhu et al. (2022) compared two value-based methods, the Deep Q Network (DQN) and the Double Deep Q Network (DDQN), with a policy-based method known as Proximal Policy Optimisation (PPO) in the context of traffic light optimisation. It was concluded that PPO performed best out of the three methods. PPO is an online policy-based learning method that generally offers stable learning due to its clipping mechanism. PPO was first introduced in 2017 by Schulman et al. (2017b) and is the successor of the more computationally expensive Trust Region Policy Optimisation, which was first introduced in 2015 by Schulman et al. (2017a).

3.6 Problem Classification and Research Gap

The drawback of most discussed research is the dependency on fixed fuel prices. This can be a problem if the truck's routes span more than one day, as refuelling decisions on day t impact those on day $t + 1$. Fuel prices used to create advice for the whole route may differ on the rest of the days. Furthermore, most of the previous approaches create a refuelling plan before the start of the day; any changes in the planning could invalidate the refuelling advice. Planning in the transportation industry is subject to frequent changes, given the unpredictability of traffic or waiting times at refuelling stations and loading and unloading locations. Sequential decision-making techniques may provide a solution to this problem.

After extensively researching all literature areas concerning this research, the problem addressed in this study can be classified as a Fixed Route Vehicle Refuelling, cleaning and resting Problem (FRV-RCRP), allowing off-route decisions without a refuelling amount requirement. This classification permits off-route decisions without a minimum refuelling amount requirement. While the off-route, no-minimum FRVRP alone has polynomial-time solving algorithms, as discussed in Section 3.2, the integration of cleaning and resting decisions means that this property can no longer be guaranteed. Due to the absence of literature on the combination of resting, cleaning, and refuelling decisions, no claims can be made about the difficulty of the problem. However, Section 4.4 gives an estimation of the problem size by evaluating the state and action space.

3.7 Summary

This chapter mapped the literature regarding the three decision areas in this research. First, several methods for decreasing fuel costs were explained by discussing core routing problems, including the Travelling Salesman Problem (TSP), Vehicle Routing Problem (VRP), and Vehicle Routing Problem With Time Constraints (VRPTW). Subsequently, the existing literature on resting, refuelling, and cleaning was reviewed. The isolated refuelling problem can be positioned in the Fixed-Route Refuelling Problem (FRVRP) category. Due to extensive research on the Fixed Route Vehicle Refuelling Problem (FRVRP), four variants of the problem were discussed, which varied in decision-making for off-route and on-route situations, and included or excluded a minimal refuelling amount. After extensively researching all relevant literature areas concerning this research, the problem at hand can be classified as a Fixed Route Vehicle Refuelling, Cleaning, and Resting Problem (FRV-RCRP), which allows off-route decisions without a refuelling amount requirement. Subsequently, modelling approaches such as Markov Decision Processes (MDPs) were discussed as a modelling technique for sequential decision-making. Classic dynamic programming (DP) and approximate dynamic programming (ADP) have been discussed as possible solution methods. However, with these methods, the curse of dimensionality becomes a problem, especially with DP, when problems involve a large number of different states and actions. The drawback of ADP is that it can struggle to capture non-linear relationships between action space and the value function. Reinforcement learning, specifically Deep Q-learning, scales to large state–action spaces while capturing the non-linear coupling between state, action, and long-term objective, allowing policies to exploit interactions and delayed effects that fixed heuristics or linear models miss.

Chapter 4

Solution Design

4.1 Introduction

This chapter focuses on the design of the research solution, which includes both a modelling and a solution methodology. Section 4.2 outlines the objectives that the model aims to address. Next, various modelling methods are explored, along with their drawbacks in the context of this research. Section 4.3 provides a mathematical representation of the selected modelling approach, and an example is given that demonstrates the model's working. Section 4.4 discusses the complexity of the problem by estimating the problem size. Section 4.5 discusses different solution strategies for the proposed modelling method, highlighting their advantages and disadvantages. Finally, the chosen solution method and its key mechanisms are explained. Section 4.6 gives a summary of this Chapter. The following research questions are answered in this Chapter: *What modelling approach best suits the problem based on the theoretical aspects of integrated decision-making in fleet management?* and *What solution method is most suitable for optimising the chosen model effectively?*

4.2 Modelling Approach

4.2.1 Model Goals

The objective of the model is to make cost-efficient, real-time decisions regarding the cleaning of trailers, refuelling of trucks, and resting of drivers. To select the most suitable modelling and solution approach, specific goals have been established.

- **Unify decisions**: optimising the timing and location of the three separate decisions in a single integrated framework.
- **Support short-term planning:** Given Nijhof-Wassink's three-day planning horizon, the model should optimise decisions within this fixed timeframe.
- **Handle re-routing:** The model must effectively manage possible re-routing changes made by the planning department of Nijhof-Wassink
- **Operational constraints:** the decisions should comply with operational constraints such as legal driving hours, fuel tank capacity and cleaning requirements.

4.2.2 Modelling Assumptions and Simplifications

To simplify the model while remaining realistic, certain modelling assumptions and simplifications have been adopted. These assumptions set the scope of the modelling approach.

- **Driving time and fuel consumption**: It is assumed that the truck drives in a straight line between sequential locations. The haversine distance is used to calculate the driving distance. Furthermore, it is assumed that the fuel consumption and driving speed remain constant.
- **Drive and resting regulations:** Due to the complexity of the driving and resting regulations imposed by the European Union, it is chosen only to implement a maximum driving time of 9 hours per day. This choice is made to still incorporate a time element to observe the effect on the selected model, while keeping the state space and constraints tractable. Fully modelling the multilayered, dependency-heavy regulations would increase the state space of the MDP and introduce long-horizon connections. This would obscure the combinatorial effects of the action that this study aims to research.
- **Uncertainty modelling**: It is chosen to exclude any external uncertainties from the model. This approach isolates the underlying economic trade-offs and enables the evaluation of the model's ability to learn these pure trade-offs without interference from uncertainty modelling. Therefore, it is assumed that the price remains constant during the three-day decision horizon. Additionally, it is assumed that there are no traffic delays and the truck can drive with an average of 80km/h. Once the model successfully learns the core operational trade-offs, uncertainties can be introduced in future research.
- **Time windows:** Customer time windows are not modelled to isolate the core operational trade-offs in the decision process. This allows assessment of whether the proposed model yields sufficiently good solutions to justify further development and deployment

4.2.3 Baseline Frameworks

Since uncertainties are excluded in the current problem, Mixed-Integer Programming (MIP) is a viable modelling technique, provided that the problem size permits its application. In practice, however, solve times escalate rapidly with problem scale, making exact MIP solutions time-consuming and operationally impractical for larger cases. Moreover, Nijhof-Wassink ultimately aims to incorporate uncertainties into the decision-making process. In this future context, MIP will not be suitable due to its inability to capture uncertainties in its most standard form, and thus it is not selected for this purpose. MIP is utilised in this study to establish a benchmark for comparing the chosen model and is further discussed in Section 6.6.1. Stochastic programming, on the other hand, would face issues with scenario explosion due to the large problem instance and the extensive range of exogenous factors such as fuel prices and traffic conditions. As a result, stochastic programming is also excluded as a potential solution approach.

4.2.4 Markov Decision Processes (MDP)

Markov decision process is a sequential decision-making modelling approach which offers a more scalable and adaptive framework than traditional MIP and stochastic programming. MDPs naturally include sequential decision making by updating the state s after executing an action a, which allows for planning over a finite or infinite horizon where actions in the current state impact the possibilities of actions in later states (Puterman, 1990). Additionally, MDPs offer solution methods such as exact dynamic programming, Approximate dynamic programming, tabular Q-learning and Deep Q-networks. The latter three significantly reduce the computational burden that comes with the combinatorial decision space of this problem. Unlike traditional methods, which require enumeration over all possible trajectories, reinforcement learning methods sample from trajectories to learn from experiences, which is a more efficient way of learning. This makes MDP more suitable than stochastic programming and MIP, and therefore this approach is chosen to model the refuelling, cleaning and resting problem.

4.3 Markov Decision Process Formulation

4.3.1 Decision Stages

Nijhof–Wassink plans at most three days ahead and thus each three-day plan for a truck is modelled as a fixed *route*. Let the set of routes be

$$\mathcal{M} = \{1, \dots, M\}. \tag{4.3.1}$$

For each route $m \in \mathcal{M}$, let $\mathcal{E}_m$ be the finite ordered list of planned events:

$$\mathcal{E}_m = (e_{m,1}, e_{m,2}, \dots, e_{m,n_m}). \tag{4.3.2}$$

and let n_m be the length of route m

$$n_m = |\mathcal{E}_m| \in \mathbb{Z}_{>0}, \tag{4.3.3}$$

Decision epochs occur at each planned event from $\mathcal{E}_m$ and also at designated decision locations for refuelling, cleaning, and resting. Essentially, these points represent all the places where the truck stops during a route. Let

$$\mathcal{T} = \{0, 1, \dots, T\} \tag{4.3.4}$$

denote the set of decision stages for an episode on route m, with terminal index T determined by the number of events and decisions on that route. The sequence of states, actions, and rewards (including the last action) is called an *episode* and is

$$\tau_m \;=\; (s_0, a_0, r_0),\; (s_1, a_1, r_1),\; \dots,\; (s_T, a_T, r_T) \tag{4.3.5}$$

where s_t and a_t follow the definitions in Sections 4.3.2 and 4.3.3, and r_t is the instantaneous reward (operating cost) at decision epoch t, further discussed in Section 4.3.5. An episode terminates when one of the following conditions is met:

1. The last planned event e_{m,n_m} has been reached, yielding the terminal state s_T
2. No feasible action exists due to low fuel level or driving hours

4.3.2 State Space

The state space S includes all relevant features necessary for making effective decisions at each stage. One crucial factor in this decision-making process is the location of the truck. Intuitively, this location is best represented by the GPS coordinates (x_t, y_t) of the truck's position.

In addition to the truck's location, it is essential to include information about the truck's condition for effective decision-making. Fuel level f_t, for example, is crucial for timing the refuelling decisions and for making trade-offs between detour costs and fuel prices. Other essential features are c_t, a binary variable that indicates if the truck needs cleaning or not, and h_t, which represents the remaining allowable driving time for the truck. The state can then be formulated as:

$$s_t = \left(x_t,\, y_t,\, f_t,\, c_t,\, h_t\right) \tag{4.3.6}$$

In this case, the location of the truck is represented by standard geographical coordinates. However, representing location through raw GPS coordinates has two important drawbacks.

First, coordinates do not exploit a useful invariance in the problem structure. Two situations can be geometrically identical (same relative positions of the truck and nearby stations) but have different absolute coordinates. In a coordinate-based state representation these appear as distinct states, even though from a decision-making perspective they are equivalent. Any learning algorithm is then forced to explore and learn separately in both copies of essentially the same situation. By instead representing location through relative distances between the truck and the stations, all geometrically identical situations collapse to the same state. This reduces the effective size of the state space and allows the learning algorithm to generalise experience across routes and regions. The invariance of geometrically identical states under a distance-based representation is formally proven in Appendix 8.4.

Second, coordinate values themselves do not have a direct numerical interpretation in terms of decision quality. A small change in latitude or longitude does not immediately translate into a meaningful change in cost or feasibility. Distances, on the other hand, are directly related to travel cost, remaining fuel requirements, and time. For example, a smaller distance to a station can naturally indicate a more attractive option than a station that is much further away. Using distance-based features therefore provides a more informative and interpretable input for any learning or optimisation method.

For these reasons, the location of the truck is represented in terms of its distances to nearby stations rather than its absolute geographic coordinates. This has several other advantages: (i) the resulting model is not tied to specific coordinates and can be applied to other regions outside the scope of this study, (ii) the learning process benefits from a smaller and more structured state space, because the truck does not need to revisit all coordinate-specific copies of intrinsically identical situations, and (iii) if the planning changes while the truck is already en route, the current state can still be

mapped consistently into the model based on relative distances, even if that exact location has not been seen during training.

It is therefore decided to represent the location of a truck with the distance to nearby stations rather than using the truck's geographical coordinates. The distance $d_{k,t}$ to every station k is added to the state expressed as the amount of fuel needed to reach this station. To increase the informativeness of the state, the detour $\delta_{k,t}$, the price p_k and the cleaning time w_k of all stations at the decision stage t are added to the state as they capture trade-offs between travel and direct action costs. Additionally, a binary variable is added to indicate whether the station is a cleaning or refuelling station, due to the existing price difference between gas stations and cleaning stations. These variables together form the *station features* component of the state representation.

In addition to these, there are four *global features* that are not tied to any specific station but provide relevant information about the truck. These global features include the fuel level f_t, the allowable remaining driving hours h_t and a binary indicator c_t which indicates if cleaning is required at time step t. Lastly, a feature l_t is added to indicate the length of the route left ahead of the truck. The full state representation at decision epoch t is now formulated as:

$$s_t = \Big(\overbrace{\big\{(d_{k,t}, p_k, \delta_{k,t}, w_k, z_k) \mid k \in K_{t,m}\big\}}^{\text{Station features}}, \overbrace{(f_t, c_t, h_t, l_t)}^{\text{Global features}} \Big) \tag{4.3.7}$$

where:

- $d_{k,t} \in \mathbb{R}_{\geq 0}$: Fuel needed (L) to reach station k at time t.
- $p_k \in \mathbb{R}_{\geq 0}$: Service price (€) at station k.
- $\delta_{k,t} \in \mathbb{R}_{\geq 0}$: Detour distance (km) to station k at time t.
- $w_k \in \mathbb{R}_{\geq 0}$: Cleaning time (h) at station k.
- $z_k \in \{0, 1\}$: 1 if station k is a cleaning station, 0 otherwise.
- $f_t \in \mathbb{R}_{>0}$: Fuel level (L) at time t, with $f_t < F_{\text{cap}}$.
- $c_t \in \{0, 1\}$: 1 if cleaning is required at time t, 0 otherwise.
- $h_t \in \mathbb{R}_{\geq 0}$: Remaining allowable driving time (h) at time t.
- $l_t \in \mathbb{R}_{\geq 0}$: Length of the route remaining (m).

Let $K_{t,m}$ denote the set of included stations in stage t in route m. A description of this set is given later in this section

The following sets represent all stations known in the database of Nijhof-Wassink:

$$\begin{aligned} J^1 &= \{j_1, j_2, \ldots, N^1\} \quad &&\text{: set of all refuelling stations,} \\ J^2 &= \{j_1, j_2, \ldots, N^2\} \quad &&\text{: set of all cleaning stations,} \\ J^3 &= \{j_1, j_2, \ldots, N^3\} \quad &&\text{: set of all resting stations} \end{aligned} \tag{4.3.8}$$

$$J = J^1 \cup J^2 \cup J^3 \tag{4.3.9}$$

The set from which the station-specific part of the state is drawn is constrained per route to narrow the state space by including only relevant stations. For instance, only refuelling and resting stations located ahead of the truck are considered, as it is generally inefficient to drive back and forth along the route. Due to cleaning restrictions for certain products, the truck may need to drive backwards from its current loading or unloading location. Therefore, the set of cleaning locations in the state includes stations located behind the truck. Furthermore, only stations within a limited distance from the route are included, as it would be impractical to travel across Europe to refuel and then return to the original route.

To define the stations included in the state of decision stage t in route m, the distance of the truck and stations along the route are defined as x_t and x_j respectively. Here, j is a station which is part of the station set J. The orthogonal distance from a station to the route line is defined as o_j and is used to check if a station is located in the buffer around the route. The mathematical representation of the station set included in the state is described as:

$$\begin{aligned}
K^1_{t,m} &= \{\, j \in J^1 \mid x_j > x_t \ \wedge\ o_j \leq R \} && \text{(refuelling stations ahead and inside buffer),} \\
K^2_{t,m} &= \{\, j \in J^2 \mid o_j \leq R \} && \text{(cleaning stations inside buffer),} \\
K^3_{t,m} &= \{\, j \in J^3 \mid x_j > x_t \ \wedge\ o_j \leq R \} && \text{(resting stations ahead and inside buffer).}
\end{aligned}$$

The complete set of station entries in the state is thus:

$$K_{t,m} = K^1_{t,m} \cup K^2_{t,m} \cup K^3_{t,m} \quad \text{(all feasible entities in the current state).}$$

The next planned event in the event list of a route $\mathcal{E}_m$ at a specific moment in time is defined as $e^{next}_{t,m}$ and the position of this event on the route is defined as $x^{next}_{t,m}$

4.3.3 Action Space

In the state, all refuelling and resting stations ahead of the truck ($K^1_{t,m}$ and $K^3_{t,m}$) and all viable cleaning stations ($K^2_{t,m}$) are included. However, the action space restricts refuelling and resting locations to stations that are ahead of the truck and lie before the next planned event $x^{\text{next}}_{t,m}$. This approach is intended to reduce the state-action space and improve training efficiency. It is inefficient to drive to a station past the next planned event only to return to that event. The action sets are defined as follows:

$$\mathcal{A}^1_{t,m} = \{\, j \in K^1_{t,m} \mid x_j < x^{\text{next}}_{t,m} \,\}, \qquad \mathcal{A}^2_{t,m} = K^2_{t,m}, \qquad \mathcal{A}^3_{t,m} = \{\, j \in K^3_{t,m} \mid x_j < x^{\text{next}}_{t,m} \,\}.$$

At each decision stage t, the driver has to choose an action in the form:

$$a_t = (a_t^1, a_t^2)$$

where:

- $a_t^1 \in \mathcal{A}_{t,m}^1 \cup \mathcal{A}_{t,m}^2 \cup \mathcal{A}_{t,m}^3 \cup \{e_{t,m}^{\text{next}}\}$
- $a_t^2 \in \{0, 10, 20, \ldots, F_{\max}\}$: Amount of fuel to add.

The actions are restricted to the following constraints:

- The selected location a_t^1 must be reachable with the current fuel level f_t:

  $$\frac{D(a_t^1)}{\rho} \leq f_t,$$

  With ρ being the fuel consumption per kilometre and $D(a_t^1)$ being the haversine distance to the location of action a_t^1.

- The refuelling amount a_t^2 must be non-negative if a refuelling station is selected

  $$a_t^1 \in K_{t,m}^1 \Rightarrow a_t^2 \geq 0$$

- If action is not a refuelling action, no fuel can be added

  $$a_t^1 \in K_{t,m}^2 \cup K_{t,m}^3 \cup \{e_{t,m}^{\text{next}}\} \Rightarrow a_t^2 = 0.$$

- Cleaning is only allowed if the truck requires cleaning:

  $$a_t^1 \in K_{t,m}^2 \Rightarrow c_t = 1$$

- The refuelling amount a_t^2 must not exceed the remaining tank capacity after arriving at the selected refuelling location $F_{\max}$.

  $$a_t^2 \leq F_{\max} - f_t - \frac{D(a_t^1)}{\rho}$$

- The driving time to the action location should be lower than the remaining allowed driving time.

  $$T(a_1^t) \leq h_t$$

  where $T(a_{1t})$ is the driving time to action a_1^t and is defined as:

  $$T(a_1^t) = \frac{D(a_t^1)}{v_{\text{avg}}}$$

4.3.4 Transition Function

The transition function is a part of the MDP that specifies how the environment evolves in response to an executed action. The transition function can be either deterministic or probabilistic, depending on the characteristics of the problem. In this research, uncertainties related to traffic and fuel prices have been excluded from the scope, making the transition function deterministic. The transition function consists of several parts, which are discussed in the following sections.

Station Sets Update

The location-specific part of the state must be updated by identifying which stations lie in the buffer and are ahead of the current location of the truck. This is done by calculating the feasible state sets $K_{t,m}$ at every new decision epoch t. Then, the detours and distances along the route are computed for each station.

Fuel Level Update

The fuel level is updated by subtracting the fuel consumed while driving to the selected location and adding the amount refuelled a_t^2. Fuel consumption is calculated by multiplying the distance travelled $D(a_t^1)$ by the truck's fuel consumption rate per kilometre ρ. This number is based on information from colleagues within Nijhof-Wassink.

$$f_{t+1} = f_t - \frac{D(a_t^1)}{\rho} + a_t^2$$

Cleaning Status Update

The cleaning status of the truck is set to 0 if the truck is just cleaned, it is set to 1 if the last location was an unloading location and is kept the same otherwise.

$$c_{t+1} = \begin{cases} 0 & \text{if } a_t^1 \in K_{t,m}^2 \\ 1 & \text{if } a_t^1 = e_{t,m}^{next} \\ c_t & \text{otherwise} \end{cases}$$

Resting requirement update

The new remaining allowed driving time is calculated by subtracting the driving time to the chosen location $T(a_t^1)$ from the current remaining driving time H_t. The remaining driving time is reset to the maximum driving time h_{max} according to EU regulations if the previous action was resting:

$$h_{t+1} = \begin{cases} H_{max} & \text{if } a_t^1 \in K_{t,m}^3 \\ h_t - T(a_t^1) & \text{otherwise} \end{cases}$$

4.3.5 Reward Function

To assess the quality of an action, an objective function is created. The objective function, also known as the reward function, represents the immediate costs of an action

without considering the downstream effects that the action may induce. The reward function is defined as:

$$R(s_t, a_t) = R_{\text{refuel}} + R_{\text{clean}} + R_{\text{rest}} + R_{\text{penalty}} + R_{terminal}$$

where the refuelling costs are given with the following formula.

$$R_{\text{refuel}} = \begin{cases} -\left(a_t^2 \cdot p(a_t^1) + C_{\text{detour}}(a_t^1) + C_{\text{stop}}(a_t^1)\right) & \text{if } a_t^1 \in K_{t,m}^1 \\ 0 & \text{otherwise} \end{cases}$$

Here a_t^2 is the chosen refuelling amount, $p(a_t^1)$ is the refuelling price of the chosen action and C_{detour} is the cost incurred by driving the full length of the detour corresponding to action a_s^1. It is chosen not to use the direct driving costs to the station in the reward function because the detour better represents the quality of an action. The costs incurred due to waiting time at any station are referred to as C_{stop}.

The cleaning costs are determined by summing the costs incurred from the cleaning action, the detour costs related to that action, and the waiting costs associated with the product being cleaned.

$$R_{\text{clean}} = \begin{cases} -\left(C_{\text{clean}}(a_t^1) + C_{\text{detour}}(a_t^1)\right) - W_{\text{clean}} \times C_{\text{hour}} & \text{if } a_t^1 \in K_{t,m}^2 \\ 0 & \text{otherwise} \end{cases}$$

Since there are no explicit costs associated with the resting action, only the driving and stopping costs are included in this part of the reward function.

$$R_{\text{rest}} = \begin{cases} -C_{\text{drive}}(a_t^1) - C_{\text{stop}} & \text{if } a_t^1 \in K^3(t, m) \\ 0 & \text{otherwise} \end{cases}$$

In the case where the truck ends up in a location where there are no available actions, a dead-end penalty is incurred. The penalty is represented by the parameter λ.

$$R_{\text{penalty}} = \begin{cases} -\lambda_{\text{penalty}} & \text{if no feasible action exists at } s_t \\ 0 & \text{otherwise} \end{cases}$$

Because the optimisation horizon of the MDP is three days long, the current reward structure without a terminal reward would bias the agent towards small top-ups, which are just enough to finish the episode. However, in reality, the trucks continue to operate beyond this horizon and take any remaining fuel with them. Small top-ups are thus generally not beneficial. To align the finite-horizon MDP with the infinite-horizon reality, a terminal reward is implemented. The terminal value is set to the fuel left after finishing the route multiplied by the average fuel price. The terminal reward is defined as:

$$R_{\text{terminal}} = p_{avg} f_T.$$

where f_T is the terminal fuel level and p_{avg} is the average fuel level for Belgium, the Netherlands and Germany in the past year.

4.4 Problem size and complexity

4.4.1 Continuous state-action space

The MDP uses continuous features such as distances, detours, fuel levels and fuel prices, so the theoretical state space is uncountable. Hence, without discretisation, the problem has an infinite (uncountable) state–action space.

4.4.2 Discrete action space

To make the combinatorial scale explicit, the per-stage decision space under discretisation consists of: (1) choosing a fuel station and a discrete refuel amount, (2) choosing a cleaning station, (3) choosing a rest station (4) choose to proceed to the next scheduled loading or unloading event. If $|K_t^1|$, $|K_t^2|$, and $|K_t^3|$ denote the numbers of fuel, cleaning, and rest options at stage t, and R^+ denotes the number of discrete refuel amounts, then the number of feasible actions is

$$B_t \;=\; |K_t^1|\, R^+ \;+\; |K_t^2| \;+\; |K_t^3| \;+\; 1.$$

For a horizon of T decision stages, the decision-space size is

$$|\text{action sequences}| \;=\; \prod_{t=0}^{T-1} B_t$$

For a typical route, it is observed that:

$$T = 10, \qquad |K_t^1| = 50\,, \quad |K_t^3| = 50\,, \quad |K_t^2| = 10\,.$$

Refuel amounts are discretised in 50 L steps up to 550 L, so the number of refuelling addition amounts is:

$$R^+ = \frac{550}{50} = 11.$$

Per stage, the total number of decision sequences is:

$$B_t \;=\; |K_t^1| \cdot R^+ \;+\; |K_t^2| \;+\; |K_t^3| \;+\; 1 \;=\; 50 \cdot 11 \;+\; 10 \;+\; 50 \;+\; 1 \;=\; 611$$

The number of decision sequences for a typical route is then given as:

$$\prod_{t=0}^{T-1} B_t \;\approx\; B^T \;=\; 611^{10} \;\approx\; 7.2 \times 10^{27}$$

4.5 Solution Approach

4.5.1 Baseline Methods

Exact Dynamic Progamming

Dynamic programming (DP) is a powerful technique which uses backwards induction to solve finite MDPs and avoids the need for complete enumeration of the entire solution configuration. To solve the Markov Decision Process (MDP), this method begins

at the terminal state and recursively computes the expected cumulative rewards for the preceding states. At each step, it selects the action that maximises the expected cumulative reward until reaching the initial state. This approach significantly reduces computational time by creating overlapping sub-problems and storing their outcomes. However, when the state-action space becomes considerably large, DP becomes computationally impractical. Even if the planning for a single truck can be computed using DP, the slightest change in planning due to unforeseen circumstances would make the solution infeasible, requiring a new computation, which takes a considerable amount of time. Due to these aspects, dynamic programming is not a feasible method for addressing this issue.

Approximate Dynamic Programming

ADP helps with the curse of dimensionality faced in this research. However, in practice, ADP relies on linear hand-crafted value function approximators. The landscape in this problem has strong non-linear, threshold effects, such as small top-up refuels that unlock downstream stations. A linear value function approximation tends to smooth these action-dependent discontinuities, producing biased value estimates unless a rich state space is provided. Given this mismatch, ADP is not implemented as a solution method in this research.

Tabular Q-learning

Tabular Q-learning, also known as Q-learning, is a model-free reinforcement learning method that solves MDPs by accumulating experiences gained in an environment. The method uses a Q-table that saves the value function $Q(s, a)$ for each state-action pair. The value function can be viewed as an approximation of the cumulative reward obtained by choosing action a in state s and then following the optimal policy thereafter.

The learning process in this method proceeds iteratively: an action is taken after which the immediate reward is observed, with which the Q-value is updated using the Bellman equation. Executing this procedure many times with sufficient exploration of states and actions and optimised hyper-parameters lets the Q-values converge to their true optimal values.

Q-learning can handle stochastic problems and avoids the need for an explicit model, and can be an effective solving technique. However, due to the vast state and action space of this problem, Q-learning would need to spend too much time exploring all possible state-action pairs, and thus, convergence would be slow. This is caused by the fact that the learning agent must visit every separate state and execute every possible action multiple times to obtain a reasonably accurate estimation of the Q-values. It is therefore decided not to use Q-learning as a solving method.

Heuristics

Heuristic solution methods utilise specific characteristics of the problem at hand to generate a solution in a relatively small amount of time. Greedy heuristics are a type

that offer a fast and straightforward decision-making approach by selecting actions based on their immediate reward. However, due to their short-sighted nature, they are not very efficient in optimising cumulative rewards over a specific time horizon. An example of a greedy heuristic is to fully refuel at the cheapest station within the truck's range when the fuel level falls below 50 litres. The chosen station might have a high price compared to stations ahead of the truck, which would make a full refuel very costly. An alternative would be to refuel only a small amount at the expensive station and then fully refuel at a cheaper station later on the route. Due to the greedy nature of most heuristics, the optimal solution in this scenario is overlooked.

Even so, heuristics are useful when speed, simplicity and interpretability matter, and they provide a clear reference point for more advanced methods. Therefore, a simple heuristic is utilised as a baseline for benchmarking the chosen solution method introduced in Section 4.5.2.

4.5.2 Deep Q-learning

Deep Q-learning extends Q-learning by using a neural network to approximate the Q-values. Instead of maintaining a Q-table with all Q-values for each separate state-action pair, the neural network approximates the action-value function $Q(s, a)$ with the use of the state values. This enables generalising across states, which means that it is not necessary to visit each individual state-action pair. This improves training efficiency and convergence. Additionally, deep Q-learning supports offline training from historical data, which allows the agent to learn from previously logged experiences without interacting with the real world, which is less costly. Next to the generalisability across states, a change in the planning schedule does not cause infeasibility of the solution. Deep Q-learning bases its decisions on learned patterns in state features, which are "trapped" inside the neural network, enabling it to create new decision advice based on the new state after a change in the planning. Due to these reasons, Deep Q-learning is selected as the solving approach for the MDP.

4.5.3 Q-network architecture

The Q-network can be defined as a function $Q(s, a; \theta)$ with parameters θ, which takes as input the state vector s and outputs a $|\mathcal{A}|$-dimensional vector of estimated action-values $\{Q(s, a_i; \theta)\}_{i=1}^{|\mathcal{A}|}$. Training proceeds by minimising the discrepancy between these estimates and targets constructed from actual observed returns over many training episodes. The standard Q-network consists of an input layer, multiple hidden layers and an output layer. Every layer consists of multiple neurons. All network layers are connected via edges representing the weights of the preceding neuron. In DQN, every input neuron is connected to a feature of the state in the MDP. The neurons in the output layer represent the Q-values of the possible actions in the MDP. The architecture of the classical Q-network can be found in Figure 8.3

One of the drawbacks of a standard neural network is that the dimensions of the neural network must remain consistent during training so that the weights in the neural network can converge to the values that lead to the best approximation of the Q-values.

This introduces a problem in combination with the selected state representation discussed in Section 4.3.2 as the number of stations in the state decreases when the truck progresses along the route. A possible solution is to add padding where empty neurons appear during training. All input neurons that are not connected to a feature of the state in time step t are fed with a zero.

The Q-network is able to generalise across states, which means that the network can calculate a Q-value for a state-action pair that it has not seen before. This implies that a one-to-one relation between the individual features of the specific state instance (e.g., a particular station) and the specific input neurons is not required. Instead, the input to the neural network is viewed as a whole, which enables the network to learn shared representations and identify meaningful patterns. In contrast, the neurons in the output layer must have a recurring one-to-one connection with the same action over different learning episodes. This presents a problem with the design of the MDP because the available stations vary across different decision stages and routes. An example is that gas station 1 is available in the first episode and is connected to the first neuron of the output layer. In the second episode, Station 1 is not included in the buffer around the route. Thus, this specific action can not be connected to the same output neuron as the previous episode. Now, a different station must be connected to this output neuron. Such dynamic remapping of stations to output neurons prevents the network to learn the true Q-values of actions over time.

One possible solution would be to create a Q-network with an output layer the size of all stations in the Nijhof-Wassink database times the number of actions per station. This way, all possible actions would have a distinct output neuron in the output layer. However, given that there are approximately 5,000 refuelling stations and an additional 350 cleaning stations, and assuming a discretisation of the refuelling range into 49 levels, the total number of distinct action combinations would amount to (5,000 + 350) x 49 = 262,150 output neurons. A neural network of this size would have to update millions of weights in every update, resulting in substantial computational time and slow learning.

To support decision-making with a variable set of candidate actions over different episodes, a parametric Q-network is used. In contrast to a standard Q-network, the parametric Q-network has one output neuron, representing the Q-value of a specific state-action pair. This allows the network structure to remain fixed regardless of the number of available actions. Hausknecht and Stone (2015) formalised a Deep Q-learning approach with parameterised actions, avoiding a fixed, giant output head. A visual comparison between the traditional Q-network and the parametric Q-network can be found in 8.3 and 8.4.

The network's input remains unchanged from the traditional DQN, but the features of the station, along with the refuelling amount, are also fed into the input side of the neural network. This is done to establish a connection between the global state of the truck and the action associated with the output neuron. The input to the neural network is as follows:

- **State features**: identical to those used in the original Q-network, describing the current situation of the environment.

  - *Global features*: This part consists of global truck features, which are: fuel level, cleaning requirement, resting requirement and the remaining route length.
  - *Station features*: These are station-specific features, such as station prices, distances, detours, and flags of all stations.

- **Action features**: These are the features of the station which is currently being evaluated by the network (the station that corresponds to the action that is connected to the output neuron). This part consists of:

  - *Station-specific attributes*: The distance to the station, the fuel price, and the detour required to reach it. These values remain constant for all refuel amounts at that station. This is necessary to indicate which action is currently being evaluated by the network.
  - *Refuel-amount attribute*: The refuelling quantity F_t currently evaluated by the network at time step t, expressed in litres. For each feasible station, all allowable refuel amounts are enumerated. Each combination of the station features, station-specific attributes and refuel amount F constitutes a distinct action vector.

4.5.4 State Vector Composition

The state coming from the MDP can only be presented to the neural network in a 1-dimensional form. To enable the neural network to discover meaningful patterns, the state vector has to be designed cleverly. As previously discussed, connecting the features of every station in Nijhof-Wassink's database to a distinct input neuron is not feasible because the neural network's size is capped at the maximum number of stations in the route buffer. Structuring the input vector between decision steps and episodes can lead to increased training stability and improved final performance. Three main strategies have been applied in creating the state vector. This is especially important for the station features part in Figure 8.4 because the size of this set changes over episodes as new stations enter the route buffer and other stations leave it.

Station features

First, the feature values are grouped by feature type, rather than grouping the features by station. Preliminary experiments have indicated that training on a vector grouped by stations yields poorer performance. This may be because the neural network is more effective at identifying patterns within larger sets of features (e.g. within the set of fuel price features of all stations), rather than focusing on specific stations and their respective features.

The second strategy involves creating predetermined blocks for each feature set. This approach ensures that the input neurons of the network consistently process the same feature types throughout the episodes. A different strategy would be to fill the neurons from top to bottom with all available features, creating a solid block of features with padding inserted below. However, initial experimentation revealed that creating fixed blocks per feature yielded better performance. Therefore, this strategy is selected.

Finally, the features per station keep connected to the same neuron within the current episode. This ensures a stable correspondence between the features of a station and the neurons, which may help the network to recognise short-term patterns and causal effects within episodes. Features of the stations that end up behind the truck get a zero value in the state vector. Equation 4.5.1 gives the station feature part $\mathbf{x_{station,t}}$ of the input vector. Every $\phi^{(k)}$ in the vector $\mathbf{x}_t^{\text{station}}$ represents a vector of the features of station k. N_{Max} is the maximum number of stations that can be represented in the neural network.

$$\begin{aligned} \mathbf{x}_t^{\text{station}} &= \left[\boldsymbol{\phi}_t^{(1)} \; \boldsymbol{\phi}_t^{(2)} \; \cdots \; \boldsymbol{\phi}_t^{(N_{\max})} \right] \in \mathbb{R}^{5N_{\max}}, \\ \boldsymbol{\phi}_t^{(k)} &= \left[\delta_{k,t},\; d_{k,t},\; p_k,\; w_k,\; z_k \right]. \end{aligned} \tag{4.5.1}$$

Global features

The global features maintain a consistent size within each episode and across multiple episodes, and are thus assigned to a fixed block of 4 neurons. The global feature part $\mathbf{x_t^{global}}$ at time t is given by Equation 4.5.2

$$\mathbf{x}_{\text{global},t} = \left[f_t, c_t, h_t, l_t\right] \tag{4.5.2}$$

Action features

The action features consist of the station-specific attributes of the action that is being evaluated and the refuelling amount of that action. The station specific attributes have been discussed before. Additionally, to indicate the type of action which is present at the output neuron, a one-hot encoding $o_{k,t}$ of the action type is added to indicate the current action type. The one-hot encoding is of length four as there are four distinct actions: refuelling, cleaning, resting and next waypoint. The action feature vector is given in Equation 4.5.3.

$$\mathbf{x}_t^{\text{action}} = \left[\delta_{k,t}, d_{k,t}, p_k, w_k, F_{k,t}, o_{k,t}\right] \tag{4.5.3}$$

The full input vector that is inserted into the neural network can then be formulated as:

$$\mathbf{x}_t^{\text{final}} = \left[\mathbf{x}_{\text{global},t},\ \mathbf{x}_{\text{station},t},\ \mathbf{x}_{\text{action},t}\right] \tag{4.5.4}$$

4.5.5 Training Process

Now that the network's architecture is explained, the training procedure can be explored. The training process consists of several parts, which are described separately in the following sections.

Environment interaction

A significant advantage of Deep Q-learning is that the network can learn from previous experiences multiple times. To achieve this, the agent must interact with the environment. At each decision step t, the agent selects an action a_t based on the current state s_t according to the exploration strategy which is used. The epsilon-greedy policy is employed in this research, which balances exploration and exploitation. The epsilon greedy policy selects a random action with probability ϵ and selects the action with the highest Q-value according to the current Q-network with probability $1 - \epsilon$. The random action selection process is divided into two stages. First, the action category is chosen uniformly from the set {*Refuel*, *Clean*, *Rest*, *Next waypoint*}. If the selected category is *Refuel*, the specific refuelling amount a_t^2 is randomly chosen from the set of feasible amounts:

$$a_t^2 \in \{\, 0,\ \Delta F,\ 2\Delta F,\ \ldots,\ \max\{0,\ F_{\max} - f_t - \tfrac{D(a_t^1)}{\rho}\}\,\}.$$

where ΔF denotes the discretisation step size in litres for the fuel addition and $F_{\max}$ is the capacity of the fuel tank.

Another option would be to choose from the complete list of feasible actions uniformly. However, per decision step, there are many stations and refuelling amounts per station, but only a single option to proceed to the next waypoint. This imbalance would lead to the over-representation of the *Refuelling* action during exploration. The chosen

exploration strategy uniformly explores the four different decision types. The chosen action is then executed, and the state is transitioned to the next state s_{t+1}. Additionally, the reward r_t is incurred, and the terminal flag is checked to determine if the episode has ended.

Experience Replay

Traditional Deep Q-learning uses temporal difference (TD) learning and stores

$$(s_t,\, a_t,\, r_t,\, s_{t+1},\, d_t)$$

into the experience buffer to enable experience replay. However, with the current MDP setup, this has some drawbacks. During experimentation, it became clear that in the initial learning phase, where the epsilon is still large, the agent tends to learn that making several smaller refuels is more beneficial than making one large refuel. The undersampling of single large refuels at the start of the training process can explain this behaviour. These single large refuels do occur later in the process when epsilon is lower. However, because the Q-values are updated with bootstrapped targets drawn from the skewed replay buffer, the Q-values of large refuel actions are never updated with a downward trajectory which has no additional refuelling. As a result, the agent is biased towards taking many smaller refuels and never learns the true optimal policy. In a situation where prices, detours, and total refuel amounts are the same, refuelling multiple small amounts is always less cost-efficient than a single large refuel because stopping costs are incurred for every refuelling stop.

To reduce this bootstrapping bias, Lin et al. (2018) combines the traditional short-horizon bootstrapped target with an episode-based Monte-Carlo return. In this work, the pure Monte Carlo component is adopted, and the network is trained with full-episode returns. For every episode, $(s_0, a_0, r_0, \ldots, s_T, a_T, r_T)$ is computed and then (s_t, a_t, G_t) is stored in the replay buffer $\mathcal{D}$. The cumulative reward G_t can be viewed as the target to which the neural network updates its parameters and is calculated as follows:

$$G_t \;=\; \sum_{k=t}^{T-1} \gamma^{k-t} \;\times R(s_k, a_k)$$

Here, γ is the discount factor and is set to 1 because the problem at hand has a short-term fixed time horizon. During each update, a batch $\mathcal{B}$ of state-action pairs is uniformly sampled from $\mathcal{D}$. This Monte Carlo–style replay leverages full return information and ensures that updates faithfully reflect real decision sequences.

Monte-Carlo targets are known to have a high variance in reinforcement learning, which can slow or destabilise training (Daley et al., 2024; Sutton and Barto, 2014). Prior work shows that the variance of the n-step return increases with the look-ahead n (Daley et al., 2024). In our setting, episodes are short, so the effective look-ahead is limited, and so the variance remains manageable. It is therefore decided to use full-episode Monte Carlo returns to avoid bootstrapping bias, while relying on the short horizon to keep variance under control.

Stratified replay buffer

Even with Monte Carlo sampling, the ϵ-greedy exploration strategy causes smaller refuel transitions to be overrepresented in the replay buffer. This is caused by the fact that a small 10L refuel is always possible, but a large refuel of 400L is sporadically possible due to the tank capacity. This feasibility imbalance causes the mini-batches to contain many more small refuel transitions, which results in their Q-values being updated more frequently and therefore converging faster. The large refuel actions remain undertrained which can bias the agent towards many small refuels.

To evenly distribute the refuel amounts of the transitions in the mini batch, a stratified replay buffer is created that bins transitions by refuel amount. The total replay buffer is divided into equal-width ranges of 50L from 0 to the fuel tank capacity. With a fuel capacity of 550L, this yields 12 bins. During training, each refuelling transition is assigned to the corresponding bin. Non-refuel transitions are stored in the 0 litre bin. During training, Mini-batches are drawn uniformly across the different bins. The frequency bias, which is present under the ϵ-greedy exploration strategy, is countered by this approach.

Loss function and updating

Updating the neural network involves evaluating the loss function and backpropagating the loss through the neural network. The loss function indicates the magnitude of inaccuracy of the current version of the neural network. This inaccuracy is calculated as the squared difference between the Q-value of the state-action pair calculated by the Q-network and the observed return G_t of that same state-action pair. Next, the average of the squared differences from all batches is computed. The loss function is defined as:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \big(G_i - Q(s_i, a_i; \theta)\big)^2.$$

The loss is then backpropagated over all the parameters θ by applying the chain rule. The backpropagation process is not further explained, as it falls outside the scope of this research. The full training process is depicted in Algorithm 1.

Training Algorithm

Algorithm 1 provides the exact training procedure, including the different aspects described in the previous section. The DQN training algorithm differs from the "traditional" DQN algorithm in three respects. First, a parametric DQN is used instead of the traditional DQN, enabling generalisation in the large action space. Secondly, Monte Carlo sampling is used instead of conventional one-step bootstrapping, eliminating the bootstrapping bias. Lastly, a stratified replay buffer is used, partitioning experiences by refuelling amount.

Algorithm 1 Deep Q-Learning with Monte Carlo Replay and Candidate Actions

Require: Train route set M; MDP/env generator; parametric Q-network $Q_\theta(s, a)$ with stratified replay buffer $\mathcal{D}$ (binned over refuel amount R); batch size B; discount factor γ; penalty p_{novalid}; ε-schedule $(\varepsilon_0, \varepsilon_{\min}, \text{decay})$.

for episode $= 1$ **to** M **do**
  Sample route from $\mathcal{M}$.
  Construct MDP with route, buffers
  draw start fuel $f_0 \in \{50, 100, \ldots, 550\}$.
  Reset environment
  done = True
  while done **do**
    $C_t \leftarrow$ valid action candidates.
    if $C_t = \emptyset \ \wedge \ e_{t,m}^{\text{next}} = \varnothing$ **then**
      $r_t \leftarrow R_{\text{penalty}}$
      done = False
      break
    end if
    Build action feature vector A_t
    Build state feature vector S_t
    Compute Q-values $q_t \leftarrow Q_\theta(S_t, A_t)$.
    Choose action according to ϵ-greedy
    Execute chosen action a_t
    Execute transition functions
    Calculate reward $R(s_t, a_t)$
  end while
  Monte Carlo returns: Calculate monte carlo returns G_t
  Push to stratified buffer: for each t push (s_t, a_t, G_t, bin) into $\mathcal{D}$
  if $|\mathcal{D}| \geq B$ **then**
    Draw an index set $\mathcal{I}$ of size B from $\mathcal{D}$ (stratified over R-bins).
    Form mini-batch $\mathcal{B} = \{(s_i, a_i, G_i, b_i)\}_{i \in \mathcal{I}}$.
    Compute loss $\mathcal{L}(\theta) = \frac{1}{B} \sum_{i \in \mathcal{I}} \big(G_i - Q_\theta(s_i, a_i)\big)^2$.
    Backpropagate: compute $\nabla_\theta \mathcal{L}$ with chain rule.
    Optimizer step: update $\theta \leftarrow \theta - \eta \, \nabla_\theta \mathcal{L}$.
  end if
  Anneal exploration: $\varepsilon \leftarrow \max(\varepsilon_{\min}, \ \varepsilon \cdot \text{decay})$.
end for
Output: neural network with optimised weights Q_{θ^*}

4.6 Summary

This chapter develops an integrated decision support approach for the refuelling, cleaning and resting decisions over a fixed three-day planning horizon. A Markov Decision Process is selected as the modelling approach. The state represents station-level features, such as distance, detour, price, and station type, as well as global truck features, including fuel level, cleaning requirements, remaining driving hours, and the length of the remaining route. The actions are modelled as a combination of lo-

cation and refuelling amount, if applicable. The selected solution method is a Deep Q-learning model. Classical Mixed-Integer Programming, Approximate Dynamic Programming, tabular Q-learning, and heuristic approaches are unable to capture long-term decision dynamics or deal with non-convex problems. Deep Q-learning utilises a Q-network to approximate the quality of different actions without enumerating the state space. To enable stable training with a large variable-action set, a parametric Q-network is selected. To mitigate sampling bias, a stratified replay buffer is combined with Monte Carlo sampling. Model training uses ϵ-greedy exploration with category balancing, full Monte Carlo replay sampling, and a stratified replay buffer that balances refuelling amount sampling. The final design aims to provide a robust understanding of economic trade-offs, timing of actions, and the selection of stations within the operational constraints of Nijhof-Wassink.

Chapter 5

Data Collection and Preparation

This Chapter explains the required input data for the MDP model and the Deep Q-learning solving approach. First, Section 5.1 covers the route training data. Next, Sections 5.2 and 5.3 discuss the retrieval of the gas- and cleaning station data set respectively. Section 5.4 outlines the steps involved in data cleaning. Furthermore, Section 5.5 describes the methods used for distance calculation required for the operation of the MDP. Section 5.6 covers the variable and feature normalisation methods. Finally, 5.7 provides a summary of this Chapter. This Chapter aims to answer the following research questions: *What specific data must be incorporated into the model to effectively address the integrated decision-making problem?* and *What preprocessing steps are required to prepare the data?*

5.1 Route Data

To effectively model the problem, route data is essential. Nijhof-Wassink possesses planning data that includes a comprehensive list of all planned loading and unloading activities. This data covers the period 2024-02-18 until 2025-02-18. This planning data can be treated as route data since connecting all the individual events creates a continuous route for each truck. Table 8.2 illustrates all the columns in the data, detailing how each is utilised and its specific purpose. The columns used in this research are ActionType, Truck, Trailer, productCode, and the x and y coordinates.

Nijhof-Wassink plans actions up to three days ahead, which naturally serves as a suitable splitting point. Loading and unloading locations beyond this three-day horizon are not yet known, making it unclear how today's decisions will affect possible actions in the future. The truck route data is divided into three-day segments, which serve as training episodes for the DQN. The dataset now includes multiple sets of loading and unloading locations over three days.

5.2 Gas Station Data

The refuelling locations and prices for Nijhof-Wassink are sourced from two distinct datasets. The first dataset includes the GPS coordinates of gas stations with which Nijhof-Wassink has pricing agreements. While more stations are available, the prices per litre at those additional locations are excessively high, making it unprofitable to

refuel at stations not included in this dataset.

The second dataset contains historical fuel prices per litre for each day. The stations listed in the first dataset can be matched with the fuel prices from the second dataset based on the country of origin, which is present in both datasets.

5.3 Cleaning Station Data

The cleaning locations and their corresponding prices are obtained from two datasets. The first dataset includes all planned cleanings scheduled between May 14, 2024, and February 18, 2025. This dataset contains information about the products that were cleaned, the address codes of the cleanings, and the associated GPS coordinates. However, it does not include the prices for the cleanings.

The prices are retrieved from a different dataset covering the period from January 5, 2021, to May 3, 2025. This dataset lists all cleaning actions, along with their corresponding product codes, address codes, and respective prices. After extracting this information, an average cost is calculated for all unique pairs of product codes and address codes. Subsequently, the costs are linked to the GPS coordinates by connecting the address codes from both datasets.

The cleaning times per product are in a separate dataset and are based on the commodity codes assigned to each product. A different dataset includes the address codes for all swapping stations. With these two data sets, the cleaning times for the transported product can be retrieved during training.

5.4 Data Cleaning

Some data cleaning steps are executed to provide high-quality data to the neural network:

- Per truck, the long sequence of loading and unloading events during the period 2024-02-18 until 2025-02-18 are split up in smaller parts of three days to resemble the plannig horizon of Nijhof-Wassink.
- The routes with only two loading and unloading locations are filtered from the dataset, as these do not provide meaningful training data.
- Routes outside of the Netherlands, Belgium and Germany are removed due to the absence of high-quality data for the other countries.
- Consecutive duplicate event coordinates in routes are reduced to a single coordinate. The first event is retained while the duplicates are removed.

5.5 Distance Calculation

5.5.1 Haversine Distance

The state requires a calculation method for both detour and direct distances. The most accurate approach would be to utilise the route API, which is also used in other Nijhof-Wassink applications. This API calculates the exact distance along the roads, providing a precise representation of the distances between two locations (for example, between the current truck location and a station). However, the calculation speed of this tool is relatively slow, with each API call taking approximately 0.5 seconds. Given that there are around four waypoints per route and that distances to 150 stations need to be calculated at each waypoint, training for 10,000 episodes would take over 34.7 days. Therefore, it has been decided to use the Haversine formula for all distance calculations as it is a lightweight, geometry-based method.

5.5.2 Detour and Direct Driving Distance

Because the driving distance to a station and the detour to reach a station depend on the event locations between these points, the direct haversine distance between the truck and the station does not always give an accurate representation of the driving distance. The calculation of the driving distance to a station can be divided into two cases. Let the truck be on segment j at location x_j (between waypoints WP_j and WP_{j+1}). Let station k lie on segment ℓ. Let $d(\cdot,\cdot)$ denote point-to-point distance.

Case A: Station on the current segment ($\ell = j$): The truck can drive directly from its current position x_j to the station k, and then continue to the next waypoint WP_{j+1}.

$$\mathrm{Distance}(k) \;=\; d\big(x_j, k\big), \tag{5.5.1}$$

$$\mathrm{Detour}(k) \;=\; d\big(x_j, k\big) \;+\; d\big(k, \mathrm{WP}_{j+1}\big) \;-\; d\big(x_j, \mathrm{WP}_{j+1}\big). \tag{5.5.2}$$

Case B: Station on a later segment ($\ell > j$): The truck must first progress along the planned route to reach segment ℓ. $\mathrm{Distance}(k)$ follows the planned path until the segment that contains the station, after which it goes from WP_ℓ to k. The detour is the local triangular excess on segment ℓ.

$$\mathrm{Distance}(k) \;=\; d\big(x_j, \mathrm{WP}_{j+1}\big) \;+\; \sum_{r=j+1}^{\ell-1} d\big(\mathrm{WP}_r, \mathrm{WP}_{r+1}\big) \;+\; d\big(\mathrm{WP}_\ell, k\big), \tag{5.5.3}$$

$$\mathrm{Detour}(k) \;=\; d\big(\mathrm{WP}_\ell, k\big) \;+\; d\big(k, \mathrm{WP}_{\ell+1}\big) \;-\; d\big(\mathrm{WP}_\ell, \mathrm{WP}_{\ell+1}\big). \tag{5.5.4}$$

Cleaning station calculation: Cleaning stations are reachable from any location. Therefore, the current-segment formulas (5.5.1) and (5.5.2) are reused for cleaning station distance and detour.

5.6 Normalization of Variables

To feed the neural network with meaningful data, normalisation is applied to the input features. The following normalisations are executed every time the feature vectors are created as described in Section 4.5.4.

Fuel level

The fuel level of the truck is normalised with:

$$f_t^{\text{norm}} = \frac{f_t}{F_{\text{Cap}}}$$

Detour

The maximum detour $D_{\max}$ observed during initial experimentation is 400 km, as trucks are allowed to visit all cleaning stations in the buffer. The detours are normalised with the following formula:

$$\widetilde{\delta}_{k,t} = \min\{\text{Detour}(k),\, D_{\max}\}, \qquad \delta_{k,t}^{\text{norm}} = \frac{\widetilde{\delta}_{k,t}}{D_{\max}}.$$

Here $\text{Detour}(k)$ is the detour for station k.

Driving distance

The driving distance to a station is expressed as the amount of fuel required to reach that station. This feature is normalised by dividing the fuel amount required to reach the station by the fuel tank capacity. This can result in numbers exceeding one, which is not typically a standard practice in feature normalisation. However, in this case, it indicates that the truck needs more than a full tank load to reach the concerned station, which can be a signal to the network that it is not worth the extra stop in between to reach this station:

$$d_{k,t}^{\text{norm}} = \frac{\text{Distance(k)}}{F_{\text{cap}}}$$

Fuel price and cleaning costs

In the input vector in Section 4.5.4, the price feature $p_{k,t}$ depends on the type of station that is currently being normalised. If the entry corresponds to a cleaning station, $p_{k,t}$ equals the normalised cleaning costs. If it corresponds to a refuelling station or a resting station, it equals the normalised fuel price. Resting stations also get a fuel price because resting and refuelling may be combined at the same station. This allows the agent to select a resting location with a low fuel price, effectively combining both actions. The fuel prices at gas stations are normalised based on the maximum price across all stations in the Nijhof-Wassink database. Instead of normalising the fuel price by the highest price within the route buffer, this method is used because once the route is completed, the truck may continue to other regions where the maximum prices may vary. A refuelling decision within the current route affects possible actions in the

consecutive route. Using a global $(P_{\min}, P_{\max})$ keeps fuel-price features comparable across routes and regions, so that decisions within the current route remain coherent with potential subsequent routes. The highest and lowest observed fuel prices are €1.6 and €1.1 respectively. The cleaning costs are first clipped by a maximum of 600, which is the maximum observed cleaning costs during initial experimentation. Then the clipped value is divided by the maximum cleaning costs.
At time t on route m, with current segment ℓ, let station $k \in K_{t,m}$. The raw price feature is selected by action type:

$$\widehat{p}_{k,t} = \begin{cases} P_k, & \text{if } a_t^1 \in K_{t,m}^1 \text{ or } a_t^1 \in K_{t,m}^3, \\ C_k, & \text{if } a_t^1 \in K_{t,m}^2. \end{cases}$$

Normalisation is action-specific. Fuel prices use a global range, while cleaning fees are clipped at $C_{\max}$ before scaling:

$$p_{k,t}^{\text{norm}} = \begin{cases} \dfrac{\widehat{p}_{k,t} - P_{\min}}{P_{\max} - P_{\min}}, & \text{if } a_t^1 \in K_{t,m}^1 \text{ or } a_t^1 \in K_{t,m}^3, \\ \dfrac{\min\{\widehat{p}_{k,t}, C_{\max}\}}{C_{\max}}, & \text{if } a_t^1 \in K_{t,m}^2. \end{cases}$$

$$P_{\min} = 1.1, \quad P_{\max} = 1.6 \quad C_{\max} = 600$$

Cleaning time

The cleaning time at a station is normalised by dividing this time by the maximum cleaning time indicated in Table 2.1.

$$w_k^{\text{norm}} = \frac{w_k}{T_{\max}} \qquad T_{\max} = 3 \text{ h}.$$

5.7 Summary

This chapter collects and preprocesses the data needed as input for the Markov Decision Process and Deep Q-learning approach. The historical loading and unloading locations are used to construct the three-day training episodes. Refuelling and cleaning stations, along with their prices, are retrieved from data sets. Furthermore, cleaning times are derived from commodity codes and swapping station locations are incorporated via address codes and GPS coordinates. All inputs are normalised to stabilise training and improve learning abilities. Lastly, the driving distance and detour calculation methods are discussed and finalised.

Chapter 6

Experiment Design and Results

This chapter outlines the experimental design employed in this research. Section 6.1 details the implementation method. Section 6.2 discusses how the train-test split was executed. 6.3 provides an explanation of the Q-network architecture. Next, Section 6.4 details the process of tuning the hyperparameters of the DQN model. Section 6.5 discusses the selection of the final trained model. Subsequently, Section 6.6 provides the performance evaluation method. Finally, 6.9 summarises this Chapter. This Chapter aims to answer the following research questions: *What performance metrics should be used to evaluate the effectiveness of the selected models and solution approaches?* and *How can the different performance metrics be measured and validated in a real-world logistics setting reflecting the operational constraints and decision-making process of Nijhof-Wassink?*

6.1 Software Implementation

The route preparation and deep Q-learning model have been implemented in Python. The entire training procedure is built from scratch, without using a standard library, to implement custom Monte Carlo sampling and a stratified replay buffer. The Q-network itself is programmed with the use of the PyTorch library as referenced in Paszke et al. (2019).

6.2 Train-test Split

To prevent overtraining on the Nijhof-Wassink data set and assess the trained Q-network's performance, a train-validate-test split is performed on the training routes. 80% of the routes are allocated for training, 15% is used to optimise the hyperparameters and 5% is used for testing the performance of the model. This is done to prevent data leakage between the different phases of this research. A fixed random seed is used to ensure reproducibility.

6.3 Q-network Architecture

The architecture of the neural network can largely influence its behaviour and performance. The relation between the architecture and the performance is largely studied in

literature; however, there is no universal optimal design for deep Q-networks. Telgarsky (2016) studied the effects of adding depth to a neural network and concluded that depth provides an exponential gain in expressivity of the neural network. To determine the optimal network architecture for this research, experiments were conducted on the depth, width, and input layer orientation. The depth of the neural network is altered by changing the number of hidden layers, and the width of the network is altered by changing the number of neurons per layer. The input layer orientation is changed by splitting the input layer into two streams: one for the action features and one for the state features. The different networks are first trained on the training data, after which the trained network is used to make decisions for the routes in the test set. The performance of each configuration is measured by calculating the average reward over all the routes in the test data. Table 6.1 presents the results of various architecture experiments. Notably, experiments 12 and 13 feature network designs with a split input layer: one block of neurons for the state features and a second block for the action features. These designs are included to test whether the separation helps the network to differentiate between the two different parts of the state, potentially improving performance. Experiment 10 achieved the highest average reward and is therefore selected as the final network architecture.

Table 6.1: Architecture experiments

Experiment	**Architecture**	**Hidden layers**	**Average reward**
1	64	1	-896.01
2	64_64	2	-723.23
3	128_64	2	-780.35
4	128_128_64	3	-714.38
5	64_128_128_64	4	-665.94
6	256_128_128_64	4	-388.60
7	64_128_128_256	4	-408.37
8	128_256_256_512	4	-555.57
9	512_256_256_128_128_64	6	-686.30
10	1024_512_256_128	4	-361.78
11	1024_512_256_128_64	5	-387.21
12	1024_512 + 1024_512 $\rightarrow$ 128_64	6	-362.83
13	1024_512 + 1024_512 $\rightarrow$ 128	5	-389.29
14	512_512_512	3	-396.23
15	1024_1024_1024	3	-410.01

6.4 Hyperparameter Tuning

In addition to the network architecture, several parameters influence the training stability and final performance of the network. Table 6.2 presents various hyperparameter ranges. These ranges have been established after initial experimentation with the parameter values. There are several strategies for hyperparameter optimisation, including grid search, random search, and Bayesian optimisation. A grid search evaluates every possible combination of hyperparameters. However, this approach scales exponentially with the number of parameter values, which can lead to significant computational complexity when working with larger ranges and multiple parameters. Random search samples combinations from the possible hyperparameter settings, but its success is not guaranteed, and it still spends a considerable amount of time on non-promising regions of the search space. Bayesian optimisation builds a surrogate model and selects new configurations by maximising a function which targets promising regions instead of wasting computational power on non-promising ones (Feurer and Hutter, 2019). The Bayesian hyperparameter optimisation is performed with the continuous ranges provided in Table 6.2. These ranges were established after initial experimentation, concluding that they offered the best initial training results.

Table 6.2: Hyperparameter settings

Parameter	**Experimentation range**
Learning rate (LR)	10^{-5} to 10^{-3}
ε-decay	0.9995 to 0.9997
ε-end	0.01 to 0.15
Batch size	32, 64, 128, 256
Buffer size	10,000; 50,000; 100,000

The Bayesian optimisation is performed using the open-source hyperparameter optimisation library Optuna in Python, as referenced in Akiba et al. (2019). The number of experiments is set to 30 to maintain a reasonable computation time. The duration of each experiment is set to 10000 episodes to ensure that even with the highest epsilon decay, the final epsilon value can still be achieved. Every 100 episodes, a checkpoint network is saved, which will later be used for evaluating the hyperparameter settings.

The hyperparameter optimisation with Optuna and all model training are conducted exclusively on the training portion, while the validation set remains untouched to prevent data leakage. After training, the network checkpoint that achieves the highest average reward over the last 100 episodes in each experiment is used to evaluate the parameter settings on the validation data. The results of all 30 experiments are documented in Section 8.8. Notably, experiment number 4 yields the highest average reward across the validation routes. The hyperparameters from this configuration were selected for training the final model and are given in Table 6.3. The number of episodes is increased to 15000 to assess whether additional training improves the model.

Hyperparameter	Final Value
Learning rate	9.87342×10^{-5}
Discount factor (γ)	1.0
Epsilon start value	1.0
Epsilon end value	0.0839
Epsilon decay	0.99970123
Batch size	64
Replay buffer size	30,000

Table 6.3: Hyperparameter settings

6.5 Trained Model

The final model is trained using the network architecture, hyperparameters given in 6.3 and general settings given in Table 8.3. Figure 6.1 shows the 500-episode moving average reward with a peak observed between episodes 6000 and 15000. To prevent overfitting the network on the training set, checkpoint networks are saved every 100 episodes. The checkpoints are then evaluated on the held-out test set, and the model with the highest mean reward across all test routes is selected. Section 8.11 shows that the highest average return was achieved with checkpoint 14300 which is therefore selected as the final network.

Figure 6.1: Q-network training process

6.6 Performance Evaluation

To evaluate the performance of the DQN model, it is essential to compare other meaningful decision approaches. Therefore, three different performance evaluation methods have been created. First, the MILP model is used to compare the DQN with exact optimal solutions from the MILP model. The second benchmark is a heuristic approach based on simple logical decision rules. While the heuristic may not give an optimal solution, it will give an interpretable baseline to compare the developed DQN approach with. Lastly, a comparison with the current situation gives an indication of whether there is an improvement in the selected KPI values.

For all approaches, evaluation is performed step-by-step along each route according to the MDP. For every run, the truck starts at the first event with an initial fuel level which is the same for a given route. At every step, a decision is made according to the active method. The environment then advances to the next step using the MDP dynamics, and the reward is calculated according to the reward function defined in the MDP. This process continues until the route terminates at the final waypoint or due to early stopping. If a route is terminated early, it is flagged as a penalty route. This protocol ensures a fair comparison among the DQN and the three chosen benchmarks.

6.6.1 Optimal Solution

The first performance evaluation compares the DQN approach with the optimal solutions for each route. A Mixed Integer Linear Program (MILP) is utilized in conjunction with the Simplex solving algorithm to determine the optimal solutions for each route. The mathematical notation for the MILP can be found in 8.6. The optimal solutions are computed using the Gurobi package in Python, as referenced in Gurobi Optimization (2025). The optimal solutions are derived for a set of 16 routes to keep the solving time reasonably small. The set includes routes for four events. Initial experimentation with the solver indicated that routes containing more than four events resulted in a significant increase in solving time. On the other hand, routes with fewer than three events lacked sufficient incentives to execute refuelling and cleaning actions, making them less interesting for analysis.

The overall performance gap between the DQN solutions and the optimal solutions is 13.4%, as calculated using Equation 6.6.1. Table 6.4 presents the average reward per route along with the values of the selected key performance indicators (KPIs). The DQN strategy plans fewer refuelling stops but performs larger volumes of fuel per stop. In contrast, the MILP approach refuels more often, with smaller fuel volumes each time.

Additionally, the MILP exhibits a significantly larger average detour per route, primarily due to the cleaning actions implemented on two specific routes. In these cases, the MILP selects cleaning stations that result in high detours but charge lower cleaning fees. Finally, the average paid fuel price is slightly lower for the MILP compared to the DQN method. The average price per litre for refuelling is higher in the MILP model, which may be due to small, expensive top-ups to reach cheaper cleaning stations.

Metric	**DQN**	**MILP**
Average Total reward	-534.46	-471.85
Std. dev. of total reward	284.20	218.91
Minimum reward	-1337.49	-984.19
Maximum reward	-162.98	-139.43
Average Refuels per route	0.81	1.00
Average Cleans per route	1.75	1.75
Average Rests per route	0	0
Average Detour per route (km)	12.42	150.41
Average refuel volume (L)	489.23	313.21
Average €/L on refuels	1.171	1.176
Failed routes	0/16	0/16

Table 6.4: Comparison of DQN vs. MILP

$$\text{Performane Gap} = \frac{\sum_m \text{DQN}_m - \sum_m \text{OPT}_m}{\sum_m \text{OPT}_m} \tag{6.6.1}$$

6.6.2 Heuristic Approach

The primary objective of the heuristic is to continue driving to the next loading or unloading event along the fixed route. It only interrupts this sequence if driving to the next stop is infeasible. This can be due to one of the following reasons:

- Truck must be cleaned before it can proceed.
- The truck cannot reach the next event due to the remaining allowable driving hours.
- The truck does not have sufficient fuel to reach the next event.

If cleaning is required, the heuristic selects the station with the lowest total cleaning costs, as determined by calculating the total costs of reaching that station and performing a cleaning action. If infeasibility occurs due to insufficient driving hours, the resting station with the smallest detour is selected because detour is the main cost driver for resting decisions. When the fuel level is too low, the heuristic will choose a refuelling station that offers the lowest fuel price. If there is a tie in price, the station with the shortest detour will be selected. In all refuelling cases, the truck is refuelled to its maximum capacity. To prevent the truck from becoming stranded after arriving at a station with only minimal resources left, the heuristic applies a universal safety buffer of 50 litres of fuel and 30 minutes of driving time. This entails that the truck requires 50 litres of fuel and 30 minutes of driving time left after arriving at a location. Figure 6.2 illustrates a flowchart of the heuristic decision-making process. The heuristic and the DQN model will be applied to the created test set, including 338 routes in total. The starting fuel for each route is selected randomly in increments of 50, ranging from 50 to the maximum fuel capacity.

Table 6.5 reports the results from applying the heuristic and the DQN approach to the test set of 338 routes. It can be seen that the DQN outperforms the heuristic on

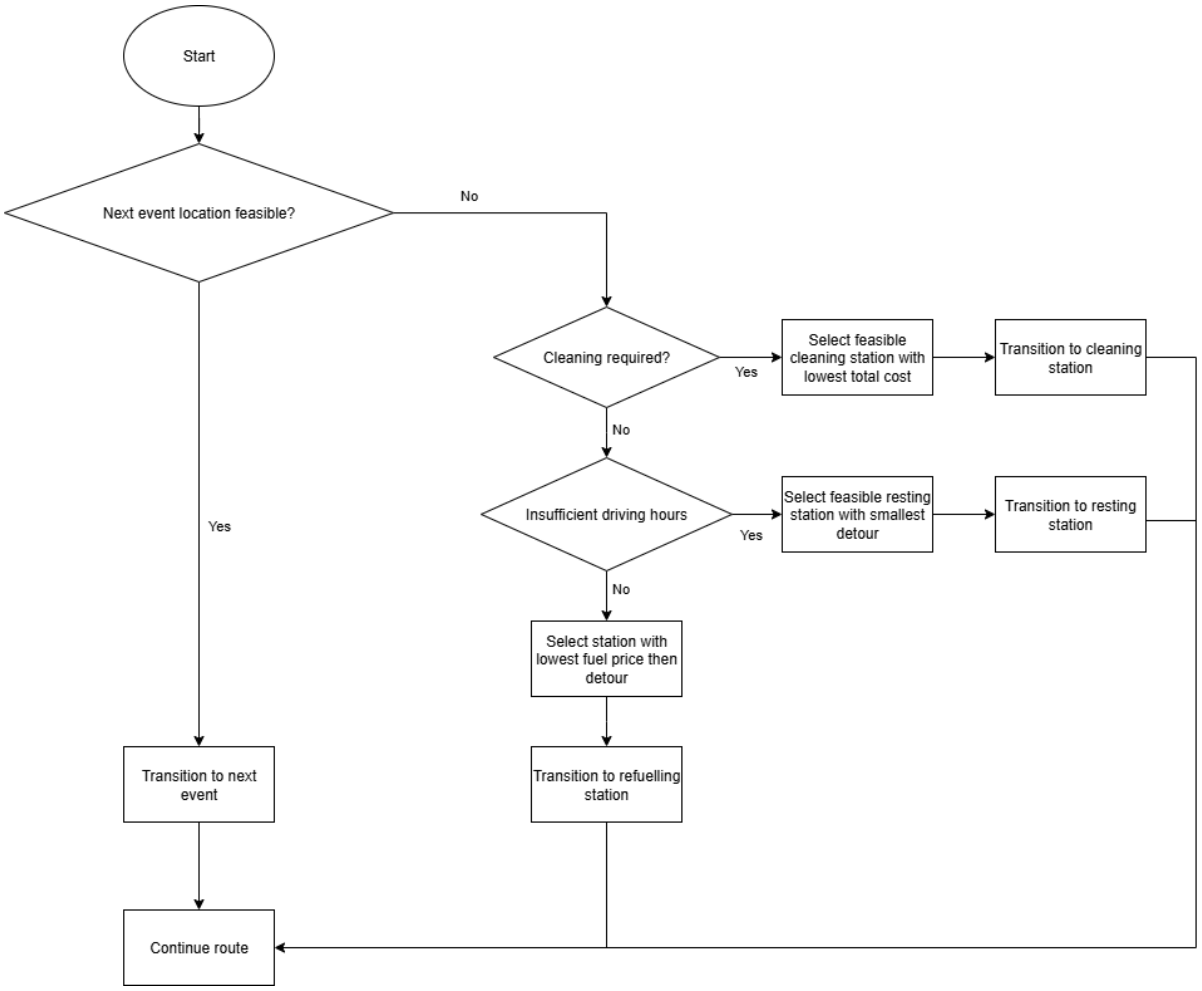


Figure 6.2: Heuristic decision logic

average, while showing similar variability and slightly better worst-case outcomes. Specifically, the DQN has an average mean reward of -410.80, which is 15.93 (3.73%) higher than an average mean reward of -426.73 for the heuristic. The worst reward for the DQN is -1432.39, which is 147.84 higher than the minimum reward of the heuristic approach, which comes down to -1580.23.

Operationally, the DQN executes fewer refuels on average (0.35 vs 0.41), but the amount of litres per refuel is higher on average for the heuristic (438.25 vs 486.76). Cleaning counts are identical, which is expected due to the fixed unloading locations which require a cleaning action afterwards. The DQN rests slightly more often than the heuristic approach, suggesting it is more cautious of ending up without enough driving hours. Finally, the DQN approach yields a higher failure rate (5 routes) than the heuristic approach (2 routes). Failed routes are cases where the agent reaches a state without any valid actions due to insufficient fuel or insufficient driving hours to reach the next location. In these instances, the route is terminated early and excluded from the descriptive statistics.

Metric	DQN	Heuristic
Average total reward	-410.80	-426.73
Std. dev. rewards	233.73	242.55
Minimum reward	-1432.39	-1580.23
Maximum reward	0.00	0.00
Average refuels per route	0.35	0.41
Average cleans per route	1.41	1.41
Average rests per route	0.51	0.37
Average refuel volume	438.25	486.76
Failed routes (%)	1.5%	0.6%
Failed routes (cnt)	5/338	2/338

Table 6.5: Results heuristic comparison

6.6.3 Current Decision Method

The cleaning and refuelling algorithms of Nijhof-Wassink provide drivers with separate recommendations on the cleaning and refuelling locations. Both refuelling and cleaning locations are merely recommendations, and the drivers are free to choose a different location if they consider it more suitable. However, for the purpose of comparing the DQN with the current method, it is assumed that drivers follow the provided advice in all their decisions.

Resting locations, on the other hand, are not included in the current decision process and are entirely chosen by the drivers. As a result, the current approach does not integrate all three separate decisions in the planning framework. To enable a comparison between the two methods, the resting decisions and constraints have been temporarily disabled in the DQN model.

Since the advice on the cleaning and refuelling locations is managed by two different software programs, the complete routes, including the chosen locations, must be constructed manually to make a comparison. A total of 29 additional routes and decisions have been created based on the current decision-making method. The two methods are evaluated using the fuel and cleaning prices on 2024-12-01. The evaluation of the routes is performed with the same settings and parameters used during the training of the DQN.

The current decision-making approach and the DQN model both have been applied to the 29 constructed test routes. On average, the DQN achieved a reward of -611.88, compared to -758.48 for the current approach, resulting in an average absolute improvement of 146.60 (19.33%) per route. Again, the standard deviation in the two approaches is comparable, indicating that the DQN's advantage is not solely driven by a few outliers. Additionally, both the best and worst routes for the DQN approach are superior to the current approach.

The current approach performs one refuel per route on average, whereas the DQN refuels 0.28 times per route on average, which is significantly less often. When DQN

does stop, it fills more per stop (471.25 L vs 421.48 L), resulting in fewer but larger refuelling actions. The mean number of cleanings in both methods is the same (1.97) because cleaning is determined by the fixed unloading locations per route. As discussed earlier, the resting decisions are left out in this evaluation because the current approach does not provide advice on resting locations to the drivers. Finally, both approaches produced 0 failed advices, which means that every route was completed without early termination.

Metric	DQN	Current decision approach
Average reward	-611.88	-758.48
Std dev rewards	329.13	263.91
Min reward	-1527.08	-1580.99
Max reward	188.33	-278.13
Average number of refuels	0.28	1.00
Average number of cleans	1.97	1.97
Average refuel volume	471.25	421.48
Average detour per route (km)	30.71	59.68
Average €/L on refuels	1.26	1.22
Failed routes (%)	0.00%	0.00%
Failed routes (cnt)	0/29	0/29

Table 6.6: Results comparison DQN with current decision approach

6.7 Validation of decision behaviour

To gain a deeper understanding of the DQN's decision-making behaviour, several validation experiments have been conducted. The goal of these experiments is to reflect on the DQN's behaviour with respect to critical decision levers identified in Section 2.3.2.

In Section 8.12.1, the DQN reduced the total number of stops during training, indicating more efficient use of stops while still completing the routes.

Section 8.12.2 demonstrates that DQN steadily decreases the detour to cleaning stations while the paid cleaning fees remain the same on average. This suggests that the DQN learns a trade-off between cleaning price and cleaning detour, with a stronger emphasis on minimising detours. The same section demonstrates that for refuelling actions, the average chosen fuel price decreases over the episodes, along with the average detour to the refuelling stations. This indicates that the DQN effectively minimises costs by reducing both fuel price and detour lengths.

In Section 8.12.3, it is shown that the model is not capable of identifying profitable bridge refuels.

Lastly, 8.12.4 shows that the DQN can perform action combinations. However, due to the policy not being equal to the optimal solutions from the MILP, no conclusions can be made about the efficiency of these combinations.

6.7.1 Sensitivity analysis

To evaluate how the policy of the trained model is influenced by a change in parameter values, two sensitivity analyses are performed. In the first analysis, the fuel price is increased and decreased by 20% from the base level. The second analysis increases and decreases the driving costs per kilometre and per hour by 20%. With every change in parameter value, the network is retrained so it can relearn the optimal policy with the changed parameter values. The trained model is then applied to the test set, after which a selected set of 6 KPIs is evaluated.

Fuel price sensitivity

Figure 6.3 shows the normalised values of the 6 KPIs under the parameter change. It can be seen that the normalised costs increase when the fuel prices are increased, which is a logical consequence. Notably, the increase in total cost is smaller than the 20% fuel price change, indicating that the learned policy adapts to the increase in price.

A more interesting pattern emerges in the other operational KPIs. When prices are increased, the average number of refuelling events per route goes up, while the average litres per refuel go down. The behaviour can be explained due to two reasons. First, because the stopping costs remain constant while fuel costs rise, the relative costs of making an additional stop fall, making it more attractive to split fuel purchase across more steps. Secondly, a price increase by a certain factor widens the absolute gap between the cheapest and most expensive stations, raising the overpayment risk of buying too many litres at a non-cheap station. The policy, therefore, prefers smaller, more frequent top-ups. Furthermore the average detour per route decreases in both the +20% and -20% situation. Because the agent refuels fewer litres per stop, the benefit of making a large detour to a very cheap station becomes smaller. In the –20% fuel-price scenario, the spread between the cheapest and most expensive stations also narrows, so the extra gain from visiting the absolute cheapest station instead of a nearby almost-as-cheap station is limited. In both cases, the policy therefore favours shorter detours and stations located closer to the route.

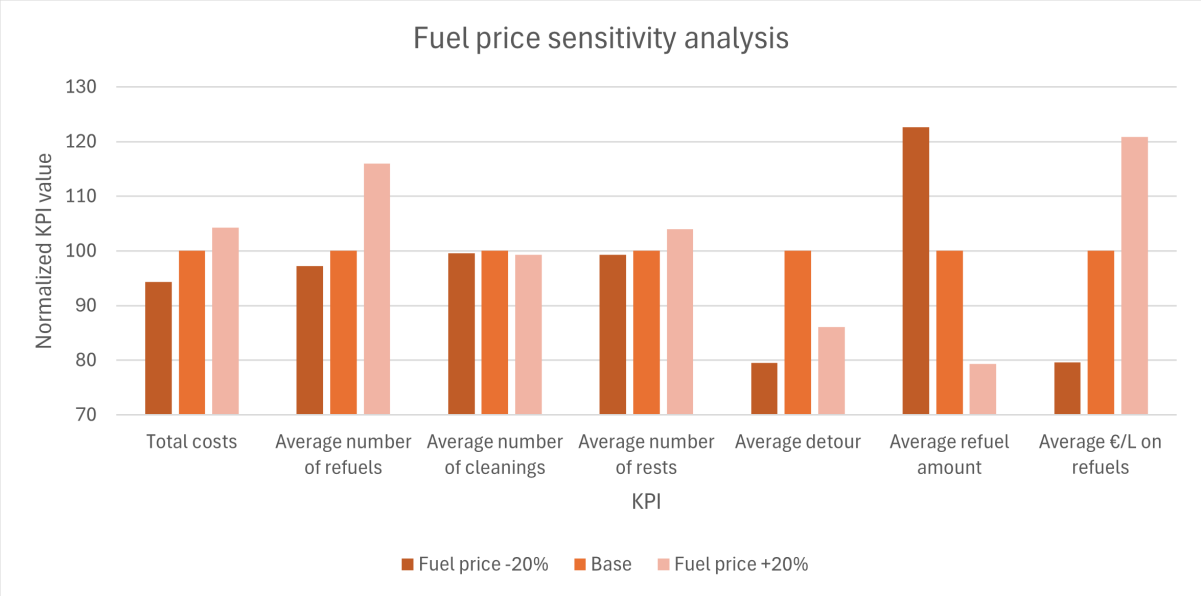


Figure 6.3: Fuel price sensitivity

Driving costs sensitivity

Figure 6.4 shows an increase in the average costs per route when increasing the driving costs by 20%. The average driven detour per route decreases, which is a logical consequence as it is less profitable to drive further to cheaper stations. Because of this, the average price paid per litre on average increased because it becomes less cost-effective to travel farther to cheaper stations.

The figure also shows that the average litres per refuel rise both when driving costs are decreased and when they are increased. One possible explanation for this is as follows: When driving costs are lower, detours become relatively cheap. The policy can then afford to reach the very best-priced stations and then exploit that advantage by refuelling the maximum amount, while skipping the more expensive stations along the route. However, when driving costs are higher, every additional detour or extra stop becomes expensive. When driving costs are higher, every detour and stop becomes costly. In that case, the policy adjusts by refuelling less frequently and purchasing more fuel per stop.

Overall, the DQN adapts by changing its operational strategy. This indicates that the chosen state space is informative and sufficiently expressive, enabling the DQN to reweight trade-offs and rebalance behaviour under parameter shifts.

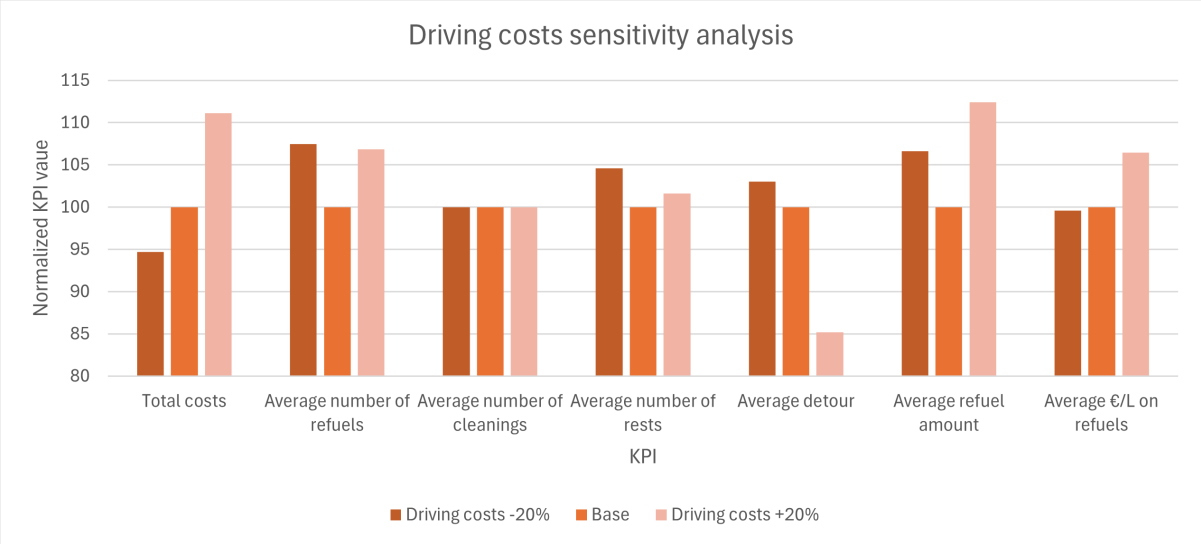


Figure 6.4: Driving costs sensitivity analysis

6.7.2 Model robustness and scenario analysis

To assess the robustness of the model, a scenario was conducted in which half of the refuelling stations were randomly removed from the total set. The model was not retrained after eliminating 50% of the stations in order to evaluate its resilience. As shown in Figure 6.5, the total costs increased by just under 2%. The average detour decreased because there are fewer inexpensive stations that make detours worthwhile. Additionally, the average refuelling amount increased due to the reduced number of available stations. The average number of refuels and cleanings remained stable. The number of rests increased after removing 50% of the stations, indicating the policy is more conservative about taking breaks so it doesn't risk timing out on driving-hours limits. The scenario analysis indicates that the policy is reasonably robust even without retraining: the DQN adapts its operational choices to a sparser refuelling network, adjusting stop frequency and fill sizes to maintain performance with only a minor cost increase.

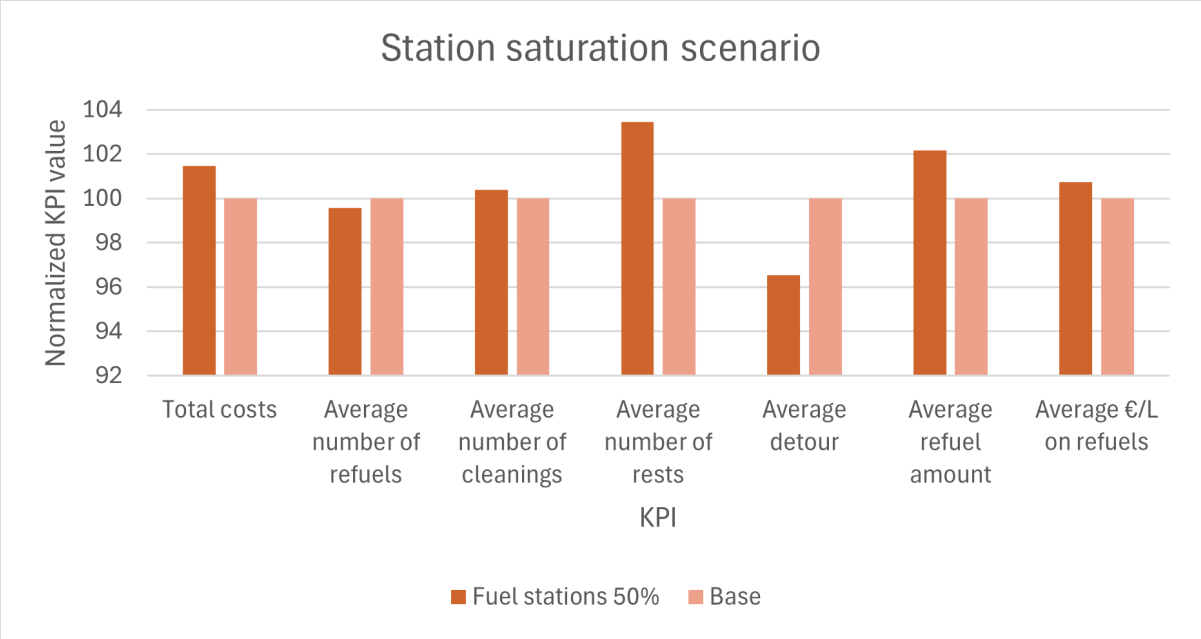


Figure 6.5: Scenario 50% less stations

6.8 Discussion

6.8.1 Strengths of the Proposed Solution

The Deep Q-Network (DQN) policy achieves better performance than both the heuristic and the current decision-making approach on the chosen evaluation protocol, which makes it a viable candidate for supporting route planning decisions. Beyond this quantitative improvement, the proposed solution has several structural strengths.

First, the DQN learns useful operational trade-offs without relying on hand-crafted rules. The validation of decision behaviour in Section 6.7 shows that the policy systematically reduced detour distances for cleaning, refuelling and resting actions while also selecting lower fuel prices per litre.

Second, the model shows robustness to changes in the station network. When the number of fuel stations is reduced to 50% of the original set, the model is able to adapt its policy by increasing the refuelling amount and increasing the refuelling and resting frequency thereby avoiding situations in which the truck truck runs out of fuel or violates rest constraints. This makes the model robust to possible station closures in practice, for example due to maintenance, contractual changes or temporary disruptions in supply.

Third, the solution is adaptable to changes in the costs structure. Sensitivity analysis in 6.7.1 showed that the model is able to extract new policies after retraining the model on the changed fuel prices and driving costs. This flexibility allows the decision aid to function under different cost configurations without redesign of the model. In the long term, it can operate as an all-round planning tool for Nijhof–Wassink, rather than requiring new, tailor-made heuristics for each scenario.

Finally, the distance-based state representation enables generalisation on identical spatial situations. By describing the truck's environment through relative distances to stations rather than absolute coordinates, the policy is not tied to specific regions which allows the model to be applied in other (future) service areas with similar operational constraints.

6.8.2 Limitations and Areas of Improvement

This section summarises the key limitations of the proposed solution. A more detailed discussion of their underlying causes, along with potential solutions to overcome the limitations, is provided in Section 8.13.

- **Failed advices:** The DQN fails to generate a valid advice for a small part of the routes. This is caused by the agent reaching a state with no feasible actions available. In practice, this can be mitigated by validating feasibility one step ahead before committing to an action and, when infeasibility is detected, handing the decision to a simpler rule-based heuristic or alternative optimiser.
- **Haversine distance:** Straight-line distances underestimate real travel distance, which can cause infeasibility to the provided advice. Replacing Haversine distance with distances from a routing API addresses this gap.
- **Bridge refuelling:** The DQN is not able to exploit bridge-refuelling opportunities, which leads to missing low-cost paths and systematically higher fuel spend on some routes. Reducing noise in the station feature block by retaining only the most relevant, cheapest or closest candidates, and rebalancing training exposure toward bridge trajectories could help reveal the underlying trade-off.
- **Limited look-ahead for coordinated stops:** Planning action combinations in advance is hampered by a state that encodes only truck-to-station features and lacks station-to-station geometry. Introducing compact neighbourhood summaries or a graph-based representation enables the policy to anticipate and time combinations more effectively.
- **External uncertainties**: External uncertainties, such as traffic delays or price volatility, are not modelled, which can cause sub-optimality when implementing the generated advice. Incorporating stochastic travel times and price dynamics improves the robustness of the model.

6.8.3 Practical Implications

After discussing the limitations and areas of improvement, the practical usefulness of this research for Nijhof-Wassink is discussed in this section. The trained DQN outperforms the heuristic and the current decision approach on the chosen evaluation protocol. On the constructed evaluation routes, it achieves an average cost reduction of €146.60 per three-day route (19.33%) compared to the current decision approach, directly translating into lower fuel, cleaning and detour-related costs. In addition, the model remains effective when station availability changes and when cost parameters

such as driving costs per kilometre or fuel prices per litre are varied, which makes the solution suitable for a changing operational environment.
This makes the proposed model well-suited to support planners and drivers by listing ranked options, rather than a fully autonomous planner. In practice, the deicision support toll could present a ranked list of refuelling, cleaning actions with associated predicted costs and time trade-offs. Planners and drivers can then combine these suggestions with their operational experience and practical considereations to make a well-balanced choice.

Before the decision tool can be automated and integrated into the daily operations of Nijhof-Wassink, more research and evaluation steps have to be executed. Firstly, the evaluation horizon should be extended when assessing the performance of different decision approaches. This better reveals the long-term performance of the selected actions and gives a fairer comparison to the current situation. Secondly, more extensive research needs to be conducted on various state space representations and the distance calculation method mentioned earlier.

Furthermore, research needs to be conducted on the implementation of the DQN model in the daily operations of Nijhof-Wassink. During this research, historical data were used to simulate a real environment locally on a device. However, during the testing and implementation phases, real-time data on the truck's location, fuel level, and daily prices must be fed into the model to enable real-time decision-making. It is essential to determine on which existing system the DQN will operate and how the data streams will be integrated.

6.9 Summary

This chapter outlines the implementation, tuning, and evaluation of the model. The Markov Decision Process environment and training process are implemented using Python, with a custom Monte Carlo return and stratified replay buffer. The route set is split into a train, validation and test set, each representing 80, 15 and 5 % of the total set, respectively. A seperate neural network architecture research is performed by experimenting with 15 different network structures. It was concluded that the network with size 1024-512-256-128 performed best on the test route set. The hyperparameters of the Deep Q-learning approach are optimised using the Optuna Python library, after which the model is trained on 15000 episodes. The performance of the model is evaluated against three benchmarks: (i) a basic heuristic that prioritises driving to the next event unless this action is infeasible, (ii) a decision method which is currently in the operational testing phase at Nijhof-Wassink and (iii) a Mixed Integer Linear Program that provides the exact optimal solution for a route. It was shown that the DQN consistently outperforms the current decision method and the heuristic. The DQN improved the current decision approach by 19.33% on the mean reward on 29 constructed test routes. The DQN performed 3.73% better than the heuristic. The sensitivity analysis reveals that the DQN can adapt its optimal policy after re-training the network when driving costs and fuel price per litre change. Furthermore, the scenario analysis shows that the DQN is robust under a 50% decrease of the number of refuelling stations without retraining the model. The validation of decision-making behaviour

shows that DQN has difficulty taking advantage of "bridge" refuels. Additional limitations include occasional infeasible advice, using the haversine as a driving distance approximation and the exclusion of external uncertainties in the MDP. The model is a credible starting point for a decision-support tool, but can not yet be implemented as an autonomous planner. Future work should evaluate the approaches on longer decision horizons, refine state representations, consider separate bridge-refuel actions and integrate external uncertainties.

Chapter 7

Conclusion

This chapter provides an overview of the main findings of this study. First, a concise summary of the main results is given in Section 7.1. Then, Section 7.3 distils the main conclusions drawn from these findings. Finally, the recommendations and future research directions are outlined in Section 7.4 to guide further development.

7.1 Summary of Results

This Section provides a concise but comprehensive overview of the main results in this research. After extensively testing the DQN on the case study of Nijhof-Wassink, the main findings are:

- The DQN approach outperformed the heuristic approach with a difference of €15.93 on average per route, which is a decrease of 3.73%
- The DQN approach outperformed the current decision approach with €146.60, which is an improvement of 19.33%
- The DQN approach has an optimality gap of 13.4% with the optimal solutions of the MILP.
- The mean number of refuels decreased with the DQN approach as opposed to both methods.
- In terms of refuel volume per stop, the DQN uses slightly smaller refuelling amounts than the heuristic approach, but larger refuelling amounts than the current decision approach.
- The DQN showed five failed advices as opposed to two for the heuristic approach. In the comparison with the current decision approach, both methods showed zero failed advices.
- Qualitatively, the scenario and sensitivity analyses indicate that the network is robust under changes in station availability and cost parameters, and that the learned policy adapts its refuelling, resting and cleaning decisions flexibly to different operating conditions.

7.2 Contributions

7.2.1 Practical Contributions

This study presents a data-driven decision support prototype that provides integrated advice on refuelling, cleaning, and resting along a fixed multi-day route. The Markov Decision Process models the daily operations of Nijhof-Wassink by incorporating several general assumptions, including route buffers, ahead and behind feasibility, tank capacity, cleaning requirements, and driving-hour constraints. The data pipeline converts historical event logs into usable three-day training episodes and connects station coordinates with prices and cleaning times, after which these variables are normalised. A parametric Q-network is implemented to handle variable candidate sets across different episodes without requiring a large neural network output layer. Three benchmark solutions are created, which can be used to re-evaluate the DQN after implementing improvements. The sensitivity and scenario analysis provide a method to assess the robustness and flexibility of the model.

7.2.2 Theoretical Contributions

This work formalises and names a new problem class: the Fixed-Route vehicle Refuelling, Cleaning and Resting problem (FRV-RCRP), allowing off-route decisions without a minimum refuelling amount requirement. To the best of our knowledge, no prior work has modelled these three operational decisions in a joint decision network. The FRV-RCRP situates itself alongside the FRVRP and the VRTDSP families but extends them by integrating cleaning decisions.

Methodologically, the study contributes a state-action design that exploits invariance properties via relative truck-to-station features while remaining compatible with a variable action set. This was achieved using a parametric Q-network, which takes both state and action features as input and outputs one Q-value per state-action pair. To reduce the bootstrapping bias of rare but economically important large-refuel trajectories, the traditional Temporal Difference target is replaced with full-episode Monte Carlo returns. Additionally, a stratified replay buffer binning the refuel amounts counters the natural overrepresented small refuels under the ϵ-greedy exploration, preventing a biased decision approach. Finally, a terminal value correction is applied to address the finite-horizon MDP, which inherently does not capture the continuous nature of the refuelling dynamics, thereby enabling a fair comparison across different decision approaches. All elements mentioned form a novel DQN variant tailored to the FRV-RCRP and to broader routing decision problems with variable action sets.

7.3 Conclusions

This thesis investigated whether integrating refuelling, cleaning and resting decisions in a single optimisation model can improve the cost efficiency and operational performance of Nijhof-Wassink. The results show that the de proposed Deep Q-network improves expected operational costs by 19.33% compared to the current decision approach and by 3.73% compared to the heuristic benchmark. The agent therefore pro-

vides a credible starting point for an integrated decision-support tool on fixed multi-day routes.

Two main insights emerge. First, the DQN is able to capture key operational trade-offs between detour, timing and fuel or cleaning price without relying on hand-crafted rules. This indicates that the current reward function and state features proved meaningful signals that the neural network can exploit. Secondly, although the network is able to combine actions in several scenarios, its performance on more complex multi-stop combinations remains suboptimal. The model could benefit from a richer station-to-station state geometry and an explicit bridge-refuel action could further increase operational efficiency.

The study also highlights the main limitations of the approach. The state representation entails truck-to-station, which does not include station-to-station information. This can hamper long-horizon coordination of stops. Furthermore, the use of haversine distance and exclusion of external uncertainties can lead to feasibility issues in the operational advice. These findings, in combination with the optimality gap of 13.4%, motivate further work on the state representation, distance calculation and uncertainty modelling.

Finally, the analysis highlights the importance of a balanced scenario representation in the replay buffer. The network can not learn a robust decision policy without a balanced coverage of all critical operational scenarios. Mapping all relevant scenarios in the daily operations of Nijhof-Wassink and adapting the exploration strategy accordingly is an important step towards an implementation-ready decision tool. Overall, the method is promising but not yet ready for direct deployment. Evaluation on a longer horizon and model refinements are required before the model can be tested in a pilot phase.

7.4 Recommendations and Future Research

This Section outlines actionable steps for Nijhof-Wassink to operationalise the integrated decision approach. The recommendations are categorised in short-term, mid-term and long-term to guide the project in the coming period.

Short-term

- **Long horizon evaluation:** to determine the long-term performance of the DQN model with respect to the current decision approach, experiments with longer evaluation horizons have to be executed.
- **Route calculation API:** Implement a route calculation API to ensure correct driving times and fuel usage during training and evaluation of the DQN. Before integrating the API, research the impact on model training time and advice generation time, as well as the additional variable costs associated with API calls.

- **Reward normalisation :** Normalise per-step costs across the three action types so that the rewards are on a comparable scale, which may reduce the bias for one of the three actions.
- **Map operational real-world scenarios:** The scenario analysis highlighted the importance of evenly representing different real-world scenarios in the replay buffer to create a well-calibrated decision agent. To achieve this, exploratory research is needed to define all scenarios and their impact on the operational efficiency of Nijhof-Wassink. Consequently, the exploration method should be adapted to ensure all scenarios are represented in the replay buffer sufficiently.

Mid-term

- **Graph transformer:** Implement a graph transformer to represent the state, which models the truck-to-station geometry and the station-to-station geometry to improve combinatorial decision making between the three action types.
- **Increase bridge-refuel execution frequency:** To enable the model to learn the bridging-refuel trade-off, it is essential to discover and sample more of these trajectories. To accomplish this, an explicit Bridge refuel action should be implemented.

Long-term

- **Different reinforcement learning techniques:** If the desired performance is not achieved after implementing the previous action points, experiments with different reinforcement learning methods, such as Proximal Policy Optimisation (PPO), could be executed. PPO is generally a better choice when continuous actions are required. In this research, the refuelling amount is continuous, which could make PPO a well-suited solution method.
- **Uncertainty implementation:** The model should be trained and evaluated under modelled uncertainties before implementation into the daily operations of Nijhof-Wassink. Stochastic travel times, daily fuel price fluctuations and station queues could be modelled by first measuring their magnitudes and establishing empirical distributions. Report the mean costs and variability statistics to sketch a robust performance picture of the proposed DQN model.

Chapter 8

Appendix

8.1 Appendix A: Problem Cluster

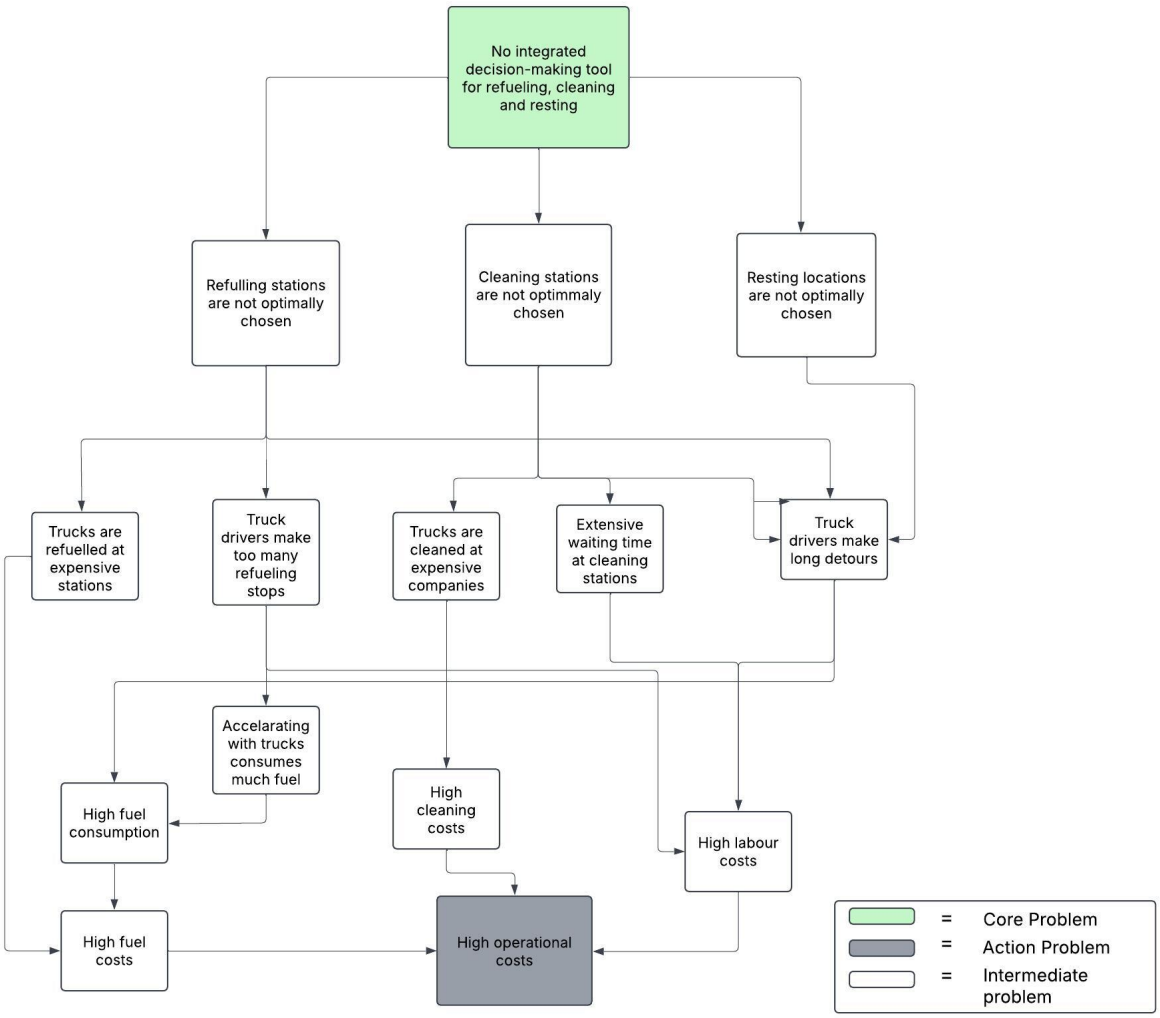


Figure 8.1: Problem cluster

8.2 Appendix B: Relevance of the research

8.2.1 Cost Savings and Profitability

In the problem cluster, three main factors account for Nijhof-Wassink's high operational costs: fuel, cleaning, and labour costs. To analyse the importance of this research on operational efficiency and costs, it is crucial to evaluate the percentage of these three cost factors in total expenses. Personnel costs are the largest at 25.78%, followed by refuelling and cleaning costs at 15.31% and 5.33%, respectively. These percentages emphasise the importance of optimising the decision-making process and decreasing these significant expenses.

Additionally, the profit margin per truck is extremely low. Making one suboptimal decision about the refuelling, cleaning or resting location can significantly reduce the profit margin of that one truck for that day. Doing this for multiple trucks can significantly reduce Nijhof-Wassink's yearly profit margin. To illustrate the severity of these low-profit margins, an example with a refuelling decision is presented.

In Figure 8.2, a historical route is visualised. The truck loaded cargo at the lower pink marker near Gent and had to deposit the cargo near Arnhem at the upper orange marker. In this example, gas stations are visualised with the red and green markers. The truck had just enough fuel to reach Gorinchem. Since the fuel tank was almost empty, the truck driver decided to refuel the tank at the red marker. The diesel price at the red marker was €1.27/L on the date of this route. The price at the green marker was €1.14/L, a difference of 13 cents. Trucks have a tank of 550 litres, and assuming that the driver would have refuelled the truck with 400 litres at both locations, the current decision of refuelling at the red marker would cost an additional (400*0.13=) 52 euros. Referring back to the profit margin per truck per day between 5 and 15 euros, this small decision has erased the profit of at least three trucks that day.

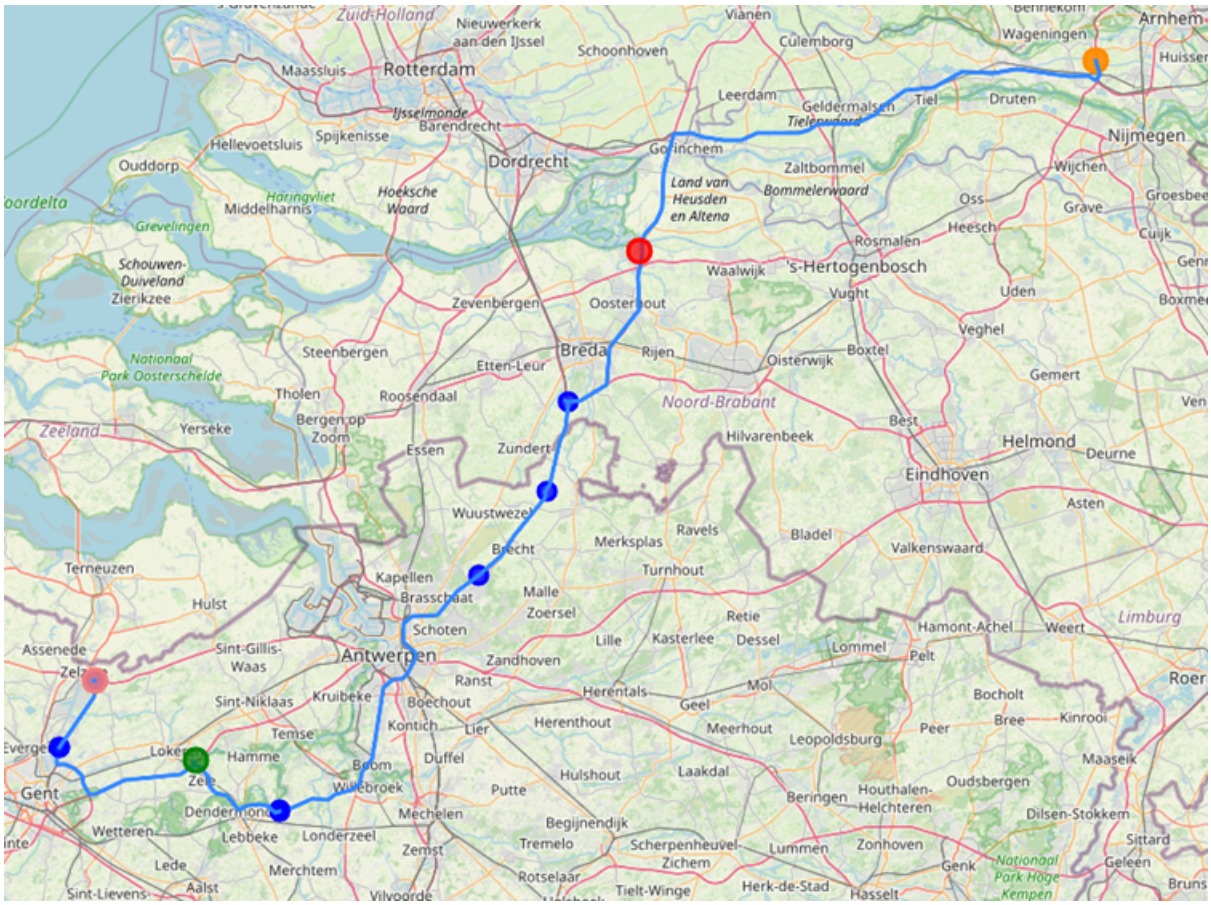


Figure 8.2: Refuelling scenario

8.2.2 Improved Employee Satisfaction

Currently, drivers experience stress from the pressure of finding efficient refuelling and resting locations. This stress leads to a higher cognitive load for drivers, reducing their ability to focus on safety regulations and maintain overall driving performance. Advising drivers about the optimal location could reduce this stress and improve employee satisfaction.

8.2.3 Academic Relevance

Little research exists on integrating the three decisions into one optimisation model. Existing literature focuses on optimising one of the three decisions separately, not including the potential interdependencies between the different decisions. Chapter 3 further discusses the existing relevant research. This research aims to explore the feasibility and effectiveness of combining the three into one decision model. Additionally, several modelling and optimisation approaches are evaluated on computational performance and cost-effectiveness to determine the most suitable approach. The findings in this research could also be used in other problems, such as electric road transportation or Automated Guided Vehicle (AGV) order picking inside warehouses, as these share characteristics similar to those of the current problem.

8.3 Appendix C: Research Design

8.3.1 Problem-solving Approach

A combination of two problem-solving approaches will be used in this thesis to arrive at a fitting solution. The managerial problem-solving method (MPSM) will be used for the problem-identification phase of the research Heerkens and van Winden (2017). The MPSM provides a structured approach to scope down the problem by listing all existing problems inside the company, analysing the cause-and-effect relations of existing problems, and bringing the problem down to a core problem to which no other cause can be found. Then, the core problem with the highest potential for improvement is chosen. The CRISP-MLQ approach is selected for the remaining part of the research as this method is tailored to data-driven problems Studer et al. (2021). There are 6 phases in CRISP-MLQ; however, adaptations are made to this problem-solving approach to better fit the research goal.

To discover the existing body of literature on modelling and solving approaches, a literature review phase is inserted after phase 1. Secondly, the model engineering phase is placed before the data collection and preparation phase, as the data collection process is influenced by the type of model selected. Furthermore, phases 4 and 5 are merged as they aim to validate the model's performance in a test environment. The fifth phase in the CRISP-MLQ approach aims to deploy the created model. However, the proposed model can not be tested in the currently operating decision-making environment during the time span of this research, as errors could significantly affect customer satisfaction and operational costs. Therefore, the models are evaluated in a test environment. This new phase is called "Quality assurance and deployment".

1. **Business and data understanding:** This phase aims to define the business problems and the objectives of the research by constructing a problem cluster. The stakeholders and their interests are mapped, and initial data is gathered to assess the feasibility and relevance of the project.

2. **Data collection and preparation:** The retrieved data is cleaned and preprocessed in this phase to ensure consistency across the data and optimise future model performance. Also, feature engineering is executed to boost future model accuracy.

3. **Model Engineering:** In this phase, the modelling and solution approach are selected. The models are also trained, and initial tests are done after which the optimal solution approach is chosen.

4. **Quality Assurance and Deployment:** Suitable metrics for evaluating model performance are first established. Then, the model parameters are fine-tuned accordingly to optimise model performance.

5. **Monitoring and maintenance:** In this phase, the implementation recommendations and limitations are discussed to ensure that Nijhof-Wassink can make a well-weighted decision about continuing research for future implementation of this model.

8.4 Appendix D: Invariance proof

$$\phi\big(T_g(x)\big) \;=\; \phi(x).$$

Here, $T_g(x)$ is the transformation on a certain feature, the coordinate of a station in this case. If this invariance property is ignored, two geometrically identical situations (e.g. a truck has the same distance to all stations in the two situations) are seen as two distinct states. This results in the agent having to explore each theoretically indifferent state in order to learn the optimal action policy, which harms learning efficiency.

This invariance property can be utilised if we use a different representation for the location of a truck, namely its relative distance to all service stations. Let $p_t = (x_t, y_t) \in \mathbb{R}^2$ and $p_s = (x_s, y_s) \in \mathbb{R}^2$ denote the GPS coordinates of the truck at time t and of station s, respectively. Define the translation operator $T_\Delta : \mathbb{R}^2 \to \mathbb{R}^2$ by

$$T_\Delta(p) = p + \Delta, \qquad \Delta = (\Delta x,\ \Delta y).$$

Under this action, the raw-coordinate feature

$$\phi_{\text{raw}}(p_t) = p_t$$

transforms as

$$\phi_{\text{raw}}\big(T_\Delta(p_t)\big) = T_\Delta(p_t) = p_t + \Delta \;\neq\; \phi_{\text{raw}}(p_t),$$

so every GPS pair changes.

By contrast, define the distance feature

$$d_{t\to s} = \big\|\, p_s - p_t \big\| = \sqrt{(x_s - x_t)^2 + (y_s - y_t)^2}.$$

Applying the same translation to both endpoints gives

$$\begin{aligned} d'_{t\to s} &= \big\|\, T_\Delta(p_s) - T_\Delta(p_t) \big\| \\ &= \sqrt{\big((x_s + \Delta x) - (x_t + \Delta x)\big)^2 + \big((y_s + \Delta y) - (y_t + \Delta y)\big)^2} \\ &= \sqrt{\big(x_s - x_t\big)^2 + \big(y_s - y_t\big)^2} \quad \text{(because the } \Delta \text{ terms cancel inside each square)} \\ &= d_{t\to s}. \end{aligned}$$

Hence the distance map $\phi_{\text{dist}}(p_t, p_s) = d_{t\to s}$ satisfies

$$\phi_{\text{dist}}\big(T_\Delta(p_t),\ T_\Delta(p_s)\big) = \phi_{\text{dist}}(p_t, p_s),$$

And thus this new representation is invariant under translations. Two configurations that differ only by a global shift Δ yield identical distance features, enabling the agent to generalise across locations.

8.5 Appendix E: Q-network architecture

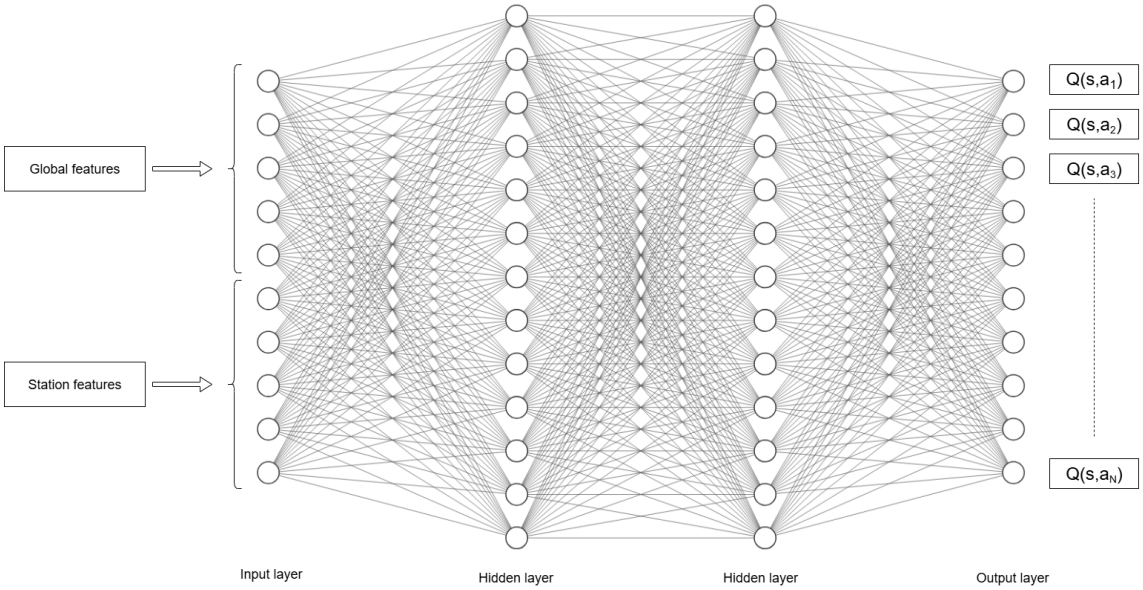


Figure 8.3: Traditional Q-network

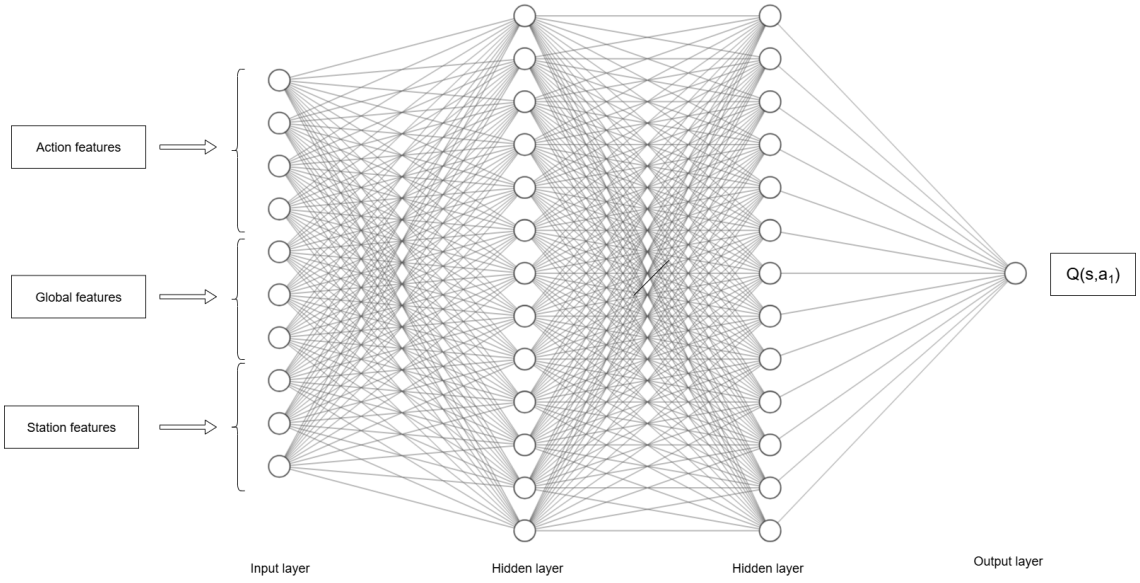


Figure 8.4: Parametric Q-network

8.6 Appendix F: Mixed Integer Linear Program

8.6.1 Sets and Indices

$\mathcal{T}$: set of route segments (between consecutive events)

$\mathcal{F}_t$: candidate fuel stations available in segment $t \in \mathcal{T}$

$\mathcal{C}_t$: candidate cleaning stations available in segment $t \in \mathcal{T}$

$t \in \mathcal{T}$ (segment index), $\quad s, s_1, s_2 \in \mathcal{F}_t,\ s_1 \neq s_2, \quad s \in \mathcal{C}_t$ (when used for cleaners)

A_t, B_t : start and end waypoints of segment t

8.6.2 Parameters

v : average driving speed (km/h)

ρ : fuel consumption (L/km)

$C_{\text{kilometre}}$: truck cost per km

C_{hour} : truck cost per hour

Cap : tank capacity (L)

ΔF : refuel step size (L)

U : $\left\lceil \frac{\text{Cap}}{\Delta F} \right\rceil$ (max steps)

F_{avg} : average terminal fuel value (€/L)

HMAX : max driving hours between rests

C_{stop} : unified stop cost for rest or refuel (€/stop)

p_s : fuel price at fuel station $s \in \mathcal{F}_t$ (€/L)

C_s : cleaning fee at cleaner $s \in \mathcal{C}_t$ (€/stop)

$W_{\text{clean},t,s}$: cleaning time (h) of cleaner $s \in \mathcal{C}_t$ on segment t

StartFuel : initial fuel at start (L)

BIGM : large constant

$d_t^{A,B}$: distance $A_t \to B_t$ (km)

T_t^{base} : base travel time $A_t \to B_t$ (h)

BaseL_t : base fuel $A_t \to B_t$ (L)

$d_{t,s}^{A,S}$, $d_{t,s}^{S,B}$: distances $A_t \to S(s),\ S(s) \to B_t$ (km)

$L_{t,s}^{A,S}$, $L_{t,s}^{S,B}$: liters for those arcs

$d_{t,s}^{A,C}$, $d_{t,s}^{C,B}$: distances $A_t \to C(s),\ C(s) \to B_t$ (km)

$L_{t,s}^{A,C}$, $L_{t,s}^{C,B}$: liters for those arcs

$d_{t,s_1,s_2}^{S,S}$: distance $S(s_1) \to S(s_2),\ s_1 \neq s_2$ (km)

$L_{t,s_1,s_2}^{S,S}$: liters for $S(s_1) \to S(s_2)$

$d_{t,s_1,s_2}^{S,C}$, $d_{t,s_1,s_2}^{C,S}$: distances $S \leftrightarrow C$ (km)

$L_{t,s_1,s_2}^{S,C}$, $L_{t,s_1,s_2}^{C,S}$: liters for $S \leftrightarrow C$

8.6.3 Decision Variables

Selection and quantities

$$z^F_{t,s} \in \{0,1\}: \text{ refuel chosen at fuel station } s$$
$$q_{t,s} \in \mathbb{Z}_{\geq 0}: \text{ integer refuel steps at } s, \;\; q_{t,s} \leq U\, z^F_{t,s}$$
$$z^C_{t,s} \in \{0,1\}: \text{ cleaning chosen at cleaner } s$$
$$y^R_{t,s} \in \{0,1\}: \text{ rest } \textit{with} \text{ refuel at fuel station } s$$
$$z^R_{t,s} \in \{0,1\}: \text{ rest } \textit{only} \text{ at fuel station } s$$
$$\text{stop}_{t,s} \in \{0,1\}: \text{ stop indicator at fuel station } s \text{ (refuel and/or rest)}$$
$$k^F_t \in \{0,1,2\}: \text{ number of refuel stops in segment } t$$
$$k^C_t \in \{0,1\}: \text{ number of cleans in segment } t$$

Logic / ordering

$$b^{\text{both}}_t \in \{0,1\}: \text{ at least one fuel and one clean in } t$$
$$o^{FC}_t \in \{0,1\}: \text{ 1 if Fuel} \to \text{Clean (else Clean} \to \text{Fuel)}$$
$$b^{2F}_t \in \{0,1\}: \text{ indicator that } k^F_t = 2$$
$$w^F_{t,s},\, w^C_{t,s} \in \{0,1\}: \text{ first-action selectors (fuel/clean)}$$
$$w^{FC}_{t,s_1,s_2},\, w^{CF}_{t,s_1,s_2} \in \{0,1\}: \text{ pair selectors (F} \to \text{C / C} \to \text{F)}$$
$$u^{FF}_{t,s_1,s_2} \in \{0,1\}: \text{ ordered pair (fuel then fuel) on } t, \;\; \text{with } s_1 \neq s_2$$
$$v^{(p)}_{t,s_1,s_2,s} \in \{0,1\}, \quad p \in \{0,1,2\}: \text{ triples (two fuels + one clean):}$$
$$p = 0: \; C \to S_1 \to S_2,$$
$$p = 1: \; S_1 \to C \to S_2,$$
$$p = 2: \; S_1 \to S_2 \to C\,.$$

State

$$\text{restAny}_t,\, \Theta_t \in \{0,1\}: \text{ any rest chosen / any action chosen in } t$$
$$C^{\text{ready}}_t \in \{0,1\}: \text{ clean immediately before event } t$$
$$\text{fuel}_t \in [0, \text{Cap}]: \text{ fuel at segment start } t$$
$$H_t \in [0, \text{HMAX}]: \text{ hours left at segment start } t$$
$$\text{endDiff} \in \mathbb{R},\, \text{endGain} \geq 0: \text{ terminal surplus fuel}$$

8.6.4 Constraints

Linking & selection.

$$
\begin{aligned}
& y^R_{t,s} \le z^F_{t,s}, \quad z^R_{t,s} + z^F_{t,s} \le 1 && \forall\, t,\ \forall\, s \in \mathcal{F}_t \\
& \Theta_t \ge \sum_{s\in\mathcal{F}_t} z^F_{t,s}, \quad \Theta_t \ge \sum_{s\in\mathcal{F}_t} z^R_{t,s}, \quad \Theta_t \ge \sum_{s\in\mathcal{C}_t} z^C_{t,s}, && \\
& \Theta_t \le \sum_{s\in\mathcal{F}_t} z^F_{t,s} + \sum_{s\in\mathcal{F}_t} z^R_{t,s} + \sum_{s\in\mathcal{C}_t} z^C_{t,s} && \forall\, t \\
& k^F_t = \sum_{s\in\mathcal{F}_t} z^F_{t,s}, \quad 0 \le k^F_t \le 2, \qquad k^C_t = \sum_{s\in\mathcal{C}_t} z^C_{t,s}, \quad 0 \le k^C_t \le 1 && \forall\, t \\
& b^{\text{both}}_t \le k^F_t, \quad b^{\text{both}}_t \le k^C_t, \quad b^{\text{both}}_t \ge k^F_t + k^C_t - 1 && \forall\, t
\end{aligned}
$$

Stop OR logic (unified stop cost).

$$
\text{stop}_{t,s} \ge z^F_{t,s}, \quad \text{stop}_{t,s} \ge z^R_{t,s}, \quad \text{stop}_{t,s} \le z^F_{t,s} + z^R_{t,s} \quad \forall\, t,\ \forall s \in \mathcal{F}_t.
$$

Selectors and coverage.

$$
\begin{aligned}
& w^F_{t,s} \le z^F_{t,s}, \qquad w^C_{t,s} \le z^C_{t,s}, && \forall\, t,\ \forall\, s \\
& w^{FC}_{t,s_1,s_2} \le z^F_{t,s_1},\ w^{FC}_{t,s_1,s_2} \le z^C_{t,s_2}, \quad w^{CF}_{t,s_2,s_1} \le z^C_{t,s_2},\ w^{CF}_{t,s_2,s_1} \le z^F_{t,s_1} && \forall\, t,\ \forall\, s_1 \in \mathcal{F}_t,\ s_2 \in \mathcal{C}_t
\end{aligned}
$$

$$
\begin{aligned}
\forall\, t,\ \forall\, s \in \mathcal{F}_t: \quad z^F_{t,s} \le\ & w^F_{t,s} + \sum_{s'\in\mathcal{C}_t} w^{FC}_{t,s,s'} + \sum_{s'\in\mathcal{C}_t} w^{CF}_{t,s',s} \\
& + \sum_{\substack{s'\in\mathcal{F}_t \\ s'\neq s}} \left(u^{FF}_{t,s,s'} + u^{FF}_{t,s',s}\right) \\
& + \sum_{\substack{s_1,s_2\in\mathcal{F}_t \\ s_1\neq s_2}} \sum_{s'\in\mathcal{C}_t} \left(v^{(0)}_{t,s_1,s_2,s'} + v^{(1)}_{t,s_1,s_2,s'} + v^{(2)}_{t,s_1,s_2,s'}\right)
\end{aligned}
$$

$$
\begin{aligned}
\forall\, t,\ \forall\, s \in \mathcal{C}_t: \quad z^C_{t,s} \le\ & w^C_{t,s} + \sum_{s_1\in\mathcal{F}_t} w^{FC}_{t,s_1,s} + \sum_{s_1\in\mathcal{F}_t} w^{CF}_{t,s,s_1} \\
& + \sum_{\substack{s_1,s_2\in\mathcal{F}_t \\ s_1\neq s_2}} \left(v^{(0)}_{t,s_1,s_2,s} + v^{(1)}_{t,s_1,s_2,s} + v^{(2)}_{t,s_1,s_2,s}\right)
\end{aligned}
$$

Order consistency for single refuel in segment.

$$
\begin{aligned}
& \sum_{s_1\in\mathcal{F}_t}\sum_{s_2\in\mathcal{C}_t} w^{FC}_{t,s_1,s_2} = o^{FC}_t\, b^{\text{both}}_t, \qquad \sum_{s_2\in\mathcal{C}_t}\sum_{s_1\in\mathcal{F}_t} w^{CF}_{t,s_2,s_1} = (1 - o^{FC}_t)\, b^{\text{both}}_t && \forall\, t \\
& \sum_{s\in\mathcal{F}_t} w^F_{t,s} = \Big(\sum_{s\in\mathcal{F}_t} z^F_{t,s} - b^{\text{both}}_t\Big) + \sum_{s_1\in\mathcal{F}_t}\sum_{s_2\in\mathcal{C}_t} w^{FC}_{t,s_1,s_2} && \forall\, t \\
& \sum_{s\in\mathcal{C}_t} w^C_{t,s} = \Big(\sum_{s\in\mathcal{C}_t} z^C_{t,s} - b^{\text{both}}_t\Big) + \sum_{s_2\in\mathcal{C}_t}\sum_{s_1\in\mathcal{F}_t} w^{CF}_{t,s_2,s_1} && \forall\, t \\
& \sum_{s_1\in\mathcal{F}_t}\sum_{s_2\in\mathcal{C}_t} w^{FC}_{t,s_1,s_2} \le 1 - b^{2F}_t, \qquad \sum_{s_2\in\mathcal{C}_t}\sum_{s_1\in\mathcal{F}_t} w^{CF}_{t,s_2,s_1} \le 1 - b^{2F}_t && \forall\, t
\end{aligned}
$$

Order consistency for double refuels in segment.

$$b_t^{2F} \geq k_t^F - 1, \qquad b_t^{2F} \leq \tfrac{1}{2} k_t^F \qquad \forall\, t \quad (\Leftrightarrow\ b_t^{2F} = 1 \iff k_t^F = 2)$$

$$u_{t,s_1,s_2}^{FF} \leq z_{t,s_1}^F, \quad u_{t,s_1,s_2}^{FF} \leq z_{t,s_2}^F \qquad (s_1 \neq s_2) \qquad \forall\, t, s_1, s_2$$

$$\sum_{s_1 \neq s_2} u_{t,s_1,s_2}^{FF} \;=\; b_t^{2F} \qquad \forall\, t$$

$$v_{t,s_1,s_2,s}^{(p)} \leq u_{t,s_1,s_2}^{FF}, \quad v_{t,s_1,s_2,s}^{(p)} \leq z_{t,s}^C \qquad \forall\, t, s_1 \neq s_2, s \in \mathcal{C}_t,\ p \in \{0,1,2\}$$

$$\sum_{s_1 \neq s_2} \sum_{s \in \mathcal{C}_t} \sum_{p=0}^{2} v_{t,s_1,s_2,s}^{(p)} \;\leq\; b_t^{2F}, \qquad \sum_{s_1 \neq s_2} \sum_{s \in \mathcal{C}_t} \sum_{p=0}^{2} v_{t,s_1,s_2,s}^{(p)} \;\leq\; k_t^C,$$

$$\sum_{s_1 \neq s_2} \sum_{s \in \mathcal{C}_t} \sum_{p=0}^{2} v_{t,s_1,s_2,s}^{(p)} \;\geq\; b_t^{2F} + k_t^C - 1 \qquad \forall\, t$$

Rest indicator (segment-level OR).

$$\text{restAny}_t \geq y_{t,s}^R,\ \text{restAny}_t \geq z_{t,s}^R \ \ \forall s \in \mathcal{F}_t, \qquad \text{restAny}_t \leq \sum_{s \in \mathcal{F}_t} (y_{t,s}^R + z_{t,s}^R) \quad \forall\, t.$$

Refuel headroom.

$$\Delta F \cdot q_{t,s} \leq \text{Cap} - \text{fuel}_t + L_{t,s}^{A,S} + \text{BIGM}(1 - z_{t,s}^F) \qquad \forall\, t,\ \forall\, s \in \mathcal{F}_t.$$

Cleanliness state.

$$C_0^{\text{ready}} = \text{startClean},$$

$$C_{t+1}^{\text{ready}} = 0 \quad \text{if event } t{+}1 \text{ is unload},$$

$$C_{t+1}^{\text{ready}} \geq C_t^{\text{ready}}, \quad C_{t+1}^{\text{ready}} \geq \mathbf{1}\{\exists s \in \mathcal{C}_t : z_{t,s}^C = 1\}, \quad C_{t+1}^{\text{ready}} \leq C_t^{\text{ready}} + \mathbf{1}\{\exists s \in \mathcal{C}_t : z_{t,s}^C = 1\},$$

$$C_{t+1}^{\text{ready}} = 1 \quad \text{if event } t{+}1 \text{ is load}.$$

Hours dynamics (using parameters T_t^{base}). Let $\text{WAIT}_t = \sum_{s\in\mathcal{C}_t} W_{\text{clean},t,s}\, z^C_{t,s}$. Define T_t^{act} as the realized chain time:

$$
\begin{aligned}
T_t^{\text{act}} = (1-\Theta_t)\, T_t^{\text{base}}
&+ \sum_{s\in\mathcal{F}_t} \Big(\tfrac{d^{A,S}_{t,s}}{v} + \tfrac{d^{S,B}_{t,s}}{v}\Big)\Big(z^F_{t,s} - \sum_{s'\in\mathcal{C}_t} w^{FC}_{t,s,s'} - \sum_{s'\neq s} u^{FF}_{t,s,s'} - \sum_{s'\neq s} u^{FF}_{t,s',s}\Big)\\
&+ \sum_{s\in\mathcal{C}_t} \Big(\tfrac{d^{A,C}_{t,s}}{v} + \tfrac{d^{C,B}_{t,s}}{v}\Big)\Big(z^C_{t,s} - \sum_{s_1\in\mathcal{F}_t} w^{CF}_{t,s,s_1} - \sum_{s_1\in\mathcal{F}_t} w^{FC}_{t,s_1,s} - \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}} \big(v^{(0)}_{t,s_1,s_2,s} + v^{(1)}_{t,s_1,s_2,s} + v^{(2)}_{t,s_1,s_2,s}\big)\Big)\\
&+ \sum_{s\in\mathcal{F}_t} \Big(\tfrac{d^{A,S}_{t,s}}{v} + \tfrac{d^{S,B}_{t,s}}{v}\Big) z^R_{t,s}\\
&+ \sum_{s_1\in\mathcal{F}_t}\sum_{s_2\in\mathcal{C}_t} \Big(\tfrac{d^{A,S}_{t,s_1}}{v} + \tfrac{d^{S,C}_{t,s_1,s_2}}{v} + \tfrac{d^{C,B}_{t,s_2}}{v}\Big) w^{FC}_{t,s_1,s_2}\\
&+ \sum_{s_2\in\mathcal{C}_t}\sum_{s_1\in\mathcal{F}_t} \Big(\tfrac{d^{A,C}_{t,s_2}}{v} + \tfrac{d^{C,S}_{t,s_2,s_1}}{v} + \tfrac{d^{S,B}_{t,s_1}}{v}\Big) w^{CF}_{t,s_2,s_1}\\
&+ \sum_{s_1\neq s_2\in\mathcal{F}_t} \Big(\tfrac{d^{A,S}_{t,s_1}}{v} + \tfrac{d^{S,S}_{t,s_1,s_2}}{v} + \tfrac{d^{S,B}_{t,s_2}}{v}\Big) u^{FF}_{t,s_1,s_2}\\
&+ \sum_{s_1\neq s_2\in\mathcal{F}_t}\sum_{s\in\mathcal{C}_t} \Big(\tfrac{d^{A,C}_{t,s}}{v} + \tfrac{d^{C,S}_{t,s,s_1}}{v} + \tfrac{d^{S,S}_{t,s_1,s_2}}{v} + \tfrac{d^{S,B}_{t,s_2}}{v}\Big) v^{(0)}_{t,s_1,s_2,s}\\
&+ \sum_{s_1\neq s_2\in\mathcal{F}_t}\sum_{s\in\mathcal{C}_t} \Big(\tfrac{d^{A,S}_{t,s_1}}{v} + \tfrac{d^{S,C}_{t,s_1,s}}{v} + \tfrac{d^{C,S}_{t,s,s_2}}{v} + \tfrac{d^{S,B}_{t,s_2}}{v}\Big) v^{(1)}_{t,s_1,s_2,s}\\
&+ \sum_{s_1\neq s_2\in\mathcal{F}_t}\sum_{s\in\mathcal{C}_t} \Big(\tfrac{d^{A,S}_{t,s_1}}{v} + \tfrac{d^{S,S}_{t,s_1,s_2}}{v} + \tfrac{d^{S,C}_{t,s_2,s}}{v} + \tfrac{d^{C,B}_{t,s}}{v}\Big) v^{(2)}_{t,s_1,s_2,s}\\
&+ \text{WAIT}_t.
\end{aligned}
$$

Hours update with optional reset (rest):

$$
\begin{aligned}
&H_{t+1} \le H_t - T_t^{\text{act}} + \text{BIGM}\,\text{restAny}_t,\\
&H_{t+1} \ge H_t - T_t^{\text{act}} - \text{BIGM}\,\text{restAny}_t,\\
&H_{t+1} \le \text{HMAX} + \text{BIGM}(1-\text{restAny}_t), \qquad H_{t+1} \ge \text{HMAX} - \text{BIGM}(1-\text{restAny}_t).
\end{aligned}
$$

Reachability guards (hours/fuel).

$$
\begin{aligned}
&H_t \ge \tfrac{d^{A,S}_{t,s}}{v} - \text{BIGM}\big(1-(w^F_{t,s}+z^R_{t,s})\big), \quad \text{fuel}_t \ge L^{A,S}_{t,s} - \text{BIGM}\big(1-(w^F_{t,s}+z^R_{t,s})\big) && \forall t,\ s\in\mathcal{F}_t\\
&H_t \ge \tfrac{d^{A,C}_{t,s}}{v} - \text{BIGM}(1-w^C_{t,s}), \quad \text{fuel}_t \ge L^{A,C}_{t,s} - \text{BIGM}(1-w^C_{t,s}) && \forall t,\ s\in\mathcal{C}_t\\
&H_t \ge \tfrac{d^{A,S}_{t,s_1}}{v} + \tfrac{d^{S,S}_{t,s_1,s_2}}{v} - \text{BIGM}(1-u^{FF}_{t,s_1,s_2}),\\
&\qquad \text{fuel}_t \ge L^{A,S}_{t,s_1} + L^{S,S}_{t,s_1,s_2} - \text{BIGM}(1-u^{FF}_{t,s_1,s_2}) && \forall t,\ s_1\neq s_2\in\mathcal{F}_t
\end{aligned}
$$

(analogous guards for each triple position $v^{(0/1/2)}_{t,s_1,s_2,s}$)

Fuel balance. Let L_t^{act} mirror T_t^{act} with liters:

$$
\begin{aligned}
L_t^{\text{act}} = (1-\Theta_t)\,\text{BaseL}_t
&+ \sum_{s_1\in\mathcal{F}_t} \left(L_{t,s_1}^{A,S} + L_{t,s_1}^{S,B}\right)\left(z_{t,s_1}^{F} - \sum_{c\in\mathcal{C}_t} w_{t,s_1,c}^{FC} - \sum_{\substack{s_2\in\mathcal{F}_t\\ s_2\neq s_1}} u_{t,s_1,s_2}^{FF} - \sum_{\substack{s_2\in\mathcal{F}_t\\ s_2\neq s_1}} u_{t,s_2,s_1}^{FF}\right)\\
+ \sum_{c\in\mathcal{C}_t} \left(L_{t,c}^{A,C} + L_{t,c}^{C,B}\right)\Big(z_{t,c}^{C} - \sum_{s_1\in\mathcal{F}_t} w_{t,c,s_1}^{CF} - \sum_{s_1\in\mathcal{F}_t} w_{t,s_1,c}^{FC} - \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}} \left(v_{t,s_1,s_2,c}^{(0)} + v_{t,s_1,s_2,c}^{(1)} + v_{t,s_1,s_2,c}^{(2)}\right)\Big)\\
&+ \sum_{s_1\in\mathcal{F}_t} \left(L_{t,s_1}^{A,S} + L_{t,s_1}^{S,B}\right) z_{t,s_1}^{R}\\
&+ \sum_{s_1\in\mathcal{F}_t}\sum_{c\in\mathcal{C}_t} \left(L_{t,s_1}^{A,S} + L_{t,s_1,c}^{S,C} + L_{t,c}^{C,B}\right) w_{t,s_1,c}^{FC}\\
&+ \sum_{c\in\mathcal{C}_t}\sum_{s_1\in\mathcal{F}_t} \left(L_{t,c}^{A,C} + L_{t,c,s_1}^{C,S} + L_{t,s_1}^{S,B}\right) w_{t,c,s_1}^{CF}\\
&+ \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}} \left(L_{t,s_1}^{A,S} + L_{t,s_1,s_2}^{S,S} + L_{t,s_2}^{S,B}\right) u_{t,s_1,s_2}^{FF}\\
&+ \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}}\sum_{c\in\mathcal{C}_t} \left(L_{t,c}^{A,C} + L_{t,c,s_1}^{C,S} + L_{t,s_1,s_2}^{S,S} + L_{t,s_2}^{S,B}\right) v_{t,s_1,s_2,c}^{(0)}\\
&+ \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}}\sum_{c\in\mathcal{C}_t} \left(L_{t,s_1}^{A,S} + L_{t,s_1,c}^{S,C} + L_{t,c,s_2}^{C,S} + L_{t,s_2}^{S,B}\right) v_{t,s_1,s_2,c}^{(1)}\\
&+ \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}}\sum_{c\in\mathcal{C}_t} \left(L_{t,s_1}^{A,S} + L_{t,s_1,s_2}^{S,S} + L_{t,s_2,c}^{S,C} + L_{t,c}^{C,B}\right) v_{t,s_1,s_2,c}^{(2)}.
\end{aligned}
$$

The fuel level at time step $t+1$ can then be defined as:

$$
\text{fuel}_{t+1} = \text{fuel}_t - L_t^{\text{act}} + \sum_{s_1\in\mathcal{F}_t} \Delta F\, q_{t,s_1}\,.
$$

Detour and costs (no base driving cost). Segment detour (km):

$$
\begin{aligned}
\text{DetourKm}_t = &\sum_{s_1\in\mathcal{F}_t} \left(d_{t,s_1}^{A,S} + d_{t,s_1}^{S,B} - d_t^{A,B}\right)\left(z_{t,s_1}^{F} - \sum_{c\in\mathcal{C}_t} w_{t,s_1,c}^{FC} - \sum_{\substack{s_2\in\mathcal{F}_t\\ s_2\neq s_1}} u_{t,s_1,s_2}^{FF} - \sum_{\substack{s_2\in\mathcal{F}_t\\ s_2\neq s_1}} u_{t,s_2,s_1}^{FF}\right)\\
+ \sum_{c\in\mathcal{C}_t} \left(d_{t,c}^{A,C} + d_{t,c}^{C,B} - d_t^{A,B}\right)\Big(z_{t,c}^{C} - \sum_{s_1\in\mathcal{F}_t} w_{t,c,s_1}^{CF} - \sum_{s_1\in\mathcal{F}_t} w_{t,s_1,c}^{FC} - \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}} \left(v_{t,s_1,s_2,c}^{(0)} + v_{t,s_1,s_2,c}^{(1)} + v_{t,s_1,s_2,c}^{(2)}\right)\Big)\\
&+ \sum_{s_1\in\mathcal{F}_t} \left(d_{t,s_1}^{A,S} + d_{t,s_1}^{S,B} - d_t^{A,B}\right) z_{t,s_1}^{R}\\
&+ \sum_{s_1\in\mathcal{F}_t}\sum_{c\in\mathcal{C}_t} \left(d_{t,s_1}^{A,S} + d_{t,s_1,c}^{S,C} + d_{t,c}^{C,B} - d_t^{A,B}\right) w_{t,s_1,c}^{FC}\\
&+ \sum_{c\in\mathcal{C}_t}\sum_{s_1\in\mathcal{F}_t} \left(d_{t,c}^{A,C} + d_{t,c,s_1}^{C,S} + d_{t,s_1}^{S,B} - d_t^{A,B}\right) w_{t,c,s_1}^{CF}
\end{aligned}
$$

$$
\begin{aligned}
&+ \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}} \left(d^{A,S}_{t,s_1} + d^{S,S}_{t,s_1,s_2} + d^{S,B}_{t,s_2} - d^{A,B}_{t}\right) u^{FF}_{t,s_1,s_2} \\
&+ \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}} \sum_{c\in\mathcal{C}_t} \left(d^{A,C}_{t,c} + d^{C,S}_{t,c,s_1} + d^{S,S}_{t,s_1,s_2} + d^{S,B}_{t,s_2} - d^{A,B}_{t}\right) v^{(0)}_{t,s_1,s_2,c} \\
&+ \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}} \sum_{c\in\mathcal{C}_t} \left(d^{A,S}_{t,s_1} + d^{S,S}_{t,s_1,s_2} + d^{S,C}_{t,s_2,c} + d^{C,B}_{t,c} - d^{A,B}_{t}\right) v^{(2)}_{t,s_1,s_2,c} \\
&+ \sum_{\substack{s_1,s_2\in\mathcal{F}_t\\ s_1\neq s_2}} \sum_{c\in\mathcal{C}_t} \left(d^{A,S}_{t,s_1} + d^{S,C}_{t,s_1,c} + d^{C,S}_{t,c,s_2} + d^{S,B}_{t,s_2} - d^{A,B}_{t}\right) v^{(1)}_{t,s_1,s_2,c}\,.
\end{aligned}
$$

Per-segment monetary cost:

$$
\begin{aligned}
\mathrm{Cost}_t = &\left(C_{\text{kilometre}} + \tfrac{C_{\text{hour}}}{v}\right) \mathrm{DetourKm}_t \\
&+ \sum_{s\in\mathcal{F}_t} p_s\,\Delta F\, q_{t,s} \\
&+ \sum_{s\in\mathcal{F}_t} C_{\text{stop}} \cdot \mathrm{stop}_{t,s} \\
&+ \sum_{s\in\mathcal{C}_t} \left(C_s + C_{\text{hour}} \cdot W_{\text{clean},t,s}\right) z^{C}_{t,s}.
\end{aligned}
$$

8.6.5 Objective

Let fuel_T and be the fuel level at the last event. The objective function is then given as:

$$
\max \quad F_{\text{avg}} \cdot \mathrm{fuel}_T \;-\; \sum_{t\in\mathcal{T}} \mathrm{Cost}_t.
$$

8.7 Appendix G: Data tables

8.7.1 Gas station data

Table 8.1: Gas station data set

Station metadata						Daily price columns			
stationID	**Name**	**Country**	**Latitude**	**Longitude**	**Region number**	**2024-01-01**	**2024-01-02**	**…**	**2025-01-25**
767	G&V Paliseul	BE	49.898285	5.123751	21	124,25	124,25	…	120,7
768	G&V Maissin	BE	49.966376	5.181373	21	124,25	124,25	…	120,7
769	Esso Saint-Hubert	BE	50.0271249	5.3738139	21	124,25	124,25	…	120,7
770	Texaco Rochefort	BE	50.1574562	5.2323675	21	124,25	124,25	…	120,7
771	GV MARCHE-EN-FAMENNE	BE	50.2193	5.34669	21	124,25	124,25	…	120,7
772	Station Thuillies	BE	50.300761	4.332872	21	124,25	124,25	…	120,7
773	Esso Express Montignies Sur Sa	BE	50.4098639	4.4782821	21	124,25	124,25	…	120,7
774	Pumpy 5	BE	50.4257724	4.4893258	21	124,25	124,25	…	120,7
775	Pumpy Courcelles	BE	50.4626385	4.358485	21	124,25	124,25	…	120,7
776	SA TAHON	BE	50.4742199	3.90166	21	124,25	124,25	…	120,7
777	Esso Express Eghezee	BE	50.5983609	4.9048	21	124,25	124,25	…	120,7
778	G&V CNG TOURNAI OUEST	BE	50.6171108043	3.3175852804	21	124,25	124,25	…	120,7
779	Esso Express Liege Grivegnee	BE	50.618787	5.5958443	21	124,25	124,25	…	120,7
780	G&V Herstal	BE	50.658018	5.620619	21	124,25	124,25	…	120,7
⋮	(more stations below)								

8.7.2 Route Data

Table 8.2: Route data column structure

Column Name	**Description**	**Used in Model**
actionState	Status of the action (e.g., finished)	No
contactName	Name of the customer or contact person	No
orderNumber	Order ID or reference number	No
productCode	Code identifying the product	No
productName	Name/description of the transported product	No
DATE	Timestamp of the action	Yes
TRUCK	Truck identifier (license plate or internal ID)	Yes
TRAILER	Trailer identifier	Yes
departmentName	Internal department or region responsible	No
addressCode	Code for the delivery/pickup location	No
addressName	Name of the company/location	No
streetName	Street part of the location address	No
cityName	City part of the location address	No
x	Longitude coordinate of the location	Yes
y	Latitude coordinate of the location	Yes
aktieid	Internal ID for the loading/unloading event	No
ACTION	Type of action (e.g., Laden = loading, Lossen = unloading)	Yes

8.8 Appendix H: Hyperparameter optimization experiments

number	value	lr	eps_end	eps_decay	batch_size	buffer_size	best_ma100	best_ep	episodes_trained
0	-382,3378703	8,46801E-05	0,143100003	0,998440372	32	10000	-354,604138	5113	10000
1	-403,1976641	1,12458E-05	0,145787379	0,99891248	256	30000	-385,9440989	9975	10000
2	-377,766134	0,000327819	0,02952914	0,99637308	128	50000	-285,994278	3161	10000
3	-384,2955546	0,000319862	0,033873377	0,995305742	64	50000	-280,3848939	2679	10000
4	-339,5165601	9,87342E-05	0,08391247	0,99970123	64	30000	-314,530226	9736	15000
5	-372,5441235	0,002522195	0,118518595	0,999415645	128	100000	-342,9012461	7494	10000
6	-397,7473778	9,17911E-05	0,047988864	0,998895066	128	30000	-311,4892945	8537	10000
7	-391,4553443	0,000818359	0,037820195	0,995025954	32	50000	-296,8184398	6747	10000
8	-396,4215259	0,001374079	0,097261738	0,996555221	256	50000	-322,4631775	7615	10000
9	-394,8980094	1,97811E-05	0,10985427	0,99857569	64	10000	-334,9351985	7499	10000
10	-369,3835959	2,71892E-05	0,04142803	0,996495261	32	30000	-272,4922416	9372	10000
11	-376,4632695	4,41633E-05	0,089174987	0,995462431	32	30000	-300,4746189	7228	10000
12	-393,8300993	3,645E-05	0,04324108	0,998085803	32	50000	-266,0537329	6780	10000
13	-384,2933082	1,05105E-05	0,022039035	0,99514469	256	50000	-281,4821556	8840	10000
14	-417,6522877	2,267E-05	0,048198797	0,996165523	32	10000	-311,1122431	6780	10000
15	-407,6735054	1,89887E-05	0,069079921	0,996440098	64	50000	-312,3122941	5487	10000
16	-714,0830626	1,27652E-05	0,128030744	0,99703413	32	50000	-362,6969207	6053	10000
17	-350,7848387	1,76337E-05	0,046025598	0,995793191	128	100000	-296,6831762	9025	10000
18	-393,8868613	8,74894E-05	0,068565471	0,995334785	128	30000	-313,9509434	4387	10000
19	-413,4921508	1,43462E-05	0,069654764	0,997316959	128	100000	-304,9381775	9866	10000
20	-388,946889	1,74846E-05	0,131474808	0,995469358	128	10000	-322,2253116	5379	10000
21	-359,312276	1,13503E-05	0,027717784	0,995472068	128	100000	-297,0556726	9671	10000
22	-387,9304144	3,12769E-05	0,028601689	0,995038369	128	100000	-257,784929	9621	10000
23	-389,8592185	1,558E-05	0,075577713	0,995378793	32	30000	-343,4125812	8350	10000
24	-362,5626039	1,6423E-05	0,011752479	0,997145303	128	100000	-254,1820508	8580	10000
25	-402,1704632	1,50562E-05	0,058172125	0,995089796	128	30000	-306,6132257	4988	10000
26	-400,9064113	8,96169E-05	0,056421721	0,996926134	128	10000	-303,6417385	4158	10000
27	-438,6543101	1,66114E-05	0,032179012	0,996507309	64	100000	-288,0809514	9491	10000
28	-368,8032261	1,80627E-05	0,0320316	0,995500773	128	10000	-293,2573325	6930	10000
29	-390,3508066	8,02683E-05	0,116076223	0,996377972	64	100000	-325,5176309	7330	10000

8.9 Appendix I: Input parameters

Parameter	Symbol	Value/Source	Units	Justification
Max stations per route	N	500	–	Expert opinion
Max refuels per route		3	–	Section 8.10.1
Max refuels per segment		2	–	Section 8.10.2
Route buffer size fuel stations		50	km	Section 8.10.3
Route buffer size cleaning stations		80	km	Section 8.10.4
Average driving speed	v	80	km/h	Expert opinion
Fuel consumption	ρ	0.33	L/km	Expert opinion
Early stopping penalty	p_{novalid}	1500	€	Section 8.10.5
Truck cost per km	$C_{\text{kilometre}}$	X	€/km	Expert opinion
truck cost per hour	C_{hour}	X	€/h	Expert opinion
Fuel tank capacity	Cap	550	L	Expert opinion
Refuel step size	ΔF	50	L	Section 8.10.6
Average terminal fuel value	F_{avg}	1.2	€/L	Data analysis
Stopping costs	C_{stop}	15	€/stop	Expert opinion
Fuel price at station	$p(a_t^1)$	Data set	€/L	Data analysis
Cleaning Costs	$C_{\text{clean}}(a_t^1)$	Data set	€/stop	Data analysis
Cleaning Waiting time	W_{clean}	Data set	€/stop	Data analysis
Initial fuel	StartFuel	Random	L	-
Learning rate		9.87e+00	–	Section 6.4
Discount factor		1.0	–	Section 6.4
Epsilon start		1.0	–	Section 6.4
Epsilon end		0.0839	–	Section 6.4
Epsilon decay		0.99970123	–	Section 6.4
Batch size		64	–	Section 6.4
Replay buffer size		30000	–	Section 6.4

Table 8.3: Model and routing parameters.

8.10 Appendix J: Parameter explanations

Some general settings require justification, as they cannot be derived from the data or established after extensive discussions with experts within Nijhof-Wassink. The following sections explain why specific values have been assigned to parameters.

8.10.1 Maximum allowable refuels per route

The maximum allowable refuels per route caps the number of refuels per route to prevent the agent from exploring degenerate regions of the policy range. The cap is set to 3, allowing for delayed full refuels but prohibiting many small refuels. Initial experimentation showed that more than 3 refuels per route is never efficient with the current maximum route size.

8.10.2 Maximum allowable refuels per segment

This caps the number of refuels per segment to prevent the agent from exploring degenerate regions of the policy range. The cap is set to 2, allowing for delayed full refuels. More than two refuels within the same segment are never cost-effective under the current reward function.

8.10.3 Route buffer range gas stations

The inclusion buffer for gas stations is set to 50 kilometres. This range keeps the running time reasonably low while still including a significant number of refuelling stations per route.

8.10.4 Route buffer range cleaning stations

The route buffer range for the cleaning stations is set to 80 kilometres so that the truck can reach a valid cleaning station for every product in the route set

8.10.5 Early stopping penalty

If the agent has no valid actions in the current state, the episode is terminated, and a penalty of 1500 is incurred. Several experiments have shown that setting the penalty too high lets the agent make pathological decisions, such as constantly topping up small amounts to avoid lower fuel levels. Setting the penalty too low results in the agent running out of fuel in many episodes.

8.10.6 Fuel amount step size

To keep the action space manageable, a refueling step size of 50L was chosen. Initial experimentation showed that decreasing the step size to 10L significantly increased the running time per episode, which greatly extended the training time.

8.11 Appendix K: Network checkpoint evaluation

Table 8.4: Average rewards for different experiment checkpoints on test route set.

Experiment	Average Reward
6000	-285.47
6100	-285.35
6200	-285.01
6300	-289.70
6400	-301.06
6500	-298.23
6600	-299.08
6700	-291.68

continued on next page

Table 8.4 — continued from previous page

Experiment	Average Reward
6800	-300.60
6900	-291.62
7000	-298.73
7100	-309.50
7200	-299.06
7300	-289.48
7400	-301.37
7500	-297.21
7600	-295.47
7700	-288.45
7800	-292.67
7900	-291.29
8000	-291.29
8100	-298.34
8200	-300.07
8300	-311.98
8400	-293.84
8500	-293.50
8600	-288.53
8700	-291.36
8800	-296.26
8900	-296.41
9000	-299.14
9100	-300.50
9200	-316.07
9300	-307.42
9400	-304.22
9500	-282.66
9600	-305.42
9700	-301.62
9800	-303.85
9900	-305.05
10000	-300.06
10100	-306.61
10200	-301.58
10300	-307.39
10400	-303.32
10500	-299.69
10600	-294.37
10700	-294.85
10800	-307.52
10900	-309.01
11000	-300.43
11100	-300.34

continued on next page

Table 8.4 — continued from previous page

Experiment	Average Reward
11200	-301.23
11300	-297.56
11400	-302.09
11500	-303.58
11600	-302.86
11700	-297.03
11800	-298.89
11900	-296.96
12000	-293.64
12100	-298.06
12200	-296.89
12300	-301.34
12400	-299.06
12500	-287.75
12600	-291.78
12700	-303.10
12800	-290.66
12900	-296.19
13000	-302.60
13100	-297.44
13200	-299.87
13300	-303.46
13400	-287.14
13500	-292.29
13600	-291.81
13700	-303.96
13800	-293.58
13900	-283.77
14000	-299.03
14100	-287.52
14200	-290.08
14300	-278.30
14400	-280.60
14500	-294.76
14600	-278.79
14700	-279.93
14800	-285.74
14900	-286.50
15000	-286.50

8.12 Appendix L: Validation of Decision Behaviour

8.12.1 Stopping frequency

Figure 8.5 shows that the average number of service stops gradually decreases over the episodes, suggesting that the DQN learns the overhead stopping costs that are incurred at every stop and therefore stops more efficiently to execute actions. In contrast, the refuelling amount per stop increases during training indicating a more efficient use of stops.

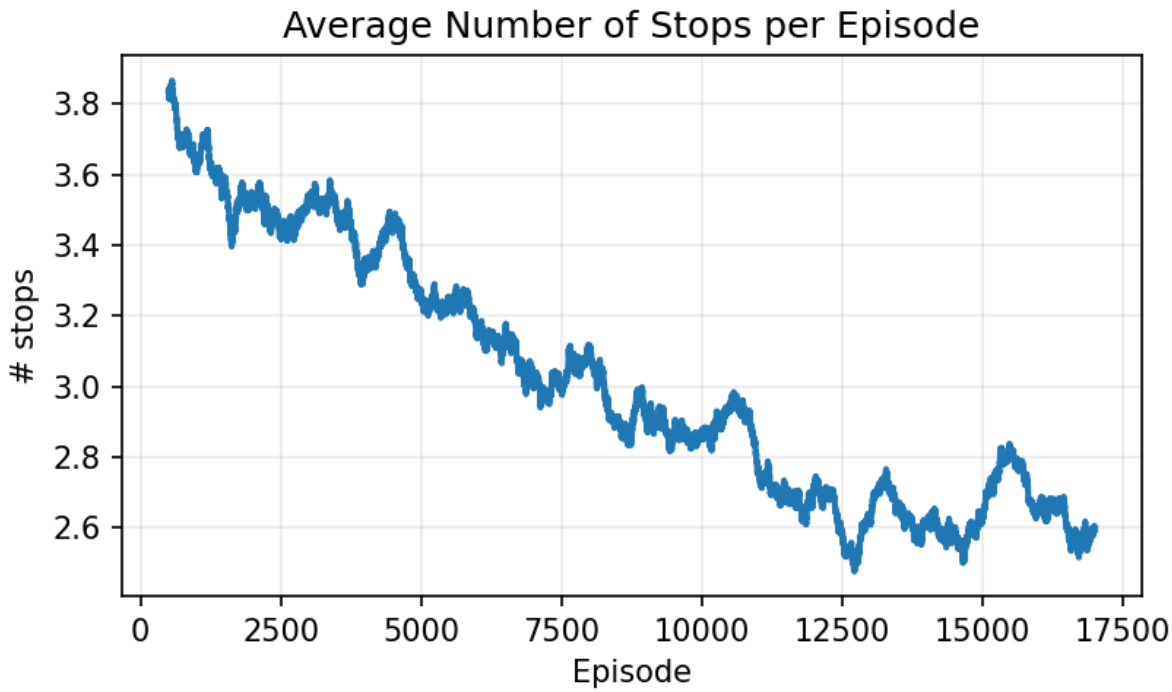


Figure 8.5: Reward per route current situation evaluation

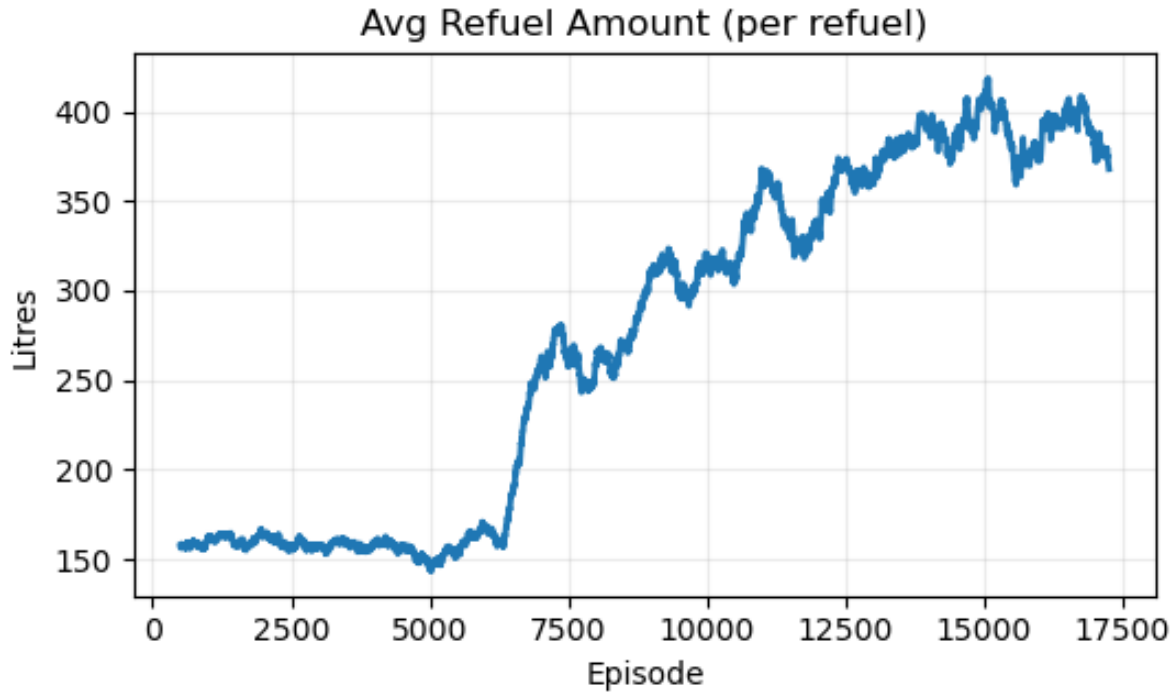
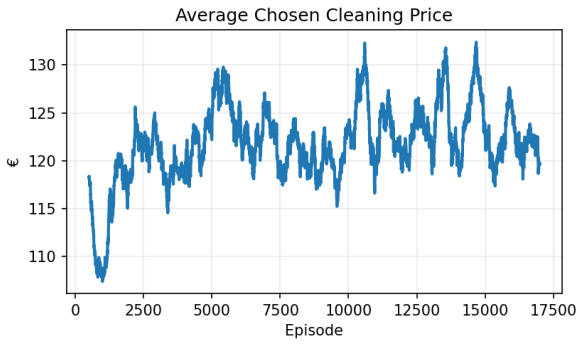


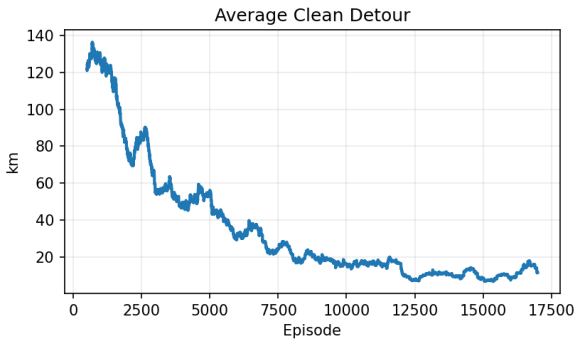
Figure 8.6: Average refuelling amount per stop

8.12.2 Fuel Price and detour balancing

Selecting a service station involves a trade-off between service costs (cleaning fee and price per litre) and detour. A rational policy should improve at least one of the two factors during training of the DQN: either paying less for services or driving a shorter distance to the stations. If both the average amount spent and the average detour increase over time, it indicates that the policy is becoming less efficient by spending more money to travel further. Figure 8.7 shows the moving average of the detour travelled to cleaning stations per route and the moving average of the cleaning prices paid per route. The DQN steadily reduces the detour to cleaning stations while keeping the cleaning fee roughly constant, implying that the learned policy saves costs by cutting kilometres.



(a) Average cleaning price

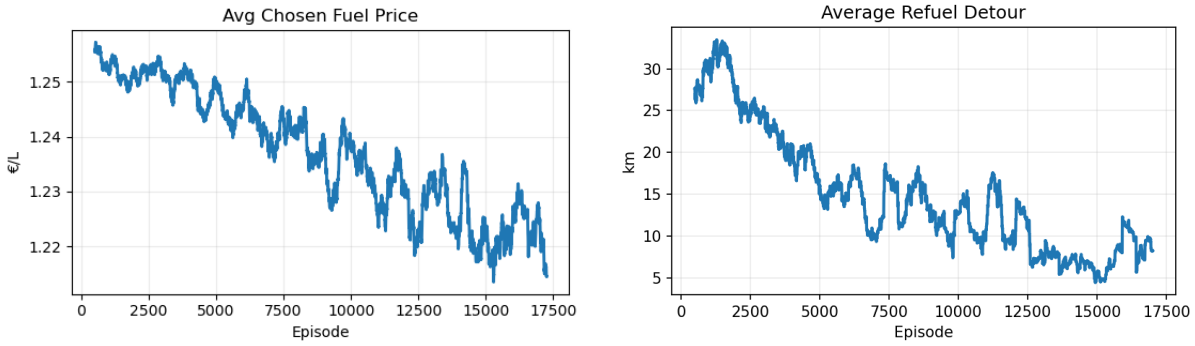


(b) Average detour to cleaning stations

Figure 8.7: Average refuel amount per refuelling stop

For refuelling, Figure 8.8 shows that both metrics decrease over the training episodes:

the DQN pays lower fuel prices on average and drivers take shorter detours to reach the fuel stations. This indicates a consistent improvement and shows that the DQN can detect price and detour differences and utilize them.



(a) Average fuel price per litre

(b) Average detour to refuelling stations

Figure 8.8: Average cleaning fee and cleaning detour

8.12.3 Bridge-Refuels

The MILP model is used to solve all routes in the test route set to identify all cost-efficient bridge refuels. Evaluation of all routes revealed that, in total, 11 cost-efficient bridge refuels were identified in the total test set of 397 routes. In contrast, the trained DQN model did not execute any bridging refuel actions in the same routes, which indicates that the bridge-refuel pattern has not been learned by the DQN model. This highlights a clear opportunity for improvement.

8.12.4 Action Combination

Assessing the DQN's ability to combine actions is non-trivial because its learned policy differs from the optimal routes coming from the MILP. Evaluation of the MILP decisions on the test route set revealed 147 combination actions. The DQN executed 256 combination actions. However, there was a 0% overlap between the two combination sets, meaning that none of the combination actions was the same for both sets. This result does not necessarily imply that the DQN cannot exploit combinations. It may realise savings through other station choices and timings under the DQN's policy that do not exactly match the MILP's selections. In other words, the absence of exact matches is a statement about policy alignment rather than the capability of the DQN model.

8.13 Appendix M: Limitation causes and solution directions

First, a limited number of routes still encounter issues when the agent reaches a state with no feasible actions available, resulting in failed advice. During operational implementation, advice is provided in advance, enabling the determination of whether the chosen action will result in a situation with no available actions by calculating the

remaining fuel and driving time after transitioning to the next state. During the operational phase, Nijhof-Wassink can then offer new advice for the next action using a different method.

A second limitation is the haversine distance calculation in the model. Because trucks never drive in a straight line from one location to the next, the actual driving time and fuel usage are different from those calculated by the DQN model. This can cause infeasibility of actions and lead to trucks getting stuck along the route with no fuel or remaining driving hours.

The Section on the validation of decision behaviour reveals a fourth limitation in this research. Currently, the agent cannot detect and exploit bridging refuelling actions, such as taking a small top-up at a nearby expensive station to reach a cheaper station downstream for a full refuel. Although it is not proven, a possible cause could be that the agent gets flooded with unimportant information from many stations in the state space, which dilutes the signal of the subtle cost trade-offs involved. This prevents the agent from learning a meaningful pattern regarding the bridging trade-off. A promising approach to explore is to replace the station features part in the state space with an aggregated summary of relevant information. An example of this could be to include only the three cheapest refuelling and cleaning stations, along with their detour and price, while still appending the global features such as fuel level and cleaning and resting requirement flags. This would only present the most critical information to the neural network, thereby reducing noise and decreasing the state dimensionality.

A second cause could be the scarcity of bridging trajectories in the training data. A training run of 10,000 episodes revealed that only 904 episodes contained a bridging pair (refuel expensive early to reach a cheaper station and fully refill there), and only 200 of those were actually cheaper than the full refuel at the expensive station. This results in only a small portion of the samples containing these trajectories. Additionally, the average main net gain of those 200 cheaper bridging trajectories was only €15.4. The combination of the rarity of these decision trajectories and the small advantage of the bridging action can make the learning signal too sparse and weak to overcome the sampling noise that exists in DQN. A potential solution could be to increase the frequency of bridging actions in the replay buffer. This can be achieved by making the bridge-refuel a distinct action alongside the full-refuel action and removing the partial refuel possibilities, which are not bridging actions. Given that the best policy is typically either a full refuel or a bridge refuel, collapsing the state space to these two actions will (i) increase the sampling probability of bridging actions, (ii) reduce the exploration of sub-optimal partial refuels, and (iii) strengthen the credit assignment to this binary decision structure. This scenario highlights the importance of understanding various real-world scenarios that can occur during the daily operations of Nijhof-Wassink. Not representing all of these scenarios in the replay buffer may cause the agent to become biased, resulting in unbalanced decision-making.

Although the DQN combines refuelling and cleaning actions, as well as refuelling and resting actions, it can only discover such combinations after the first action has been executed. This is because the state only encodes truck-to-station features and lacks station-to-station geometry. As a result, the agent cannot decide to postpone a refu-

elling action in advance to combine it later with a cleaning action. A possible solution would be to include information about the k-nearest stations, the k-cheapest stations around the concerned station, and their distances. However, this would significantly increase the state space, as every station would get an additional large list of features, making the state space even more extensive. Such an expansion could impact the computational efficiency. A more scalable alternative could be using a graph transformer as proposed by Shehzad et al. (2024). This approach can represent relationships between multiple stations without creating a huge per-station proximity vector.

Finally, the model does not account for any external uncertainties, such as traffic delays or daily fuel price fluctuations. These factors can significantly impact decision timing, action feasibility and realised costs. It was deliberately chosen not to include uncertainties in this model to isolate the model's ability to detect and utilise economical trade-offs in refuel timing, station choice and action combinations. Future work could incorporate stochastic travel times and fuel price dynamics to assess the model's robustness.

Bibliography

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

C. Archetti and Martin Savelsbergh. The trip scheduling problem. *Transportation Science*, 43:417–431, 11 2009. doi: 10.1287/trsc.1090.0278.

Okan Arslan, Barış Yıldız, and Oya Ekin Karaşan. Minimum cost path problem for plug-in hybrid electric vehicles. *Transportation Research Part E: Logistics and Transportation Review*, 80:123–141, 2015.

A. Bernhardt, Teresa Melo, Thomas Bousonville, and Herbert Kopfer. Truck driver scheduling with combined planning of rest periods, breaks and vehicle refueling. *Econstor*, (14), 2017. URL https://hdl.handle.net/10419/175088.

Ismail Capar, Michael Kuby, V Jorge Leon, and Yu-Jiun Tsai. An arc cover–path-cover formulation and strategic analysis of alternative-fuel station locations. *European Journal of Operational Research*, 227(1):142–151, 2013.

In-Chan Choi, Seong-In Kim, and Hak-Soo Kim. A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. *Computers Operations Research*, 30(5):773–786, 2003. ISSN 0305-0548. doi: https://doi.org/10.1016/S0305-0548(02)00050-3. URL https://www.sciencedirect.com/science/article/pii/S0305054802000503.

Carlos Cuartas and Jose Aguilar. Hybrid algorithm based on reinforcement learning for smart inventory management. *Journal of Intelligent Manufacturing*, 34, 08 2022. doi: 10.1007/s10845-022-01982-5.

Brett Daley, Martha White, and Marlos C. Machado. Averaging n-step returns reduces variance in reinforcement learning. 2024. URL https://arxiv.org/abs/2402.03903.

G. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6:80–91, 1959.

G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954. ISSN 00963984. URL http://www.jstor.org/stable/166695.

Nahid Parvez Farazi, Tanvir Ahamed, Limon Barua, and Bo Zou. Deep reinforcement learning and transportation research: A comprehensive review. 2020. URL https://arxiv.org/abs/2010.06187.

Matthias Feurer and Frank Hutter. *Hyperparameter Optimization*. The Springer Series on Challenges in Machine Learning, 2019. URL https://www.automl.org/wp-content/uploads/2019/05/AutoML_Book_Chapter1.pdf.

Ted Gifford, Tracy Opicka, Ashesh Sinha, Daniel Brink, Andy Gifford, and Robert Randall. Dispatch optimization in bulk tanker transport operations. *Interfaces*, 48: 403–421, 10 2018. doi: 10.1287/inte.2018.0956.

Fotios Gkatzoglou, Theophilos Papadimitriou, and Periklis Gogas. Fuel price networks in the eu. *Economies*, 12:102, 04 2024. doi: 10.3390/economies12050102.

Asvin Goel and Stefan Irnich. An Exact Method for Vehicle Routing and Truck Driver Scheduling Problems. *Transportation Science*, 51(2):737–754, 8 2016. doi: 10.1287/trsc.2016.0678. URL https://doi.org/10.1287/trsc.2016.0678.

Asvin Goel and A. Kok. Truck driver scheduling in the united states. *Transportation Science*, 46, 09 2010. doi: 10.2307/23263545.

LLC Gurobi Optimization. Gurobi optimizer reference manual. 2025. URL https://www.gurobi.com. Version 11.0.

Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015. URL https://arxiv.org/abs/1511.04143.

Hans Heerkens and Arnold van Winden. *Solving Managerial Problems Systematically*. Noordhoff Uitgevers, 2017. ISBN 9789001887957. Translated into English by Jan-Willem Tjooitink.

Shieu Hong Lin, Nate Gertsch, and Jennifer R. Russell. A linear-time algorithm for finding optimal vehicle refueling policies. *Operations Research Letters*, 35(3):290–296, 2007. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2006.05.003. URL https://www.sciencedirect.com/science/article/pii/S0167637706000666.

Seong Wook Hwang, Sang Jin Kweon, and Jose A Ventura. Locating alternative-fuel refueling stations on a multi-class vehicle transportation network. *European Journal of Operational Research*, 261(3):941–957, 2017.

J.C. Kessels. Deep reinforcement learning to support dynamic decision-making in a transport network amid travel- and handling time uncertainty. March 2024. URL http://essay.utwente.nl/98669/.

Samir Khuller, Azarakhsh Malekian, and Julián Mestre. To fill or not to fill: The gas station problem. *ACM Trans. Algorithms*, 7(3), July 2011. ISSN 1549-6325. doi: 10.1145/1978782.1978791. URL https://doi.org/10.1145/1978782.1978791.

Dukwon Kim and Panos M. Pardalos. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Operations Research Letters*, 24(4):195–203, 5 1999. doi: 10.1016/s0167-6377(99)00004-8. URL https://doi.org/10.1016/s0167-6377(99)00004-8.

M.N. Kruit. Smart refueling decisions using a reinforcement learning approach : a case study at nijhof-wassink. March 2024. URL http://essay.utwente.nl/98539/.

Rahma Lahyani, Leandro C. Coelho, Mahdi Khemakhem, Gilbert Laporte, and Frédéric Semet. A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia. *Omega*, 51:1–10, 9 2014. doi: 10.1016/j.omega.2014.08.007. URL https://doi.org/10.1016/j.omega.2014.08.007.

Hoong Chuin Lau, Melvyn Sim, and Kwong Meng Teo. Vehicle routing problem with time windows and a limited number of vehicles. *European Journal of Operational Research*, 148(3):559–569, 2003. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(02)00363-6. URL https://www.sciencedirect.com/science/article/pii/S0377221702003636.

Shieu-Hong Lin. Finding optimal refueling policies in transportation networks, 06 2008.

Shieu-Hong Lin. Multi-objective vehicle refueling planning using mixed integer programming. pages 677–681, 2014. doi: 10.1109/IEEM.2014.7058724.

Zichuan Lin, Tianqi Zhao, Guangwen Yang, and Lintao Zhang. Episodic memory deep q-networks, 2018. URL https://arxiv.org/abs/1805.07603.

Manuel López-Ibáñez, Christian Blum, Jeffrey W. Ohlmann, and Barrett W. Thomas. The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization. *Applied Soft Computing*, 13(9):3806–3815, 2013. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2013.05.009. URL https://www.sciencedirect.com/science/article/pii/S1568494613001658.

Teresa Melo, Alexandra Bernhardt, Thomas Bousonville, and Herbert Kopfer. Scheduling of driver activities with multiple soft time windows considering european regulations on rest periods and breaks. 11 2016. doi: 10.13140/RG.2.2.27918.77122.

TURISMO Y COMERCIO MINISTERIO DE INDUSTRIA, SECRETARIA GENERAL DE TRANSPORTES, DIRECCION GENERAL DE TRANSPORTES POR CARRETERA, and Enrique Jiménez. *Guía para la gestión del combustible en las flotas de transporte por carretera*. 1 2006.

Alejandro Montoya. Electric vehicle routing problems: models and solution approaches. 2016.

Mohammad S. Moshtagh, Yun Zhou, and Manish Verma. Dynamic inventory and pricing control of a perishable product with multiple shelf life phases. *Transportation Research Part E: Logistics and Transportation Review*, 195, 2025. doi: 10.1016/j.tre.2025.103960. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85214945129&doi=10.1016%2fj.tre.2025.103960&partnerID=40&md5=d9ad778cb49f0608082606ec69473a0d. Cited by: 1.

Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1): 60–100, 1991. doi: 10.1137/1033004.

Kleitos Papadopoulos and Demetres Christofides. A fast algorithm for the gas station problem. *Information Processing Letters*, 131:55–59, 12 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. 2019. URL https://ar5iv.org/html/1912.01703.

Warren Powell. Approximate dynamic programming: Solving the curses of dimensionality. 08 2011. doi: 10.1002/9781118029176.

Warren B. Powell. Perspectives of approximate dynamic programming. *Annals of Operations Research*, 241(1-2):319–356, 2 2012. doi: 10.1007/s10479-012-1077-6. URL https://doi.org/10.1007/s10479-012-1077-6.

Martin L. Puterman. Chapter 8 markov decision processes. 2:331–434, 1990. ISSN 0927-0507. doi: https://doi.org/10.1016/S0927-0507(05)80172-0. URL https://www.sciencedirect.com/science/article/pii/S0927050705801720.

John Rust. Dynamic programming. *The new Palgrave dictionary of economics*, 1:8, 2008.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. 2017a. URL https://arxiv.org/abs/1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017b. URL https://arxiv.org/abs/1707.06347.

Arne Schulz and Yoshinori Suzuki. Bounds and valid inequalities for the fixed-route vehicle-refueling problem. *Computers Industrial Engineering*, 170:108340, 2022. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2022.108340. URL https://www.sciencedirect.com/science/article/pii/S0360835222003916.

Arne Schulz and Yoshinori Suzuki. An efficient heuristic for the fixed-route vehicle-refueling problem. *Transportation Research Part E: Logistics and Transportation Review*, 169:102963, 2023. ISSN 1366-5545. doi: https://doi.org/10.1016/j.tre.2022.102963. URL https://www.sciencedirect.com/science/article/pii/S1366554522003404.

Ahsan Shehzad, Feng Xia, Shagufta Abid, Ciyuan Peng, Shuo Yu, Dongyu Zhang, and Karin Verspoor. Graph transformers: A survey, 2024. URL https://arxiv.org/abs/2407.09777.

Stefan Studer, Thanh Binh Bui, Christian Drescher, Alexander Hanuschkin, Ludwig Winkler, Steven Peters, and Klaus-Robert Müller. Towards crisp-ml(q): A machine learning process model with quality assurance methodology. *Machine Learning and Knowledge Extraction*, 3(2):392–413, 2021. ISSN 2504-4990. doi: 10.3390/make3020020. URL https://www.mdpi.com/2504-4990/3/2/20.

Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. 2014. URL https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

Yoshinori Suzuki. A decision support system of dynamic vehicle refueling. *Decision Support Systems*, 46(2):522–531, 2009. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2008.09.005. URL https://www.sciencedirect.com/science/article/pii/S016792360800170X.

Yoshinori Suzuki. A decision support system of vehicle routing and refueling for motor carriers with time-sensitive demands. *Decision Support Systems*, 54(1):758–767, 2012. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2012.09.004. URL https://www.sciencedirect.com/science/article/pii/S0167923612002394.

Yoshinori Suzuki. A variable-reduction technique for the fixed-route vehicle-refueling problem. *Computers Industrial Engineering*, 67:204–215, 2014a. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2013.11.007. URL https://www.sciencedirect.com/science/article/pii/S0360835213003707.

Yoshinori Suzuki. A variable-reduction technique for the fixed-route vehicle-refueling problem. *Computers Industrial Engineering*, 67:204–215, 2014b. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2013.11.007. URL https://www.sciencedirect.com/science/article/pii/S0360835213003707.

Yoshinori Suzuki and Bo Lan. Cutting fuel consumption of truckload carriers by using new enhanced refueling policies. *International Journal of Production Economics*, 202:69–80, 2018. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2018.05.007. URL https://www.sciencedirect.com/science/article/pii/S0925527318302019.

Yoshinori Suzuki, Frank Montabon, and Shih-Hao Lu. Dss of vehicle refueling: A new enhanced approach with fuel weight considerations. *Decision Support Systems*, 68:15–25, 2014. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2014.10.005. URL https://www.sciencedirect.com/science/article/pii/S0167923614002504.

Fuxiao Tan, Pengfei Yan, and Xinping Guan. Deep Reinforcement Learning: From Q-Learning to Deep Q-Learning. pages 475–483, 1 2017. doi: 10.1007/978-3-319-70093-9\{_}50. URL https://doi.org/10.1007/978-3-319-70093-9_50.

Matus Telgarsky. Benefits of depth in neural networks. 2016. URL https://arxiv.org/abs/1602.04485.

UN Trade and Development. Global trade hits record high of 28.5 trillion in 2021, but likely to be subdued in 2022, 2 2022. URL https://unctad.org/news/global-trade-hits-record-high-285-trillion-2021-likely-be-subdued-2022?utm_source=chatgpt.com.

Wouter van Heeswijk, Martijn Mes, and Marco Schutten. *An approximate dynamic programming approach to urban freight distribution with batch arrivals*. Number 474 in BETA working paper. BETA Research School for Operations Management and Logistics, 2015.

Fei Wang, Honghai ZHANG, Sen DU, Mingzhuang HUA, and Gang ZHONG. C-sppo: A deep reinforcement learning framework for large-scale dynamic logistics uav routing problem. *Chinese Journal of Aeronautics*, 38(5):103229, 2025. ISSN 1000-9361. doi: https://doi.org/10.1016/j.cja.2024.09.005. URL https://www.sciencedirect.com/science/article/pii/S1000936124003662.

Hang Xu, Zhi-Long Chen, Srinivas Rajagopal, Sundar Arunapuram, and Manugistics Inc. Solving a practical pickup and delivery problem. *Transportation Science*, 37, 03 2001. doi: 10.1287/trsc.37.3.347.16044.

Yue Zhu, Mingyu Cai, Chris Schwarz, Junchao Li, and Shaoping Xiao. Intelligent traffic light via policy-based deep reinforcement learning. *International Journal of Intelligent Transportation Systems Research*, 20, 08 2022. doi: 10.1007/s13177-022-00321-5.

Zhengyu Zhu, Jutao Guo, Youlong Lyu, Liling Zuo, and Jie Zhang. Deep reinforcement learning method for flexible job shop scheduling; []. *Zhongguo Jixie Gongcheng/China Mechanical Engineering*, 35(11):2007–2014 and 2034, 2024. doi: 10.3969/j.issn.1004-132X.2024.11.012. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85214693543&doi=10.3969%2fj.issn.1004-132X.2024.11.012&partnerID=40&md5=a03849307c1b2845382321f79c8d1268. Cited by: 0.