Daan van Gisbergen

Waiting and rebalancing strategies in an urban waterborne transport environment

Thesis

Waiting and rebalancing strategies in an urban waterborne transport environment

Daan van Gisbergen

In partial fulfillment of the requirements for the degree of
Master of Science
in Mechanical Engineering

At the Department Maritime and Transport Technology of the Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology

Studentnumber:	4661044
MSc track:	Multi-Machine Engineering
Report number	2023.MME.8880
Supervisors:	Dr. Bilge Atasoy
	Dr. Breno Alves Beirigo
Date:	2023

Abstract

This thesis explores waiting and rebalancing strategies in urban waterborne transport, intending to improve the efficiency and operational performance of water transportation systems. The study is based on a review of the literature on waterborne transport and waiting and rebalancing strategies. This is followed by a case study where the found strategies will be implemented for the water taxi case in Rotterdam.

The literature review provides an overview of the current state of research on waterborne transport, highlighting its potential as a sustainable and efficient mode of urban mobility. The review also identifies waiting and rebalancing strategies as key factors for improving the performance of water transportation systems. Waiting strategies involve managing the distribution of slack time in the route of the vehicle, while rebalancing strategies aim to optimize the use of resources by redistributing vessels across the service area.

The implementation of waiting and rebalancing strategies on the water taxi system in Rotterdam provides a practical example of how waiting and rebalancing strategies can be implemented in a real-world context. This study analyzes the performance of the water taxi system using the actual data provided by Flying Fish. The analysis results in a clear overview of the requirements of the experimental setup that is needed to evaluate the strategies in this specific waterborne transport environment.

A detailed description of the simulation model is given. The simulation model is based on a mobility-on-demand simulation model. Multiple modifications needed to be made in order to fit the model to the water taxi scenario. After a working simulation model has been created, a benchmark is created to make a comparison between the strategies.

The experiments reveal that waiting and rebalancing strategies can be effective in improving the performance of water transportation systems, but their implementation requires careful planning and coordination. The study identifies several strategies that improve the systems performance in terms of one factor, but deteriorate other performance factors. The trade-off between pick-up delay on one side and the vehicle usage and traveled distance on the other side is a recurring theme when looking at the results of the strategies.

Overall, the findings of this research contribute to the growing body of knowledge on innovative solutions for urban mobility and provide a basis for further exploration and implementation of waiting and rebalancing strategies in waterborne transport systems. The study highlights the potential of these strategies to improve the efficiency of urban transport and underscores the importance of considering waterborne transport as a viable and valuable mode of urban mobility.

Preface

Dear reader,

In this thesis, I have done my best to provide a clear and extensive description of my research on waiting and rebalancing strategies in an urban waterborne transport environment, like the water taxi case in Rotterdam. I started this project in January 2023 with the literature research, which led to a total project duration of almost a year. This research is the last milestone in my student career and although I will look back at this project as a more pleasant experience than expected, I am also happy to move on and see in what way I can apply the gained knowledge from this project.

I would like to thank my supervisors, Bilge Atasoy and Breno Alves Beirigo for their guidance during this project over the past year. The balance in guiding me but also letting me figure things out myself has worked great and was efficient, just how I like it. Also, I would like to thank Tim Visser from Flying Fish for lending his expertise in managing water taxis in Rotterdam and giving his insights where it was needed.

This work focuses on the difficulties that emerge when trying to converge a simulation model to a model that exactly fits the real-life scenario, how to implement strategies on it, and review its results. I hope that readers find this thesis both informative and inspiring, and I invite you to explore its contents.

Daan van Gisbergen

Rotterdam

December 4, 2023

Contents

Glossary

CSV Comma Separated Values, a text file format that uses commas to separate values. 15

DF The Drive First waiting strategy. 7

ES Evolution strategy, a tuning algorithm that uses evolution to iteratively create better parameter sets. 10

GeoJSON A dictionary file specialized in storing geographical data. 24

H3 Uber's Hexagonal Hierarchical Spatial Index, used for creating the grid network. 24

ILP Integer Linear Programming, a variant of LP in which some or all of the variables are restricted to be integers. 31

LP Linear programming, an optimization method to achieve the best outcome in a mathematical model whose requirements are represented by linear relationships. 11

LSTM Long Short-Term Memory, a specific type of neural network used for predicting future requests. 12

MoD Mobility on Demand, a type of mobility service that can be completely tailored to the user and only serves dynamic requests. 35

MPC Model predictive control, a model where the current state is used to predict the next state of the model. 12

PDP Package Delivery Problem, a variant of a VRP, where vehicles are delivering packages. 7

PDPTW Package Delivery Problem with Time Windows, a variant of the PDP where the delivery of packages is also bound by a time constraint.. 8

RTV graph Request - Trip - Vehicle graph, a graph structure where the different combinations of RR and VR pairs are stored. 31

SQL Structured Query Language, a domain-specific language used in programming and designed for managing data held in a relational database management system. 15

VRP Vehicle Routing Problem, a general problem where the route of one or multiple vehicles is optimized. 7

WF The Wait First waiting strategy. 8

List of Figures

List of Tables

1. Introduction

Urban waterborne mobility, the movement of people and goods through waterways in cities, has been gaining attention as a sustainable transportation alternative (Tanko and Burke, 2017). With the increasing population density in cities and the need for efficient transportation options, the use of waterways is becoming more important. Waterborne mobility offers numerous advantages over traditional modes of transportation, including reduced traffic congestion and faster travel times. In recent years, many cities have implemented initiatives to develop and promote waterborne mobility (Cheemakurthy, 2017). This has resulted in the establishment of new waterway transport systems, such as ferries and water taxis, as well as the revitalization of existing waterways. These initiatives aim to reduce the negative impacts of transportation on the environment.

One of those initiatives is a project done by Flying Fish (*Flying Fish* 2023) in Rotterdam. They have modernized the water taxi system in multiple ways. Flying Fish has developed a water taxi Operations System that handles all position and motion data, as well as incoming requests. The requests can be made on their in-house developed app that provides the user with real-time booking information based on the network state. The system successfully services thousands of people on busy days. There are, however, always improvements to be made. In almost every transport system, passengers experience waiting times. These are the result of inefficiencies in the reaction of the system due to a supply and demand imbalance. For transport companies, the challenge lies in improving this reaction while keeping the added costs low. Various methods have been developed to improve the system performance while the equipment used remains unchanged.

Methods to improve the efficiency of the system are, for instance, rebalancing strategies or waiting strategies. A rebalancing strategy will try to optimally distribute idle vehicles over the service area. This results in a distribution over the network where new requests are serviced quicker because the requested vehicle was already in the neighborhood. On the other hand, waiting strategies aim to distribute the slack time in the route as efficiently as possible. The slack time is time that occurs when one customer is served, but the next request follows not directly after. By waiting at strategic moments, the vehicle may be able to serve more requests in the same period. For both of these strategies, multiple variants will be analyzed and compared with each other.

In order to review the results of these strategies in the Rotterdam water taxi scenario, a simulation has to be performed. The simulation should include the spatial data of the Rotterdam water taxi service area. It should be able to handle and serve incoming requests as close to reality as possible. The research done in this thesis includes the creation of this simulation and after that, test selected waiting and rebalancing strategies by simulation. To encompass this goal, the following research question has been formulated.

How can the operational performance of a water taxi service be improved while maintaining a positive customer experience?

This is a broad research question and is difficult to answer on its own. In what ways could the operation be improved and what exactly can be defined as a positive customer experience? To aid in answering this research question, several subquestions have been formulated as well.

- RQ1: What are the available waiting and rebalancing strategies that could be applied to an urban waterborne transport environment?
- RQ2: What are the system characteristics and requirements of the Rotterdam water taxi system?
- RQ3: In what way can waiting strategies reduce the required fleet size and total traveled distance?
- RQ4: In what way can rebalancing strategies decrease customer wait time and reduce the required fleet size?
- RQ5: How can these methods be integrated successfully?
- RQ6: What is the benefit of proposed methods under different scenarios of interest?

The subquestions break the problem down into easier portions. The characteristics and requirements of the water taxi system in Rotterdam can be largely determined by doing data analysis. However, for a data analysis, data has to be provided in the first place. Fortunately Flying Fish was able to provide the required data for this research. The different possibilities for waiting and rebalancing strategies can be analyzed in a study of the current literature about the strategies. The integration of these strategies can be done by simulating the problem and tuning the strategies. Also, how these methods influence the performance of the system can be evaluated by analyzing the results of the simulation.

The report is structured as follows. In Chapter 2, the existing literature will be analyzed. In Section 2.1 a brief review will be performed on literature about urban waterborne transport technologies. Also, emerging technologies like autonomous transport and ride-sharing will be reviewed in this waterborne context. After that, different waiting strategies are discussed in Section 2.2, and their advantages and disadvantages are outlined. The same procedure is followed for rebalancing strategies in Section 2.3. This chapter will answer RQ1. When the literature is studied, the next step is to perform data analysis in Chapter 3. In Section 3.1, the original data structure is shown and the way this data was handled is explained. When data is handled and processed correctly, the relevant information can be extracted. In the following 3 sections, statistics about locations, demand, and trips are shared and reviewed. In Section 3.5 a custom network is created to structure the movement of vehicles and paths in the river. This is an important step because other spatial structures like roads do not exist on water. With the network in place, important aspects like distance of paths and duration of trips can be derived, answering RQ2. Chapter 4 entails the construction of the simulation model. In Section 4.1 the most relevant parts of the received model from B. Alves Beirigo are explained. The parts consist of request generation and handling, matching, fleet and requests update, and rebalancing. The modifications made to the simulation to make it fit the Rotterdam scenario are shown in Section 4.2. After that, modifications made in request handling, like adding reveal time and handling mixed requests are explained in Section 4.3. In Section 4.4, the modifications made to the matching method are explained. This includes the addition of future assignments and tuning of ILP objectives. A clear benchmark needed for further testing is created in Section 4.5. With this work done, the tests on strategies can be performed. First up are the waiting strategies in Chapter 5, where RQ3 is answered. Here, the implementation of a waiting strategy into the model is explained in Section 5.1. This is followed by the strategies themself with Wait First in Section 5.2 and the intensity waiting strategy in Section 5.3. The chapter is concluded with the results summarized in Section 5.4. After that, the rebalancing strategies will be implemented and tested in Chapter 6. Starting with rebalancing to location, and after that rebalancing based on delay. Rebalancing strategies that include historical requests in their decision process are the forecast rebalancing strategy in Section 6.3 and the intensity rebalancing strategy in Section 6.4. The last one is a new rebalancing strategy that is introduced in this research. All the results are analyzed together in Section 6.5, where RQ4 is answered. In the discussion in Chapter 7, different aspects of the research are discussed. The results of waiting and rebalancing strategies are compared with each other and combined to see if this improves results in Section 7.1. This section will also answer RQ5. It is followed by a sensitivity analysis in Section 7.2. Here, the robustness of the developed model is evaluated by performing some sensitivity tests. After that, various assumptions and simplifications are outlined and justified. Finally, this thesis concludes in Chapter 8. Here, the research questions are answered and the contributions as well as future work in this research field are mentioned.

2. Literature Review

The cost of road congestion in Europe is estimated to be over €110 billion a year (Chang, Lee, and Choi, 2017). As a result, cities are increasingly looking for new ways to expand their public transport offering (Cheemakurthy, 2017). For water-rich cities, the development of transport networks on their urban waterways is being considered. By attracting more passengers to these alternative ways of transport, the number of people using the road network during peak hours can be decreased considerably. The alternative service can be made attractive by offering faster or more reliable transit times, a better experience, or lower fares.

2.1. Urban waterborne transport

Traditional modes of urban transportation, such as cars and buses, often cause and are affected by issues such as road congestion and air pollution. These issues can lead to longer commute times, decreased quality of life, and negative impacts on the environment. However, waterborne transport can offer a solution to these problems. Outcomes may vary depending on the scenario and the implementation of this additional mode of transport.

Ferries have always been the most used type of urban waterborne transportation (Cheemakurthy, 2017),(Thompson, Burroughs, and Smythe, 2006). The advantage of ferries is that the large capacity of a ferry allows the transportation of large groups of people in an efficient way. Furthermore, the scheduled operation of ferries makes it easy to integrate with other public transport systems, such as buses (Soltani et al., 2015). They do however not provide a customizable service. A service such as taxis provide on land. This type of transport completely adapts to the instructions of the customer. Water taxis provide the same type of service, but in this case on the water. van Gisbergen (2023) analyses multiple cases where a water taxi service would provide a useful service and could improve the urban transportation system as a whole.

One scenario where waterborne transport could be advantageous is in a city with a large river running through it. Rotterdam for example, see Figure 2.1. In this case, a route is formed between the city park, which is located near the museum square of Rotterdam, and the SS Rotterdam, a historic ship with restaurants and a hotel inside. Due to the geological features of the old harbor and the busy roads leading up to the Erasmus Bridge, the ride by car would take 16 minutes. A water taxi would only need to cross the River Maas which it would do in 3 and a half minutes, boarding and disembarking included.

By creating a faster alternative route, waterborne transport can reduce congestion on roads and highways, which can lead to faster commute times and increased productivity. Additionally, waterborne transport can reduce air pollution and greenhouse gas emissions, which can have positive impacts on the environment and public health. This is already demonstrated by Watertaxi-Rotterdam (2023) who have electric and hydrogen-powered vessels in their fleet. Furthermore, waterborne transport can increase mobility and accessibility for people who live or work near waterways. A large portion of customers from the water taxis of Rotterdam are people who work in the harbor area and would not have been able to reach their work location as easily without water taxis.

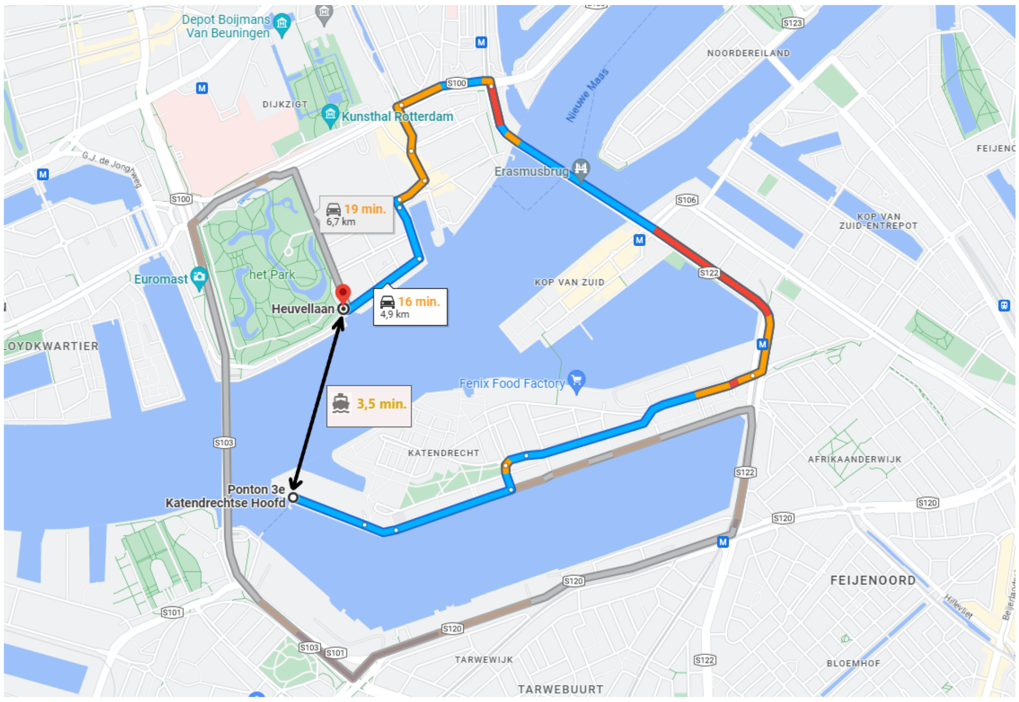


Figure 2.1.: The difference in transport types in Rotterdam. Where a route is formed between the city park and the historical ship hotel SS Rotterdam.

Autonomy With emerging new technologies like autonomous driving, questions arise how this technology could be implemented into different modes of transport. Milakis, Arem, and Wee (2017) and Duan et al. (2020) established in their research that autonomous driving is beneficial for traffic in general. A benefit for taxi companies is that eliminating the taxi driver saves on labor costs, making way for cheaper transport. Moreover, it eliminates the constraint of working hours of the driver for the taxi company, resulting in easier planning and logistics.

On top of the advantages of eliminating human labor costs, autonomous taxis offer significant advantages in vehicle rebalancing and dynamic rerouting (Kretschmann, Burmeister, and Jahn, 2017). Taxi drivers generally follow the principle of maximizing self-profit and do not cooperate in a centralized way. Rebalancing does not work if that principle is followed by all vehicles. The most potent area for future requests is what most taxis will be converging to, which does not follow an efficient distribution. Autonomy removes the principle of maximizing self-profit and ensures an operation that can be controlled in a centralized form. Apart from the reduced operational costs, there are extra benefits to the application of autonomy in watertaxis, as presented by Gu and Wallace, 2021. They state that there are 3 additional advantages for autonomous watertaxis: unprecedented terminal locations, multi-terminal locations, and increased routing flexibility.

Ride-sharing Ride-sharing has clear advantages for the community and environment. Effectively, the number of cars needed to bring the same number of people to their destination is significantly reduced. This results in less cars on the roads, meaning less traffic congestion and air pollution (Saranow, 2006). However, driving your own car has the great advantage of immediate access to door-to-door transportation (Agatz et al., 2012). The costs of losing this feature have to be compensated with the benefits of ride-sharing in a mobility on-demand transportation system. There are multiple factors that play a role while deciding to share a ride or not (Lo and

Morseman, 2018). The benefits consist of a considerably lower fare for the ride and the fact that the user is contributing to reducing traffic. The role of price, however, by far is the most important benefit for the user. The costs on the other hand are: possibly having to walk to a pick-up point, longer travel time, and availability.

Ride-sharing on water has slightly different implications than ride-sharing on land. First, because the number of pick-up or drop-off locations is discretized into a small selection of docks. This results in locations that are mostly far apart from each other. Not having the flexibility to drop someone off at the side of the road can result in difficulties for ride-sharing (Stiglic et al., 2016). However, the decrease in combinations also increases the chances of finding perfect matches, where the start and end location and time are identical (Kooti et al., 2017), (Lo and Morseman, 2018). And last, the capacities of watertaxis are generally larger than conventional taxis. This means that there is more potential in ride-sharing because there are more potential passengers over which the fare can be shared.

The costs and benefits of waterborne transportation over conventional on-demand transportation are summarized in Table 2.1.

Table 2.1.: Summary of the important aspects between taxis over land or over water

Category	Factor	Conventional taxis	Watertaxis
General	Service area	Bounded by road network	Bounded by water
	Capacity	Up to 8 passengers (van)	Up to 20 passengers (Rotterdam)
	Pick up and drop off locations	Anywhere near the road	Dedicated docking spots
	Travel continuity	Heavily dependent on traffic flow	Independent of traffic
Autonomy	Environment	Highly structured road environment	Unstructured waterway
	External forces	Minimal disturbances	Significant disturbances
	Other users in network	High density traffic	Low density traffic
Ride-sharing	Multiple locations in one trip	Minimal delay if route is optimized	Always delay caused by distance between locations and slow docking
	Perfect match	Hard to accomplish	Easier to accomplish
	Potential	Limited due to low capacity	High due to high capacity

2.2. Waiting strategies

Techniques that can be used to improve serviceability and reduce the required fleet size in vehicle routing problems (VRP) or pickup and delivery problems (PDP) are waiting strategies. In most dynamic transportation systems, vehicles face slack time during their service. This occurs when a request is completed and the vehicle does not immediately have to service another request. The slack time during the service day is something that in general would be minimized, but in most cases, some slack time is also inevitable. In a dynamic environment, waiting during some moments in time might be more beneficial than others. This has to do with possible incoming requests that require rescheduling that could lead to a better solution. Waiting strategies aim to optimally distribute this time throughout the service. In this section, some waiting strategies will be elaborated.

2.2.1. Drive First

The Drive First (DF) strategy requires a vehicle to always drive if possible. In other words, to leave a location at the earliest departure time. Let i be a location with a customer with time window $[a_i, b_i]$. For a vehicle traveling towards location i, let $\underline{A}_i$ be the earliest possible arrival time and $\overline{A}_i$ be the latest possible arrival time. When DF is applied, the scheduled arrival time A_i will be equal to $\underline{A}_i$. The vehicle does wait at the arrived location i if the location is not open for service yet. The waiting time at location i before the service is equal to $max\{0, a_i - \underline{A}_i\}$. The waiting time after the service is always zero. The DF strategy is commonly used when transporting letters or parcels, for instance, see Benyahia and Potvin (1998), Gendreau et al. (2006). This is because these situations are static vehicle routing problems. Drive First is the only appropriate strategy for a static vehicle routing problem because, without dynamism, the earliest possible arrival is always preferred. Also in a dynamic environment, this strategy can be preferred. This is because of the intuitive point of view that being somewhere too early is better than being too late. A DF strategy creates in this way a larger margin of error, which is always the 'safest' option and requires no extra computation. A visualization of the strategy is displayed in Figure 2.2.

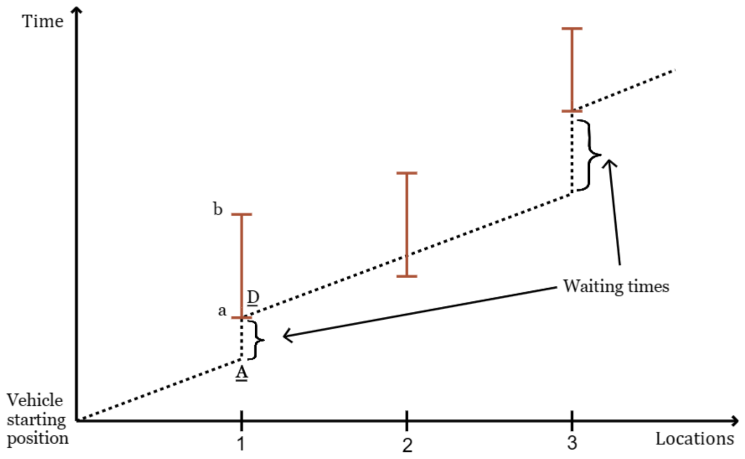


Figure 2.2.: Drive First strategy visualized, where 3 locations are visited by the vehicle. The time windows of the locations are the red bars and the progression of the vehicle is represented by the dotted line. Illustration based on Mitrović-Minić and Laporte (2004)

2.2.2. Wait First

Opposite to the DF strategy, Wait First (WF) requires a vehicle to always wait at the current location as long as possible. It means that the vehicle leaves the location at the latest possible departure time. For the same customer location i, with a time window $[a_i, b_i]$ and a vehicle with earliest possible arrival time $\underline{A}_i$ and latest possible arrival time $\overline{A}_i$, the arrival time A_i will always be the latest possible $\overline{A}_i$. The location is always serviced immediately and the vehicle is allowed to wait at the current location. For the WF strategy, the earliest possible departure time $\underline{D}_i$ from location i and the latest possible departure time $\overline{D}_i$ is introduced as well. The waiting time at location i is then given by $max\{0, \overline{D}_i - b_i\}$. WF has the advantage over DF in that it has the potential to build shorter routes. By waiting longer it can incorporate changes in requests and has a longer period to optimize routing. The disadvantage is that WF requires more vehicles to service the same set of locations. This has to do with the fact that WF concentrates a lot of the waiting time in the beginning, leaving too little time in the end. It is therefore unlikely that the vehicle is able to service requests that appear after it leaves its starting position. This requires more vehicles to complete the service. It is illustrated in Figure 2.3 that most of the waiting time is accumulated at the beginning of the service period.

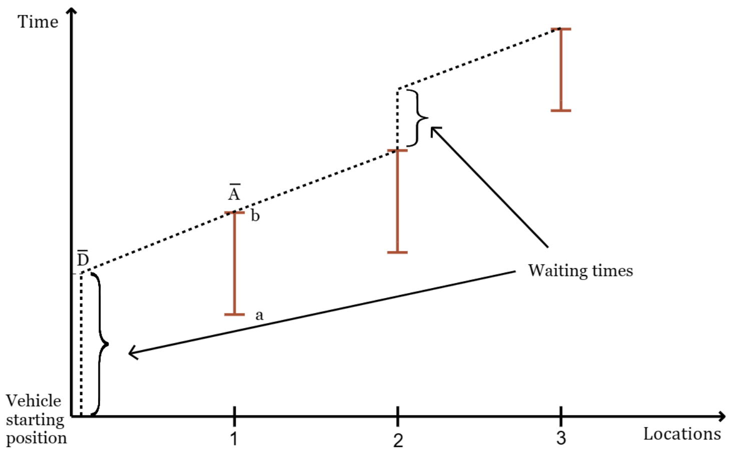


Figure 2.3.: Wait First strategy visualized, where 3 locations are visited by the vehicle. The time windows of the locations are the red bars and the progression of the vehicle is represented by the dotted line. Illustration based on Mitrović-Minić and Laporte (2004)

2.2.3. Intensity waiting strategy

The intensity strategy was proposed by Vonolfen and Affenzeller (2016). Their strategy, applied on a PDPTW, moves away from the dependence on stochastic knowledge because of the fact that stochastic knowledge may require intensive pre-processing steps to result in high-quality predictions (Ferrucci, Bock, and Gendreau, 2013). Their strategy introduces a new variable on which decisions can be based: the intensity measure, which is why this strategy will be called the *intensity* strategy. This measure aims at removing the restrictions that a stochastic model imposes.

The intensity measure is based on two features, the transition time and historical requests. In the study by Vonolfen and Affenzeller (2016), a set of training instances is used as historical data. A service request is defined as $s \in S$ where S is the total set of service requests. The request s has a reveal time r_s, a pick-up location l_s^p, and a drop-off location l_s^d. The transition time is the total amount of time that is between the end of service on a location, and the beginning of service on the next location. Potential waiting time due to time windows is included in this transition time. The transition time is given by the following formula.

$$TransTime(l_1, l_2, t) = dist(l_1, l_2) + max\{0, o_{l_2} - (t + dist(l_1, l_2))\} \tag{2.1}$$

Where $dist(l_1, l_2)$ is the distance between the two locations. The current time is denoted as t and the opening time of the time window of location l_2 is denoted as o_{l_2}. Essentially, Equation 2.1 states that the transition time is equal to the distance between the two locations with the addition that if there is waiting time, this is added to the transition time. The derivation for the intensity measure is displayed in Equation 2.2.

$$Intensity(l, t) = 1 - \frac{\sum_{\{s \in S : r_s > t\}} \frac{TransTime(l, l_s^p, t)}{h}}{|\{s \in S : r_s > t\}|} \tag{2.2}$$

The intensity is defined by the average transition time for unrevealed requests in the historical set S. The transition time is normalized by the planning horizon h. The intensity of a location is high if the averaged transition time to potential future requests l_s^p is low. This results in a high intensity if there are nearby historical requests, and a low intensity corresponds to historical requests that are far away.

The *intensity* strategy will make the decision on whether to wait at a given location R_i in a given route R or to move on to the next location R_{i+1} at time t_i based on three factors:

- Transition time
- Intensity
- Slack time

These 3 factors are combined into a single value v to make the decision if the vehicle should wait at the given location R_i. The calculation is displayed in Equation 2.3.

$$\begin{aligned} v(R_i, t_i) = {} & \alpha' \cdot \frac{TransitionTime(R_i, R_{i+1}, t_i)}{(h - t_i)} \\ & + \beta' \cdot \frac{Intensity(R_i, t_i)}{Intensity(R_i, t_i) + Intensity(R_{i+1}, t_{i+1})} \\ & + \gamma' \cdot \frac{Slack(R_i, t_i)}{(h - t_i)} \end{aligned} \tag{2.3}$$

The Slack function returns the slack time, which is the maximum time the vehicle is permitted to wait without violating time windows. The values for transition time and slack time are normalized by the time remaining in the planning horizon. The intensity is divided by the sum of the current location's intensity and the intensity of the next location. This results in all factors having a range of [0,1]. Each of the 3 factors is weighted by the tuning parameters α', β', γ'. Making room to give some factors more importance than others.

Based on the value that results from the given location R_i, a formula is constructed to convert this value to a waiting time. The formula is described in Equation 2.4. The equation uses the variable v_{max} which is the maximum value of v from Equation 2.3. Note that in this calculation for v, the absolute maximum value that the fractions can become is 1. This leaves the maximum value for v described by $v_{max} = max\{0, \alpha'\} + max\{0, \beta'\} + max\{0, \gamma'\}$.

$$WaitingTime(R_i, t_i) = max\left\{0, Slack(R_i, t_i) \cdot \frac{v(R_i, t_i) - v_{max} \cdot \epsilon'}{v_{max} - v_{max} \cdot \epsilon'}\right\} \tag{2.4}$$

The parameter ϵ' is added. This defines the threshold the value of waiting must overcome. If the value v is not reaching the threshold, the fraction becomes negative, making the waiting time zero. If the value exceeds the threshold, the wait time will be a portion of the slack time. With the latest addition, the strategy now counts four tuning parameters, consisting of $\alpha', \beta', \gamma', \epsilon'$, where the first 3 are for weighting of the different factors and ϵ' for defining a threshold. Together, they form a strategy that is highly customizable and therefore has a high adaptability to the problem.

It does however also mean that extensive parameter tuning is required before this strategy is successfully implemented. Vonolfen and Affenzeller (2016) use a self-adaptive evolution strategy (ES) proposed by Beyer and Schwefel (2002). In this ES, the problem is simulated multiple times while the parameters are variated. In their specific ES, 2 parents, 16 children, and 50 generations are used. This means that 16 combinations of parameters are generated, and each instance is mutated by a small amount. The best 2 mutations are chosen that generate another 16 children. This process is repeated 50 times resulting in a well-performing combination of parameters. The parameterized waiting strategy is evaluated on a training scenario that is similar to the target scenario. The intensity strategy is visualized in Figure 2.4.

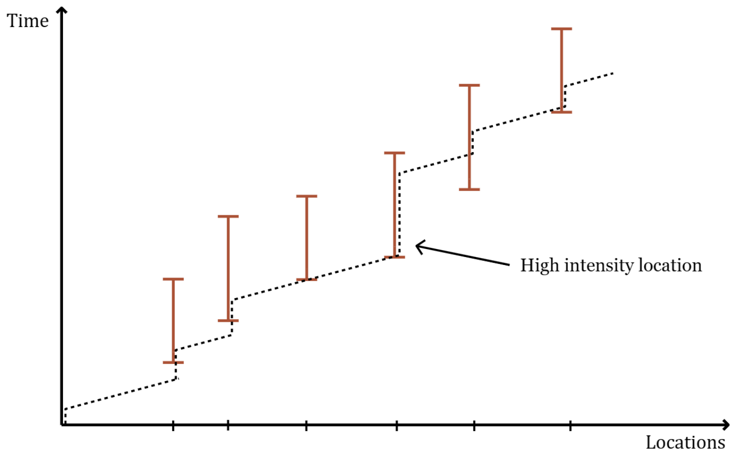


Figure 2.4.: The intensity strategy visualized. Where the vehicle waits at a location once a threshold in value is reached. The value is dependent on the intensity of the location, the transition time to the next location, and the remaining slack time.

2.3. Rebalancing strategies

Vehicle rebalancing is described as the movement of idle vehicles to preferred locations within the service area. A vehicle is considered idle when it is not actively serving a customer or moving toward a customer. Through rebalancing, waiting time for customers can be decreased and serviceability can be increased, as well as the utilization of the fleet. Different strategies for rebalancing have been developed in numerous studies. Each of them improves their solutions to their problem in its own way. Of course, no problem is the same, so there is no definitive answer to what strategy works best since this is highly dependent on the characteristics of the problem. A selection of strategies will be evaluated in this section. Some of them are more comprehensive than others.

2.3.1. Rebalance to desired distribution

When no information about future demand is available, static rebalancing strategies like presented by Pavone et al. (2012) are needed. Their idea is that there is some desired distribution of vehicles $v_i^d(t)$ over nodes $i \in \mathcal{N}$. Where $\mathcal{N}$ is the set of nodes in the service area. This distribution can be anything the user wants to insert and can be, for instance, the result of an empirical study. Pavone et al. (2012) demonstrate a possible distribution where all idle vehicles, also called excess vehicles $v_i^{\text{excess}}(t)$ are distributed equally over all nodes. They do this by defining the set of all vehicles as V and all customers waiting at node i is defined as $\sum_i c_i(t)$. The desired distribution is then given by Equation 2.5.

$$v_i^d(t) = \left\lfloor \frac{V - \sum_j c_j(t)}{n} \right\rfloor \text{ for each node i} \tag{2.5}$$

Where n is the number of nodes in the service area. This equation ensures that the desired number of vehicles stationed at a node adapts to the number of excess vehicles in the service area. This desired distribution can then be inserted in the linear program that aims to optimize this transition to the desired distribution. The linear program (LP) is given by:

$$\text{minimize} \sum_{i,j} \tau_{ij} \text{num}_{ij} \tag{2.6a}$$

$$\text{subject to: } v_i^{\text{excess}}(t) + \sum_{i \neq j} (\text{num}_{ji} - \text{num}_{ij}) \geq v_i^d(t) \qquad \forall i \in \mathcal{N} \tag{2.6b}$$

$$\text{num}_{ij} \in \mathbb{N} \qquad \forall i, j \in \mathcal{N}, i \neq j \tag{2.6c}$$

Where τ_{ij} represents the costs to travel from node i to node j and num_{ij} is the number of vehicles that travels from node i to node j. This linear program minimizes the rebalancing costs while making sure that every node in the service area has the desired number of excess vehicles.

The method is very straightforward and does not require any knowledge about demand in the future. But there also lies its weakness. Because future demand is not incorporated into this method, there is also no way of adapting the rebalancing strategy to fluctuations in future demand. This means this strategy will only provide adequate results when the demand corresponds to the expected demand on an ordinary day. This expected 'normal' demand can be incorporated into the desired distribution of vehicles $v_i^d(t)$. Pavone et al. (2012) decided to demonstrate that this distribution can be divided equally over the nodes, depending on how many excess vehicles there are in the service area. But in reality, a distribution that is tailored more to the specific situation would be preferable. For instance, by calibrating the preferred distribution to an average demand profile. In that case, the strategy would work optimally on an average day, but it would be unable to efficiently handle deviations from this average demand profile.

2.3.2. Rebalance to failed request locations

Rebalancing the idle vehicles of the system to locations where requests have been rejected is another form of rebalancing without knowledge of future demand. It is proposed by Alonso-

Mora, Samaranayake, et al. (2017), who use this strategy in an on-demand high-capacity ride-sharing scenario. Although this method does not explicitly incorporate predicted demand, it keeps in mind that a location with a failed request has the possibility that another request in that area can not be fulfilled with the current system state. It would improve serviceability to send an idle vehicle to that location to await future requests.

Alonso-Mora, Samaranayake, et al. (2017) define the set of undefined requests $\mathcal{R}_{ko}$ and idle vehicles $\mathcal{V}_{idle}$. Also, $\tau_{v,r}$ which is the travel time between a vehicle $v \in \mathcal{V}_{idle}$ and failed request $r \in \mathcal{R}_{ko}$. The variable $y_{v,r} \in \mathcal{Y}$ indicates the individual assignment of rebalancing vehicle v to failed request r. The linear program to optimally rebalance the idle vehicle looks as follows:

$$\text{minimize} \sum_{v \in \mathcal{V}_{idle}} \sum_{r \in \mathcal{R}_{ko}} \tau_{v,r} y_{v,r} \tag{2.7a}$$

$$\text{subject to:} \sum_{v \in \mathcal{V}_{idle}} \sum_{r \in \mathcal{R}_{ko}} y_{v,r} = \min(|\mathcal{V}_{idle}|, |\mathcal{R}_{ko}|) \tag{2.7b}$$

$$0 \leq y_{v,r} \leq 1 \qquad \forall y_{v,r} \in \mathcal{Y} \tag{2.7c}$$

Where the |...| denotes the number of elements of a set. This linear program produces the shortest rebalancing trips. Constraint b makes sure that the amount of rebalancing trips is either equal to the amount of failed requests, or to the number of idle vehicles. Whatever is less. The inputs $\mathcal{V}_{idle}$ and $\mathcal{R}_{ko}$ are the result of the batch assignment that was computed before the rebalancing phase. The procedure for the batch assignment and rebalancing is displayed in Algorithm 2.1.

Algorithm 2.1: Full procedure assignment and rebalancing

```
1 Vehicle-request assignment gives:
2     $\mathcal{R}_{ko} \leftarrow$ failed requests
3     $\mathcal{V}_{idle} \leftarrow$ idle vehicles
4 $\tau_{v,r} \leftarrow$ calculate travel time between every combination of $v$ and $r$
5 $y_{v,r} \leftarrow$ solve LP
```

When there are no failed requests, no rebalancing will take place. This saves costs during, for instance, nighttime when most requests can be satisfied because no unnecessary rebalancing will take place. On the other hand, rebalancing will only take place when requests are not getting satisfied, which may not always be the case.

While the previous strategy from Section 2.3.1 can be described as a static strategy, its desired distribution is determined *a priori* and does not react to the system state. The strategy described here is reactive. Meaning that the system only reacts to current states and not to future states. It recognizes where the system has failed and reacts to it by sending vehicles over. It does not, however, do anything to prevent the same problem from happening in the long-term future.

2.3.3. Rebalancing based on forecast

The strategy proposed by Iglesias et al. (2017) is the first strategy being discussed here that incorporates forecasting of demand. In their research, they compute their predictions using a Long Short-Term Memory (LSTM) neural network (Hochreiter and Schmidhuber, 1997). They describe their rebalancing strategy as model predictive control (MPC), where current state variables are the input and the rebalancing assignment is the controlled variable. To execute this MPC they propose an algorithm, summarized in Algorithm 2.2, that can be repeatedly called during the operation of the system.

Algorithm 2.2: MPC procedure

```
1 $\mathcal{S} \leftarrow$ count idle vehicles and estimate trip arrivals
2 $\lambda_{ij0} \leftarrow$ count outstanding customers
3 $\hat{\Lambda}_{t_0,t_0+T_{forward}} \leftarrow f(\theta_t)$
4 $\mathcal{X}, \mathcal{Y}, \mathcal{W}, \mathcal{D} \leftarrow$ solve problem 2.8
5 Assign $\{y_{ij1}\}_{ij1}$ to available vehicles
```

This algorithm is as follows: at a given time t_0 the available vehicles are observed and stored in the set $\mathcal{S}$. Also the outstanding customer demand λ_{ij0} is obtained. These 2 inputs can be observed in the current system state. Next, the future customer demand $\hat{\Lambda}_{t_0,t_0+T_{forward}}$ for the next $T_{forward}$ time steps is predicted. This is done using a prediction model f, trained with historical data θ_t. With these inputs the problem 2.8 can be solved which will result in the optimal rebalance strategy $\mathcal{Y}$. Then only the first time step of this rebalancing strategy y_{ij1} is assigned to the available vehicles.

Iglesias et al. (2017) define $w_{ijt} \in \mathcal{W}$ as the decision variable denoting the number of outstanding customers who wish to travel from i to j, and are picked up at time t. The slack variable $\mathcal{D} = \{d_{ijt}\}_{ijt}$ denotes the number of predicted customers that remain unsatisfied. c^w_{ijt} and c^d_{ijt} are the cost related to letting an outstanding customer wait, and the cost for not servicing a predicted customer. Similarly, the cost for rebalancing is c^y_{ijt} and it is related to the distance that is traveled.

To solve the optimal rebalancing strategy, a mixed-integer linear program (MILP) is used which is shown below.

$$\text{minimize} \sum_{(ijt)} c^y_{ijt} y_{ijt} + c^w_{ijt} w_{ijt} + c^d_{ijt} d_{ijt} \tag{2.8a}$$

$$\text{subject to: } x_{ijt} + d_{ijt} - w_{ijt} = \hat{\lambda}_{ijt} \qquad \forall i,j \in \mathcal{N}, t \in \mathcal{T} \tag{2.8b}$$

$$\sum_{j \in \mathcal{N}} x_{ijt} + y_{ijt} - x_{ijt-\tau ij} - y_{ijt-\tau ij} = s_{it} \qquad \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{2.8c}$$

$$\sum_{t \in \mathcal{T}} w_{ijt} = \lambda_{i,j,0} \qquad \forall i,j \in \mathcal{N} \tag{2.8d}$$

$$x_{ijt}, y_{ijt}, w_{ijt}, d_{ijt} \in \mathbb{N} \qquad \forall i,j \in \mathcal{N}, t \in \mathcal{T} \tag{2.8e}$$

The objective function in Equation 2.8a aims to minimize the costs related to rebalancing, customers waiting, and predicted customers rejected. Constraint 2.8b makes sure the assigned customers and rejected customers are equal to the predicted customers and the waiting customers. In other words, requests are either assigned or rejected. Constraint 2.8c states that the difference between the number of arriving vehicles and departing vehicles equals the net flow of vehicles in that region. Constraint 2.8d ensures that all outstanding passengers are set to wait and therefore will be served. The last constraint limits the decision variables to integers.

This MILP is more extensive than the LPs we have seen so far. It does not only include future predicted requests in the rebalancing decision-making, but it also allows tuning the parameters. By changing the cost of rebalancing, waiting, or rejecting, a preference can be imposed on the solution. When, for instance, the cost of rejecting is increased, the strategy will lean towards a solution where as many customers get served as possible. This will be at the cost of more distance traveled and longer waiting times, so the tuning depends on what is found more important for the specific case.

2.4. Summary

The studied waiting and rebalancing strategies can all be used in an urban waterborne transport environment. But each has its characteristics and limitations. The Drive First waiting strategy is the default waiting strategy, where a vehicle will drive to its destination once it gets the order. This strategy can form a baseline for the other strategies. The Wait First strategy will be an extreme waiting strategy. It will wait at every moment possible. According to studies, this will increase vehicle usage, but decrease the traveled distance. The Intensity waiting strategy will result, according to the literature, in a better performance than DF or WF. This is because a vehicle will only wait when the value of waiting is high enough. This strategy will however also be the most difficult to implement. Not only because of the various equations that have to be implemented but also because this strategy requires intensive tuning. The rebalancing strategies also have these differences in difficulty in implementation. The strategy that rebalances to a desired distribution needs the least preparation. A fixed distribution can be set, and idle vehicles will be assigned according to the ILP. Rebalancing to failed request locations requires one aspect more, the failed request locations need to be determined. Once these are extracted from the simulation, the assignment of the vehicles will be similar. It must be noted that for this strategy to work, there must be failed requests. Without failed requests, modifications would be needed for this strategy to still provide rebalancing actions. The rebalancing strategy with the most extensive implementation is rebalancing based on delay. This strategy requires a full prediction model and also needs tuning for the strategy to work. It is hard to determine what strategy will result in the best performance because this is very dependent on the situation. The rebalancing based on prediction strategy has the highest probability, however, because it is the only strategy that is tunable and includes predictions in its rebalancing assignment.

3. System characteristics

Not every PDP or VRP is the same. This is the reason why numerous strategies exist in the first place. Decisions made in a specific scenario can cause great improvements, but deteriorate the solution in another scenario. This is why it is important to analyze the problem thoroughly before methods can be applied. The system characteristics of the water taxi Rotterdam scenario consist of dock locations, the demand, trip statistics like durations and ride-sharing, and the network. These characteristics have to be extracted from the data that is provided by Flying Fish. The steps that have been taken to do this are described in this chapter.

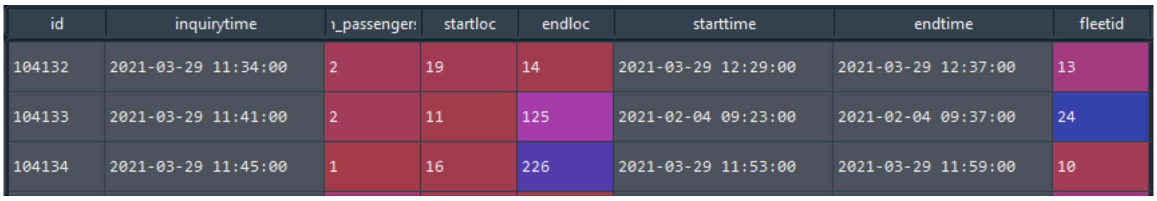
3.1. Data structure

The data provided by Flying Fish is a copy of the water taxi database for the period of one week. Every interaction with the system, change in planning, or update of estimated times is stored in this database. The data was shared as a MySQL dump file. Such a file is commonly used as a backup file for a database managed by the MySQL program. To access the data this file would have to be loaded in one's own MySQL database. This database was however not present and setting one up was not possible on short notice. The best alternative was to parse the SQL source file, extract the useful data, and convert it to a format that is interpretable by a Python program. The preferred format was CSV because of its ease of opening in MS Excel and a fast way to import using the Pandas module.

Once the conversion from SQL to CSV was made, a new compact data structure needed to be created. A single request could have more than generated 50 items in the database. This is because an item is generated with every change in planning or other updates of the request. All this data can be compressed to data that shows only the final planning. For now, that is all that is needed. To have an idea of how much time there was available to handle individual requests, the first interaction with the system is also stored in the compressed data. In the created compressed format, a single request consists of the following attributes:

- request ID
- Inquiry time
- Number of passengers
- Start location ID
- End location ID
- Start time
- End time
- Fleet ID
- Type

The fleet ID is the number of the boat that is assigned to the request and the type is imported as an attribute to later filter taxi rides only. This is because the cruises that the watertaxis also facilitate are not of importance for this research. A visual representation of the dataframe containing 3 requests is displayed in Figure 3.1.



id	inquirytime	ı_passenger:	startloc	endloc	starttime	endtime	fleetid
104132	2021-03-29 11:34:00	2	19	14	2021-03-29 12:29:00	2021-03-29 12:37:00	13
104133	2021-03-29 11:41:00	2	11	125	2021-02-04 09:23:00	2021-02-04 09:37:00	24
104134	2021-03-29 11:45:00	1	16	226	2021-03-29 11:53:00	2021-03-29 11:59:00	10

Figure 3.1.: The request data compressed into a Pandas DataFrame. 3 separate requests are shown.

This structured data made acquiring the information needed to construct the system characteristics possible.

3.2. Locations

Watertaxi Rotterdam offers its service to 248 locations in and around Rotterdam. The locations are specified with the name, the ID number, and the coordinates. The coordinates can be used to project the locations on a map. The locations, placed on the map can be seen in Figure 3.2. Not all of these locations are directly bookable. On the Watertaxi website, only 50 locations are presented. The remaining locations are not publicly available and can only be booked after consultation with Watertaxi Rotterdam.

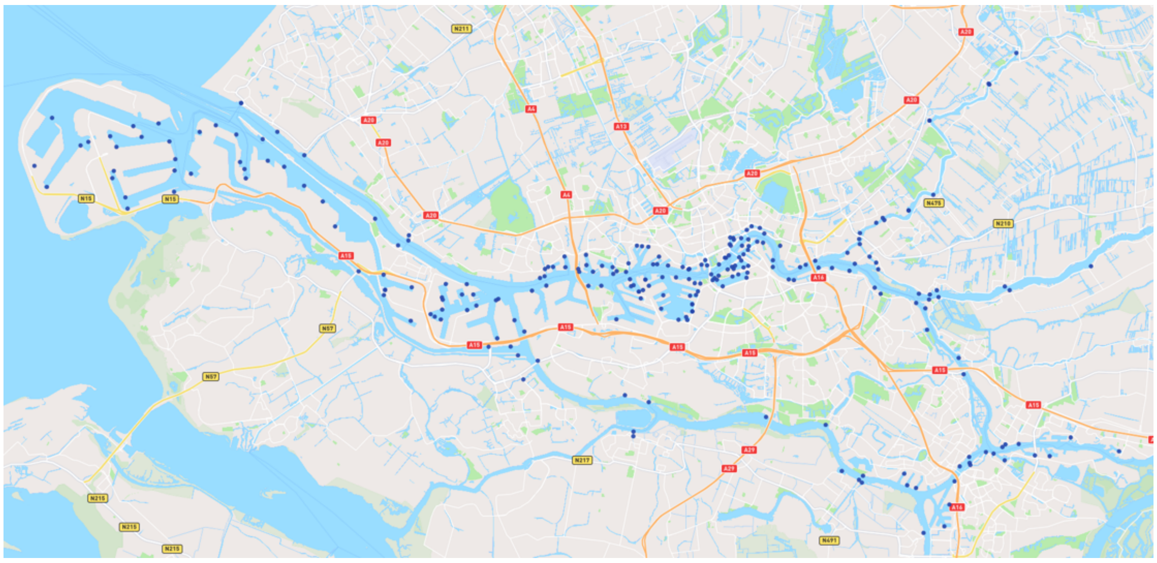


Figure 3.2.: A map of the Rotterdam area where every dot represents a location where Rotterdam Watertaxi offers its service.

The locations that are actually used in the provided week of data are presented in Figure 3.3. One additional location was used in the Maasvlakte port area, far from the urban area. However, for a more detailed map of the urban area, this location is not displayed. In one week, 54 locations were used. This number corresponds better with the number of available locations on the website. The extra locations are the result of custom rides that are negotiated with Watertaxi Rotterdam.

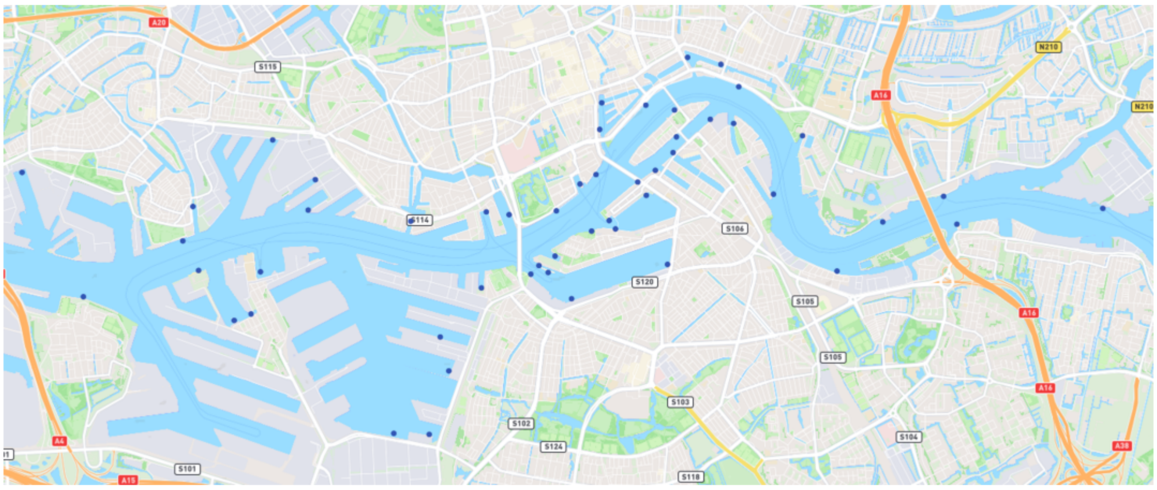


Figure 3.3.: The locations that are actually used in the week of data

3.3. Demand

The demand arising from a certain location can be described by the number of requests that emerge from that location. When the requests in all locations are combined, this results in the total demand. The total demand can be an indication of how 'busy' a certain period in time was for the system as a whole. The total number of requests, grouped per hour, is displayed in Figure 3.4. Note that there are some requests that appear in the following week. These requests are clearly incomplete and will not be taken into account when performing data analysis.

The total demand is useful for determining the overall load on the system. However, in order to reproduce the demand of a real-life scenario, a more detailed analysis is needed. The demand per location is needed to determine how active locations are during the day and how they differ from each other. Some locations will have demand over the whole day while others have spikes in demand during the morning and evening. The average demand over the whole week per location is visible in Figure 3.5.

It is clearly visible that requests are focused around the city center. With the location near Hotel New York as the most popular. This is also where the terminal of Watertaxi Rotterdam is located. The waterways around Rotterdam have formed because of port construction, transforming the southern bank into its iconic shape starting from 1870 (Laar, 2022). The earlier port construction created a small peninsula at that location, making it an efficient location for waterborne transportation. The second most used location is the one in the Leuvenhaven, near the Maritime Museum. This location is the most central location of the Rotterdam city center, so the popularity of this location is explainable.

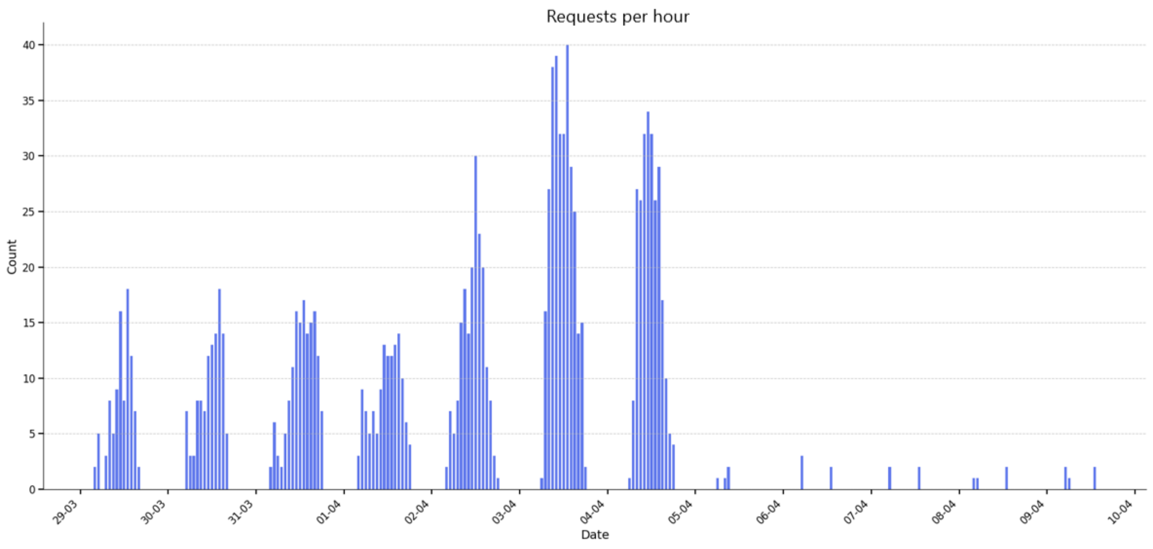


Figure 3.4.: Total requests registered for the week of data provided. Showing the number of request that have been made per hour.

3.4. Trips

Trips are different from requests. In most cases, a trip will result from a request. A request will become a trip when a taxi and a start time are assigned to that request. A request that is infeasible will not become a trip. Also, the requested start time may be adapted to fit the schedule of the taxi. Another important difference is that two requests can be merged into one trip. The two requests will then be ride-sharing the trip.

3.4.1. Ride-sharing

So Ride-sharing is a way to merge multiple requests into one trip. To know how many of the rides are shared, a clear distinction needs to be made. The data received was only structured in requests. To see whether some of these requests were merged into one ride needed some extra investigation. In an earlier stage, hard requirements were set in order for a ride to be considered as shared. The start location as well as the start time needed to match perfectly. This resulted in low numbers of ride-sharing because sometimes the data could be one minute off, resulting in an unrecognized shared ride. The better option was to label a ride as shared once the travel interval of successive requests for a single taxi overlapped with each other. Request overlap when the start of the second request s_2 happens earlier in time than the end of the first request e_1. This is also visualized in Figure 3.6.

This method results in 119 requests that have been shared over one week. Compared to all 1232 requests in that week, this results in a ride-sharing percentage of about 10%.

This more flexible method also allowed the recognition of shared requests that do not have the same pick-up point but do have the same destination. This was not possible with the method using hard requirements. The shared request data shows that the majority of shared requests have the same origin and destination but mixed locations are not uncommon as well. A breakdown of the shared rides per day, compared to the total number of rides is shown in Figure 3.7.

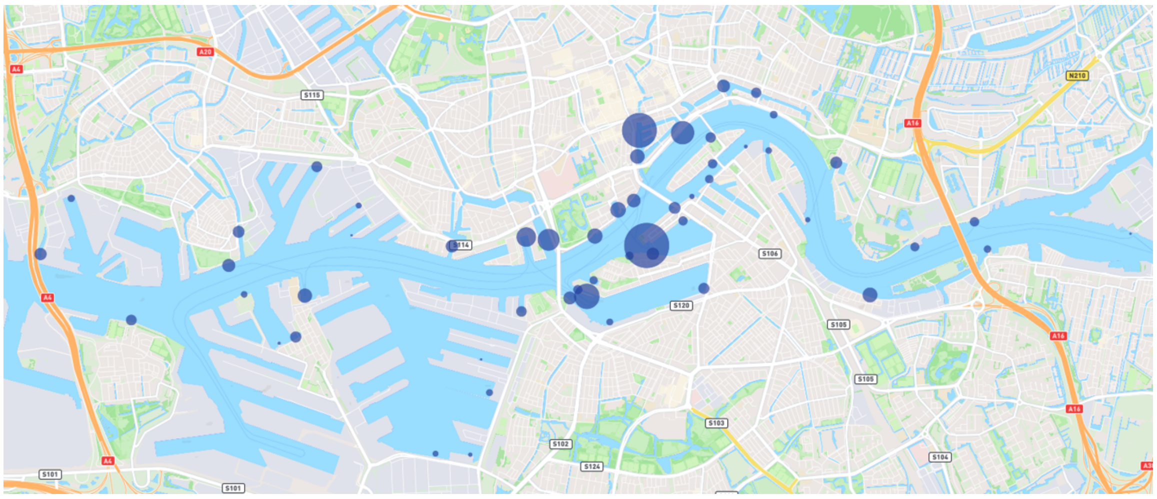


Figure 3.5.: The demand per location visualized, where the size of the circle is proportional to the number of requests emerging from the location.

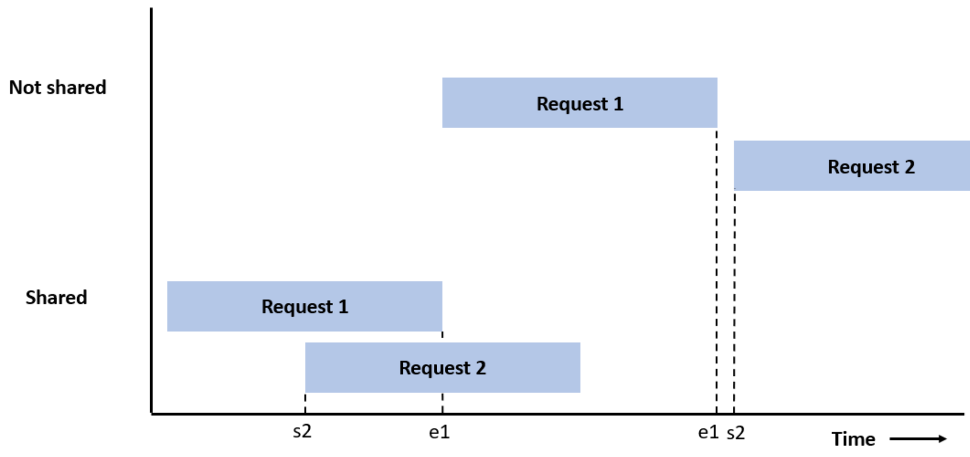


Figure 3.6.: Visualization of the definition of shared requests

3.4.2. Trip statistics

Apart from the ride-sharing aspect, there are more interesting statistics in relation to the trips that can be extracted from the data. These statistics give more insight into the real-life scenario. With that extra information, a simulated environment of that scenario can be replicated more accurately.

Trip duration

The duration of a trip is an important factor that can identify if everything in the trip has worked accordingly. The duration of the trip is a combination of entering the boat, actual transit of the boat, and leaving the boat. The duration can tell us something about the speed of the boat and

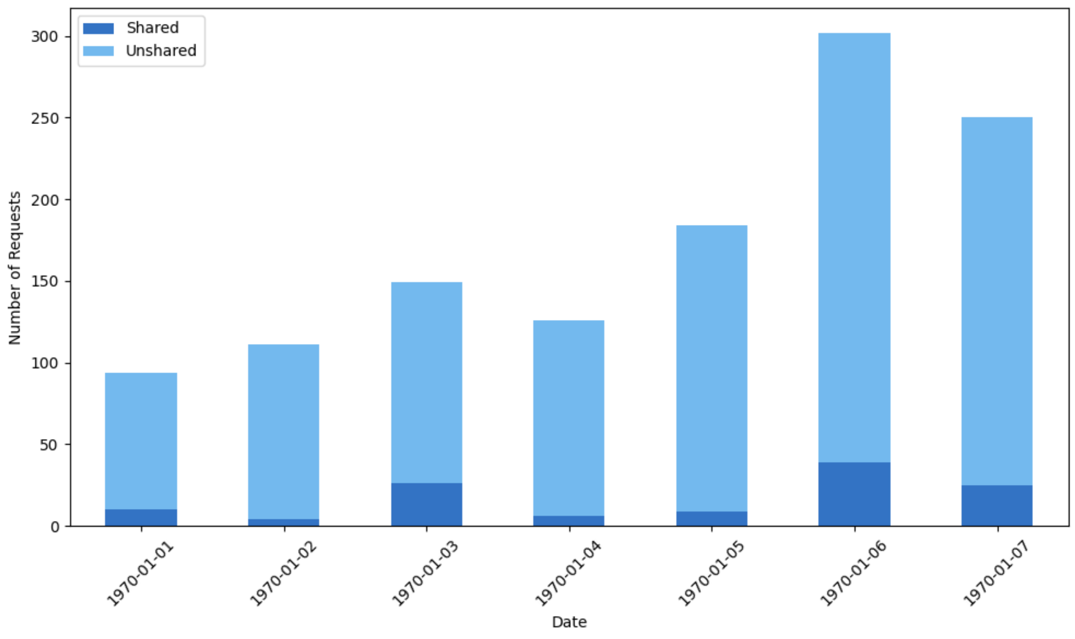


Figure 3.7.: The number of rides per day, split into two parts. The rides that are shared below with on top the remaining 'unshared' rides.

how long the loading and offloading of the taxi typically takes. The derivation of these properties will be further handled in Section 3.5.4. The distribution of trip duration according to the provided data is shown in Figure 3.8.

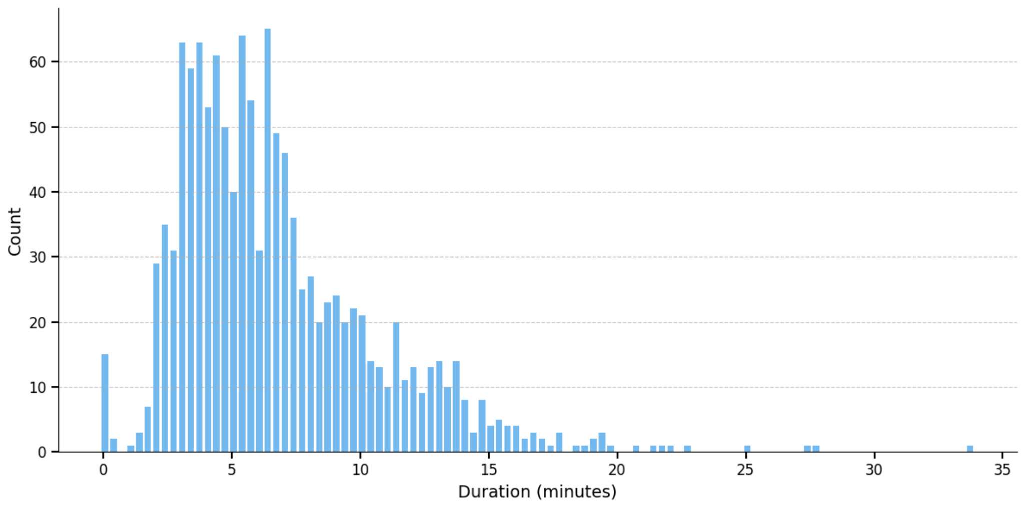


Figure 3.8.: The duration of all trips, depicted as a distribution grouped in intervals of 20 seconds.

The figure shows that most trips are completed within 10 minutes, although a higher duration is not uncommon. There are a few trips with higher durations than 15 minutes. This is either caused by an unusually long distance or unforeseen circumstances that cause a delay. There is also a spike at zero. After discussing this with Flying Fish, it was determined that this is caused by erroneous usage of the administrative system and can be ignored. The same holds for negative values that have been removed from the figure.

Inquiry duration

In addition to the duration of a trip, there is also another duration that indicates the performance of the system. The inquiry duration is the time that has elapsed between the inquiry of the trip and the start of a trip. It indicates how far ahead of time customers request their taxi trip. The duration between inquiry and the start of a trip can consist of 2 parts. One is the time between the inquiry and the requested start time, and the other one is a potential delay between the requested and actual start time. The inquiry duration indicates the dynamism of the problem. If all customers request a taxi with a starting time as soon as possible, the problem is dynamic. The inquiry duration for all requests will then only consist of delay. On the other hand, when all customers request a taxi for the next day the problem is static. The inquiry duration will then only consist of the time between inquiry and the requested start time, assuming the system has enough capacity. In this case, we are dealing with a combination of both, where it is uncertain what the ratio of these two parts is. The wide range of inquiry durations can be seen in Figure 3.9.

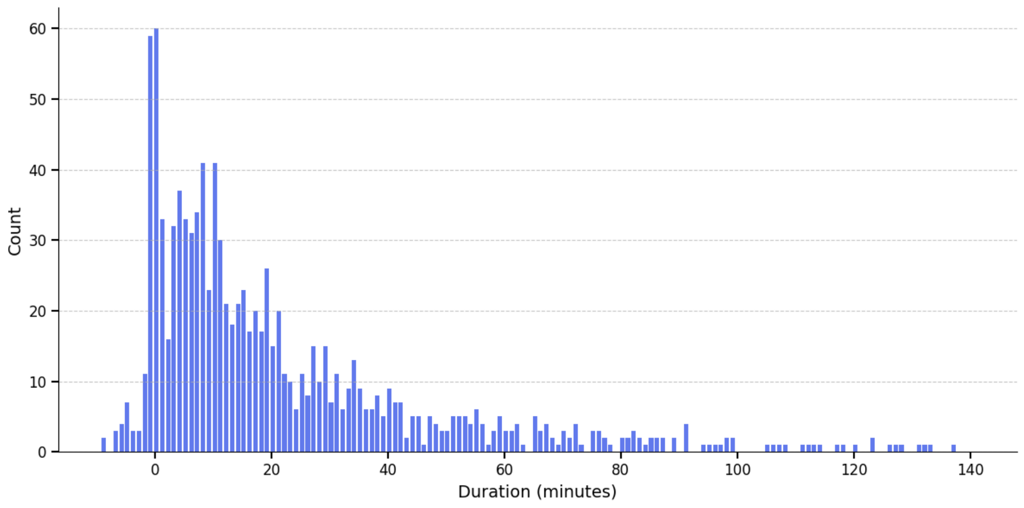


Figure 3.9.: The distribution of inquiry duration of all trips. Only durations shorter than 140 minutes are displayed

In this figure, the durations longer than 140 minutes have been dropped in favor of visibility. The inquiry durations extend to a maximum of 27 days, but the number of occurrences drops drastically. From this wide variety of inquiry durations, we can state that the problem is partially dynamic. Inquiry durations of 10 minutes or lower have a high probability that they were dynamically requested, consisting only of delay. It is also visible that a lot of requests were served directly. This is a result of customers walking up to a boat and asking for a ride. Since the inquiry was made at the same time the passengers moved on to the boat, the inquiry duration is zero. The negative values are again caused by erroneous use of the administrative system.

Number of passengers

The number of passengers on a trip indicates how much of the boat capacity is being used. The number of passengers is influenced by the request size and ride-sharing. When a ride is shared, the number of passengers from both requests is added. The ride can therefore only be shared when the total number of passengers is lower or equal to the capacity of the taxi. Without ride-sharing, the number of passengers in a taxi is only influenced by the group size of the requests. A full or nearly full taxi will then be uncommon because groups usually tend to be smaller in size. An increased amount of ride-sharing causes more usage of the taxi's large capacity, thus, efficient operation. The distribution of the number of passengers is displayed in Figure 3.10.

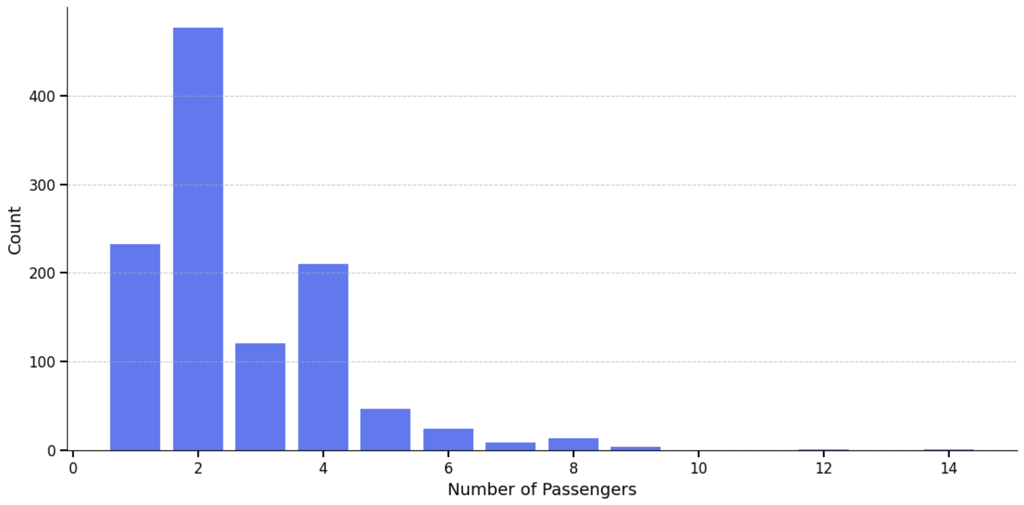


Figure 3.10.: A distribution of the number of passengers for all trips.

The figure shows that the most common group size is 3, followed by 1 and 4, and after that 3. An interesting distribution but can be partially explained by ride-sharing. If 2 is the dominant group size it makes 4 the most usual size of shared rides, which partially explains the increase in 4 as group sizes. The figure also shows 14 as the number of passengers that occurred at one time. This was due to the sharing of a group of 10 and 4. This shouldn't be possible but the group consisted of 2 children so probably they made an exception or the group size in reality was not as large. The limited amount of ride-sharing in general can also be seen from this figure. Because the distribution is mainly determined by the group size of original requests and a high percentage of the capacity is reached rarely.

3.5. Network

The water taxi system can be seen as a large network. Where the docks are nodes and the paths between the docks are links. If we can create a virtual model of this network, calculations can be made to reveal more properties about trips. Moreover, if there is going to be a simulation of the water taxi operation, a network can serve as an input for that simulation as well. Having a network is beneficial, but how is an accurate network of this environment created? Most of the docks can not be connected in a straight line. But drawing out and measuring every possible path from one dock to another would be a very time-consuming task. In the process of making an accurate network a simple version was created to start with, to test if investing more effort was worth it. Later a more extensive and accurate network was created to achieve better results.

3.5.1. Simple network

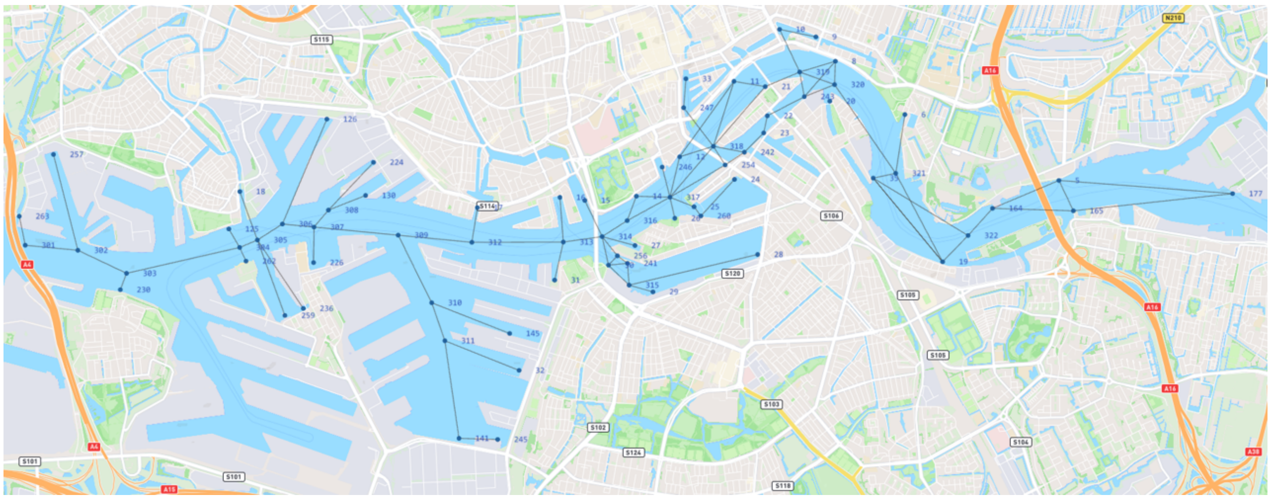


Figure 3.11.: The simple network shown on the map. Displayed with node numbers. Numbers higher than 300 are added intermediate nodes.

The goal of the simple network was to create a network as a proof of concept while keeping the amount of effort needed to create it limited. Since there were already nodes for every dock a taxi could visit, only intermediate nodes needed to be added so that a possible path between docks would be somewhat realistic. In other words, possible paths should not cross land. With the insertion of 22 nodes, this was broadly achieved. The nodes were inserted in the middle of the river or in the middle of major branches of the river. The coordinates of the inserted nodes were noted for later distance calculation. With the nodes in place, the links between them can be created. In total 93 links were made. In the process, it was made sure that links between nodes can only exist if the connection does not cross land. The visual representation of the network, projected on the map is shown in Figure 3.11. The only thing left is assigning length to the links. This was done by calculating the great circle distance between the 2 points using the Haversine formula (Inman, 1849). The node and link information could be used to perform network calculations on, like shortest path, so the proof of concept was accomplished.

However, as expected, the network lacked accuracy and most paths were not realistic. Illustrated by Figure 3.11, in numerous situations the path taken to get from one dock to another results in an unrealistic detour when following this network. A solution must be found where a pathfinder has more freedom in choosing its path, just like a skipper on a water taxi determines its route on the river. In the following section, the creation of this type of network is elaborated.

3.5.2. Grid-based network

When the feature is added of having the freedom of choosing an undetermined path in a bounded area, a grid-like network is needed. This grid will cover the area over which the taxi will operate, in this case, the water. Docks will have to be assigned to nodes on this grid and also the smaller waterways should be reachable using this grid size. The nodes should therefore be spaced close enough together to provide an accurate representation of the real world. On the other side, decreasing the distance between nodes also increases the amount of nodes needed and so, the size of the network. Finding the right balance in these factors was necessary to create an accurate but computationally light enough model to work with.

The start of the process involved defining the service area. Since the data included only one trip outside of the urban area to the, more remote, port area, it was decided to keep the service area inside of this urban environment. See Figure 3.12 for the shape of the chosen area. Every point in the polygon was drawn manually. The coordinates to construct this large polygon were stored inside a GeoJSON, a dictionary file specialized in storing these types of geographical data.

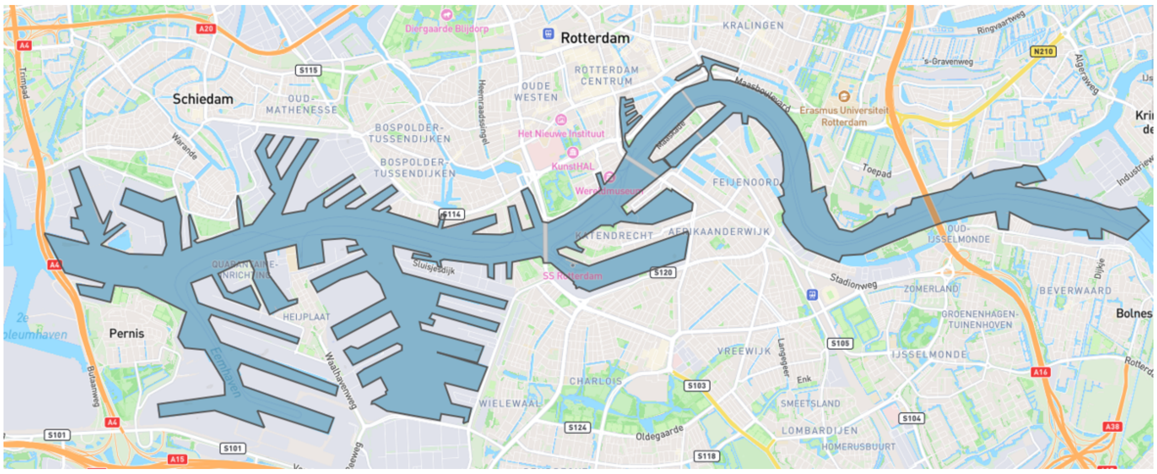


Figure 3.12.: The polygon that resembles the service area for the water taxis. Consists of the body of water that includes every dock used.

Next, a grid needs to be generated inside of this polygon. It was decided to use the Hexagonal Hierarchical Spatial Index, or H3, developed by Uber (Brodsky, 2018). This grid system is easy to use, has multiple layers of resolution, and offers a lot of Python features. One of those features is to get grid points, at a desired resolution, inside a defined polygon stored in a GeoJSON. This function is just what was needed to create a grid in the service area. Illustrated in Figure 3.13 is that a resolution of 10 was not accurate enough to represent the smaller waterways. A resolution of 10 means that the hexagons that divide the grid have an edge length of 75 meters. A grid size that was assumed to be small enough considering the river that has a length of 15 kilometers inside the service area.

When the resolution is increased by one, every hexagon splits into 7 smaller hexagons. Increasing the number of points on the grid by 7. A H3 resolution of 11 fits the service area better, visible in Figure 3.14. Places where a single hexagon did not fit before, now fit multiple.

In total, the grid consists of 7841 nodes, where each node represents an area of 2150 m^2 or a hexagon with edges of 28 meters. As said before, the increase in grid size does not only come with benefits. Computation time for every step in the analysis increases as well. To give an example, the generation of the grid at resolution 10 costs 3.1 seconds while the grid generation at resolution 11 costs 19.5 seconds. The larger grid computational time is more than 6 times longer.

The links of the network connect the newly generated nodes with each other. A link is defined as a set of 2 node IDs. The order of the IDs does not matter because in this case, all links are bi-

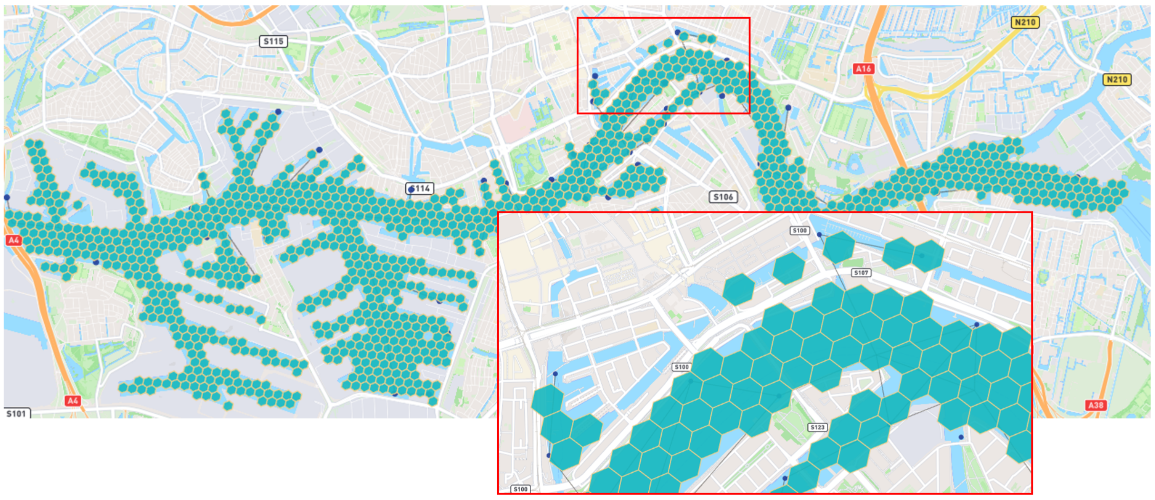


Figure 3.13.: A H3 grid of resolution 10, bounded by the service area polygon. Resolution is not good enough for smaller waterways.

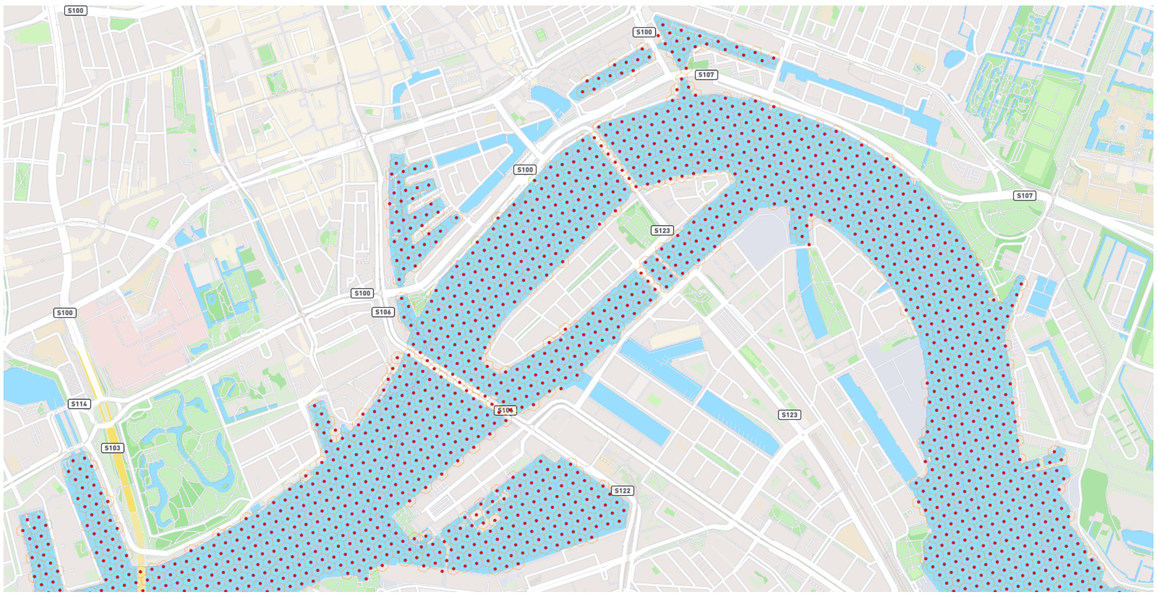


Figure 3.14.: The central part of the service area covered with a H3 grid of resolution 11. Every dot represents the center of the hexagon and will be a node in the network.

directional. Generally, every node has a link with every directly neighboring node. A node in the center of the grid will consequently have links going in 6 directions. Another clever functionality of the H3 package in Python is that it is possible to get the neighbors of a specific node. When implemented in a loop that also checks that the neighbor is within the service area, almost all the links can be generated. Almost, because there were a few cases that required manual modification. In some places, nodes were indeed neighbors but there was a thin slice of land in between them that prevented a connection between them for a vessel. And also the other way around, where due to a bridge the nodes were separated 2 tiles apart but should have a connection. The result of the process is visible in Figure 3.15.

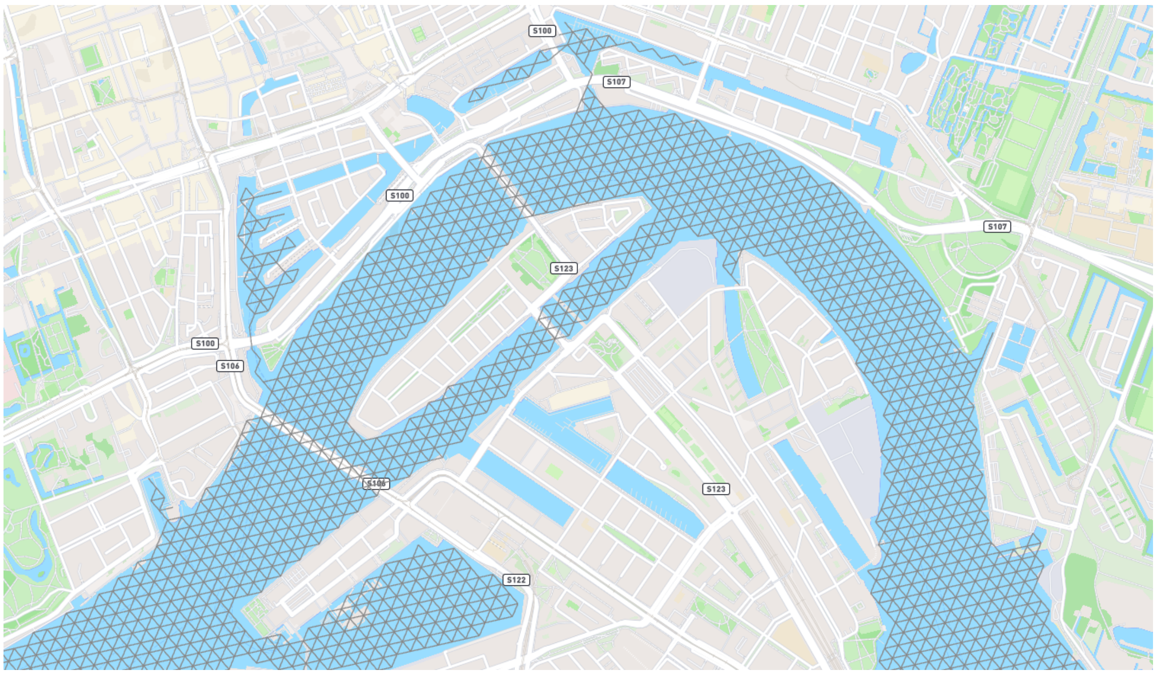


Figure 3.15.: The links that connect the network. A center node has a link with each of its six neighboring nodes.

There were 41837 links created in total. This costs 22 seconds of computation time. A significant part of that also comes from iterating through the links and removing the ones that correspond with the IDs of links that should be removed. With the nodes created and links assigned to them, the network is set up and ready for analysis.

3.5.3. Distances

The key information we would like to extract from the network is how much distance a vessel travels from one specified dock to another. This is useful information because it can ultimately lead to an estimation of the duration of trips.

When a distance is desired from one point to another, first, a path needs to be determined. A path is the sequence of nodes an agent travels to reach its destination. In large networks, and certainly in this case, a large number of paths can take an agent to the same destination, so what is the right one? In reality, a water taxi will travel a path that is most beneficial for the company and the customer. The customer likes the shortest amount of travel time and the company would like the least amount of fuel consumed. Both interests result in the same path, the shortest one. So that is the path that the water taxi takes, thus the path that needs to be found in the network.

Luckily, finding the shortest paths in networks is a very developed field of research and has led to numerous software applications that perform network analysis. A popular one of them is the Python package NetworkX (2014), which has multiple built-in options for finding shortest paths and more. The package was used to find the shortest paths for all trips in the data. The shortest paths for all recorded trips in the data are visualized in Figure 3.16. Apart from the actual trips, the network allows us to get a shortest distance from every combination of docks possible. When all of those distances are organized in a table, we get the distance matrix of the water taxi network, visible in Table 3.1. In this matrix, the columns are the origin of the path, and the rows are the destination, or vice versa. This matrix is the basis for trip duration estimation and will be discussed in the next section.

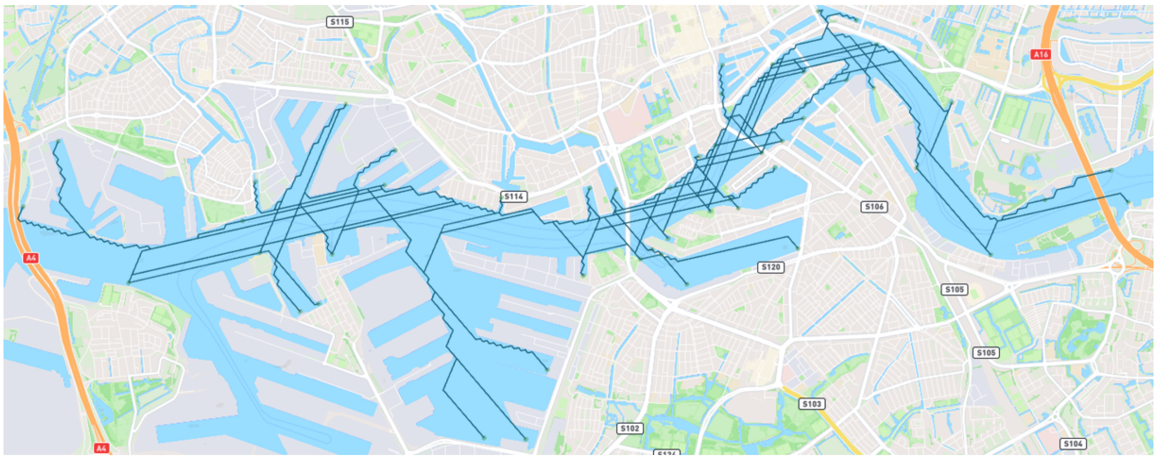


Figure 3.16.: The generated shortest paths for all recorded trips, projected on the map.

Table 3.1.: A portion of the distance matrix, stating the shortest distances in meters between a chosen combination of docks.

	5	**6**	**8**	**9**	**10**	**11**	**12**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
5		3166	4051	5026	4793	5028	5907	6662	7544	7962	8992	12087	1884	3880
6	3166		1227	2408	2175	2409	3289	4044	4926	5344	6374	9469	2010	1262
8	4051	1227		1180	948	1182	2317	3073	3955	4372	5403	8498	2895	548
9	5026	2408	1180		500	1307	2485	3241	4122	4540	5571	8666	3870	1369
10	4793	2175	948	500		1074	2252	3008	3890	4307	5338	8433	3637	1136
11	5028	2409	1182	1307	1074		1178	1933	2815	3233	4263	7358	3872	1371
12	5907	3289	2317	2485	2252	1178		755	1637	2055	3085	6180	4751	2250
14	6662	4044	3073	3241	3008	1933	755		882	1299	2330	5425	5507	3006
15	7544	4926	3955	4122	3890	2815	1637	882		863	1894	4989	6389	3887
16	7962	5344	4372	4540	4307	3233	2055	1299	863		1699	4794	6806	4305
17	8992	6374	5403	5571	5338	4263	3085	2330	1894	1699		3463	7837	5336
18	12087	9469	8498	8666	8433	7358	6180	5425	4989	4794	3463		10932	8431
19	1884	2010	2895	3870	3637	3872	4751	5507	6389	6806	7837	10932		2724
20	3880	1262	548	1369	1136	1371	2250	3006	3887	4305	5336	8431	2724	

3.5.4. Duration

As stated in Section 3.4.2, trip duration is the amount of time that passes between passengers boarding, and leaving the boat. It is a valuable resource to roughly know how long a certain trip will take beforehand. With that knowledge, a company is able to make a planning and give customers information about their estimated time of arrival for instance. With a large data set, this estimation can be made by averaging the value of the durations of a certain trip. Another way would be to use the distance of a trip and divide that by the speed of the boat to get a duration. Both ways have their downsides. The data set would have to be very large to cover all possible trips. In this case, only 231 unique trips were recorded compared to 1250 possible unique trips. If the speed would be used to calculate the duration, what speed would be the best to use? And what if some trips have busier waterways or include waterways with a speed limit? And what time should be incorporated for boarding and leaving the boat? The following method aims to combine the two ways and answer these questions.

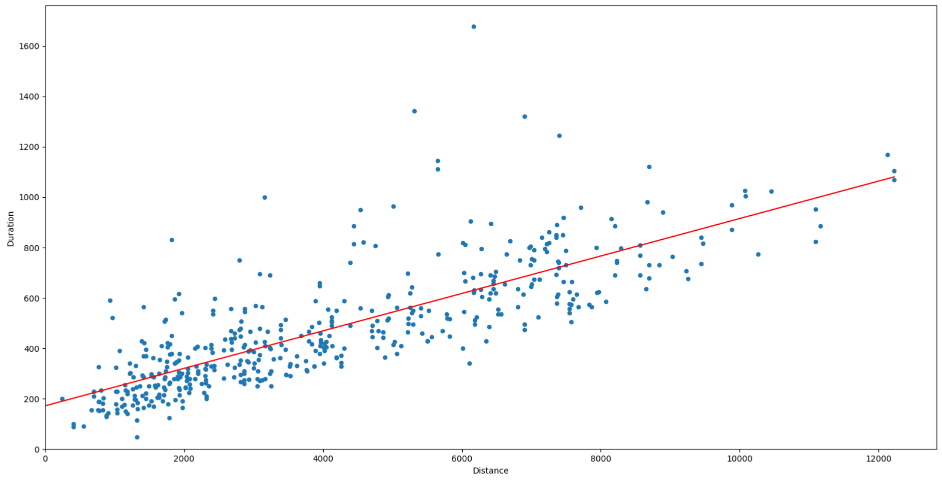


Figure 3.17.: The relation of duration vs distance in the data, the ride line is the resulting linear function.

With the calculated distances and recorded durations, there is a good baseline for estimations of trip durations. The best estimation is the average value of previous trips. This duration includes possible obstacles or advantages that are trip-specific. But, as previously stated, there are a lot of unknown trip durations. However, with all of the recorded durations and the distances of their trips, there are a lot of data points usable for defining a relation between duration and distance. From looking at the data points in Figure 3.17, the duration has a high deviation, but roughly follows a linear relation with the distance. The straight line drawn does not only give an average speed of the boat but the offset with the x axis is the average amount of time needed for passengers to board and leave the boat. The data shows an average speed of 48,4 km/h with an offset time of 172 seconds.

With this relation specified, the trips with unknown durations can now be estimated as well. We can put the duration of every dock combination in a similar matrix as is done in Table 3.1. Only here, the recorded durations are averaged per trip and unknown durations are calculated using the duration and the distance matrix. The result is shown in Table 3.2.

	5	**6**	**8**	**9**	**10**	**11**	**12**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
5		410	473	546	528	546	611	667	578	764	840	1070	312	460
6	280		263	351	334	351	416	406	538	569	646	876	321	266
8	473	263		260	242	260	200	400	380	497	574	804	387	90
9	546	351	260		209	269	357	413	478	560	586	816	460	274
10	528	334	242	209		252	339	396	392	400	569	799	442	257
11	426	385	260	269	252		260	215	266	300	330	719	330	274
12	611	358	344	357	260	140		228	218	325	273	630	525	339
14	667	427	355	398	340	237	228		238	269	345	460	430	395
15	558	512	659	490	589	353	300	238		236	313	543	595	461
16	764	569	497	949	588	402	302	269	236		298	470	635	492
17	840	646	574	586	569	489	401	345	313	298		395	754	569
18	1070	817	804	816	799	719	631	575	543	528	429		984	799
19	312	321	387	460	442	330	808	430	485	565	575	984		375
20	460	266	213	274	178	274	339	395	461	492	569	799	375	

Table 3.2.: A portion of the duration matrix, stating the estimated travel duration in seconds between a chosen combination of docks.

4. Experimental setup

With the relevant data extracted described in Chapter 3, an environment can now be set up to test the strategies. To get relevant and comparable results from the experiments, the experimental setup needs to be close to reality. However, the computational expense of the experiments can not be too long because otherwise the tuning and testing of strategies can not be completed within the project's time. Finding a good balance between realisticness and feasibility will be an important theme in this chapter. The chapter starts by explaining key parts of the simulation model that was provided. As will be explained, a lot of modifications needed to be made to make the simulation environment fit the real-life scenario of the water taxi case. This chapter concludes by constructing a clear benchmark for further testing called, the base case. Furthermore, simulation settings that will be used for all experiments will displayed and elaborated.

4.1. Simulation model

A working simulation model was provided by Breno Alves Beirigo. This model was already applied to simulate a taxi service in Manhattan, New York. It creates the scenario by loading the Manhattan road network as adjacency and distance matrices. This, together with the coordinates of the nodes creates a network where calculations can be performed on, just like the network that was created in Section 3.5. The scenario is not complete with the network alone. Demand data is loaded using a CSV file of the historical data of the New York taxis. Other characteristics like the fleet size, the capacity per vehicle, or the speed at which the vehicles travel are specified in JSON input files. With all the data and settings provided, the model can be executed and simulate the taxi service. It does this by handling the incoming requests and matching the requests with vehicles to form a trip. It matches multiple requests in one vehicle if the constraints allow it. Every timestep, the fleet and request status is updated accordingly. It also has already implemented an optional rebalancing strategy, that follows the principle of Alonso-Mora, Samaranayake, et al. (2017), where vehicles rebalance to locations where requests were previously rejected. In the following sections, these components are further elaborated.

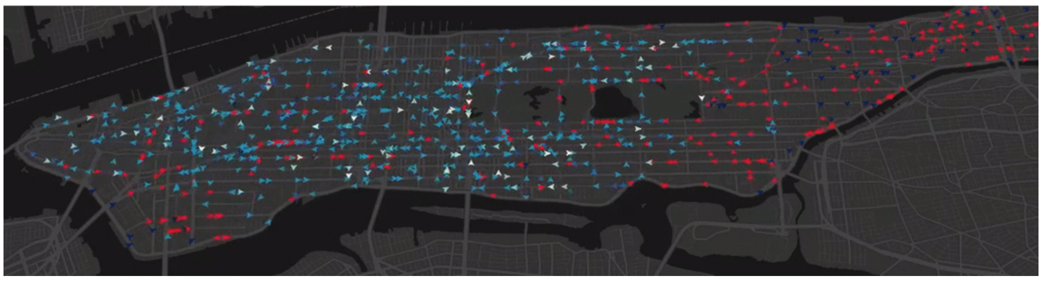


Figure 4.1.: A visualization of the simulation of the New York taxi service, where the arrows indicate a taxi either servicing, waiting, or rebalancing.

4.1.1. Request generation and handling

A request consists out of the following attributes:

- Request ID
- Start time
- Start location
- End location
- Number of passengers

This is all the information that is needed for the simulation to work with. Requests need to be introduced to the system in a controlled manner. The number of requests per minute should match the number of requests that are recorded in reality in order to create a realistic scenario. The historic request data is stored in a file. First, the complete data is read in order to create the full request list. Then, for every timestep of the simulation, a batch of requests is pulled from the total request list. This is done by calling the method *getRequestsBewteen()* that has the start time of the timestep and the end time of the timestep as input parameters. The method will search the list and will return any request that has a start time between the two input times, see also Figure 4.2. The requests are added to the *unassignedUsers* group which means they will be included in the next matching round.

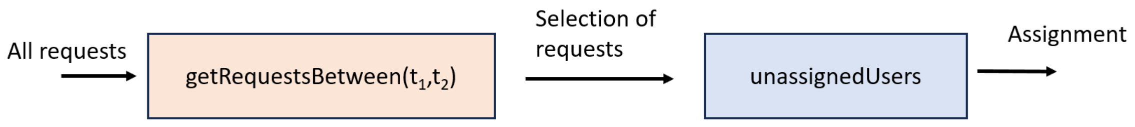


Figure 4.2.: The method for the generation of requests.

4.1.2. Matching

The group of unassigned requests is an input for the matching method. The method that is used in the model is a matching method proposed by Alonso-Mora, Wallar, and Rus, 2017. The high-level steps of this method are shown in Figure 4.3. The unassigned requests, together with the current time, the time windows of requests, and the location and status of vehicles form the input for an RTV graph graph. An RTV graph explores the feasible combinations of requests and vehicles. The RTV graph then serves as an input for the Integer Linear Program (ILP) that solves the optimal assignment of requests to vehicles. Depending on constraints and time windows, it is possible that not all requests are assigned. These requests are put back into the unassigned request group so that they are included in the next matching round. Only when the current time is outside the time window of the requests, the request is rejected. The RTV graph and the ILP assignment are further explained below.

RTV graph

An RTV graph can be seen as a sort of pre-processing step of the inputs for the assignment ILP. It creates every feasible combination of vehicles and requests. This method also considers combinations of multiple requests and one vehicle, meaning that ride-sharing is included in this

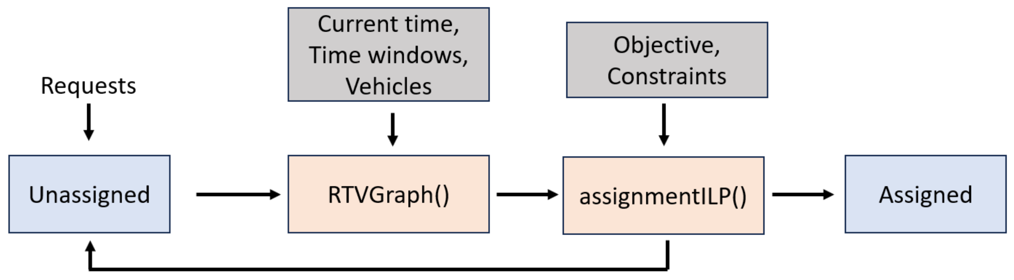


Figure 4.3.: The high-level steps necessary for matching requests to vehicles.

matching method. The first step, also visible in Figure 4.4, is the construction of a pairwise request-vehicle shareability graph (RV graph). In the RV graph, pairs of requests and vehicles are made if a vehicle can serve the request within the time constraints. Also, request pairs are formed if the time constraints allow the service of both requests in one trip. The results of the RV graph are request-vehicle pairs and request-request pairs. The RTV graph then combines the pairs created in the RV graph. A feasible trip always consists of a request-vehicle pair, with optional request-request pairs added to that. When all pair combinations are made, all feasible combinations of requests and vehicles are made and the RTV graph is finished. This then serves as an input for the ILP assignment.

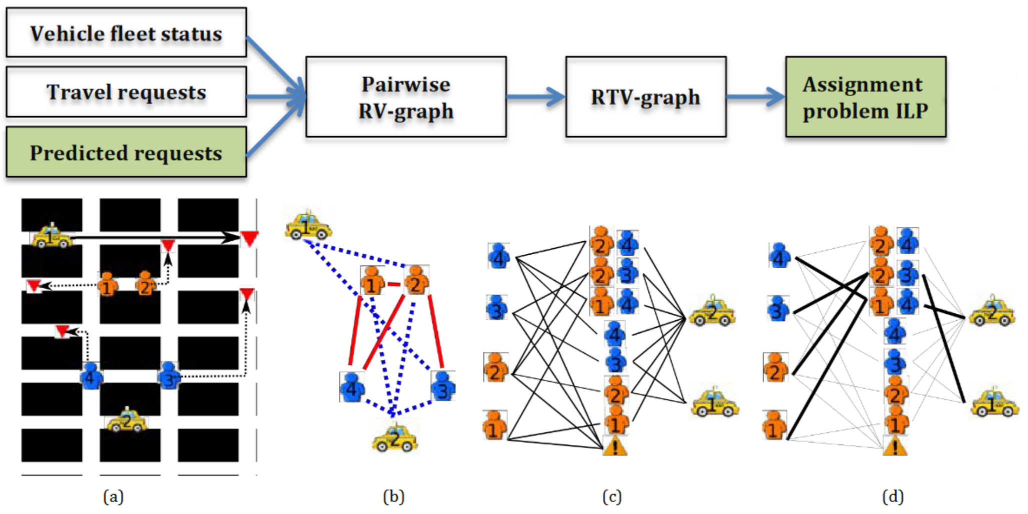


Figure 4.4.: The construction of an RTV graph visualized by Alonso-Mora, Wallar, and Rus (2017)

ILP assignment

What possible combinations eventually are the best choice, is determined via linear programming. Here, the optimal assignment of vehicles to trips is determined by minimizing an objective function while complying with the constraints

A binary variable X_{ij} is created. The variable is equal to 1 if an assignment between a trip i and a vehicle j is made, and zero otherwise. The amount of delay that results from a trip $i \in \mathcal{T}$, carried out by vehicle $j \in \mathcal{V}$ is denoted as d_{ij}. The number of requests that are served in a trip $i \in \mathcal{T}$ is denoted as r_i. The formulation of the assignment ILP is shown in Equation 4.1.

$$
\begin{aligned}
\text{maximize:} \quad & \sum_{i \in \mathcal{T}} r_i x_{ij} - \sum_{i \in \mathcal{T}, j \in \mathcal{V}} 10^{-5} d_{ij} x_{ij} & & \text{(4.1a)} \\
\text{subject to:} \quad & \sum_{j \in \mathcal{V}} x_{ij} \leq 1 & \forall i \in \mathcal{T} & \quad \text{(4.1b)} \\
& \sum_{i \in \mathcal{T}} x_{ij} \leq 1 & \forall j \in \mathcal{V} & \quad \text{(4.1c)} \\
& x_{ij} \in \mathbb{N} & i \in \mathcal{T}, j \in \mathcal{V} & \quad \text{(4.1d)}
\end{aligned}
$$

In the objective function 4.1a, the number of served requests is maximized while the delay caused by trips is minimized. The delay is multiplied with a factor of 10^{-5} to ensure that the requests served are maximized before the delay is considered. Constraint 4.1b makes sure that a trip is served by at most one vehicle and 4.1c ensures that a vehicle serves at most one trip. The last constraint 4.1d limits the variable to positive integers.

4.1.3. Fleet and Requests update

Another important part of the simulation is that after matching the vehicles and requests realistically progress through the system and produced data is properly stored. This is done by updating the status of the fleet and requests.

A vehicle can acquire the following statuses.

- At origin
- Parked
- Cruising
- Servicing
- Rebalancing

Every vehicle starts at the origin, this status means that it has not been used. It is different from the parked status because it means that the vehicle has been used, but is not at the moment. Both 'At origin' and 'Parked' are considered idle states. When a request is assigned to a vehicle, this vehicle is no longer idle and its status needs to be changed in order to not be included in the next matching round. Depending on the location of the vehicle, the vehicle first needs to cruise to the pick-up location. When the vehicle arrives at the pick-up location, the passengers come on board and the status is changed to 'Servicing'. The vehicle locates itself to the drop-off location and there the service has ended. Depending on what its next orders are, the status changes accordingly. The last status is assigned during rebalancing. It is similar to cruising but instead of a pick-up target, the vehicle is now traveling towards a rebalancing target.

In a similar way, the requests are updated as well. The status of a request can become the following.

- Waiting
- Assigned
- In vehicle

- Serviced
- Rejected

When a request enters the system. Its first status is always 'Waiting'. It means that the request has not yet been assigned and that the passengers are literally waiting to receive an assignment to a vehicle. The waiting status can evolve into two following states. When a successful assignment has been made to a vehicle, the status changes to 'Assigned'. If after multiple iterations still no feasible solution can be found until the point where the time window of the request is violated, the request is rejected. An assigned request still has to wait for the vehicle to arrive at the location. When this has happened, the request is being served and the status changes to 'In vehicle'. It keeps this status until the vehicle has reached the drop-off location. Then, the request is completed and its status can change to 'Serviced'.

The updates of statuses may seem trivial but in the model, this is the backbone of the registration of results and the current state of the system. Important performance aspects like vehicle usage and serviceability can be calculated by using these statuses.

4.1.4. Rebalancing

One form of rebalancing was also already implemented in this model. The method described by Alonso-Mora, Samaranayake, et al. (2017) was implemented where the vehicles would rebalance to locations where rejected requests had occurred. Every timestep, the pick-up locations of rejected requests and the idle vehicles were gathered and inserted in the ILP formulation shown in Section 2.3.2. The rebalancing would take place after the matching phase. If no idle vehicles were available or no rejections had occurred, there was no rebalancing action performed.

4.2. Scenario modification

The provided model had a lot of useful elements, like the ones explained in the earlier sections above. However, in order to make this model simulate the water taxi Rotterdam scenario, a lot needs to be changed as well. Apart from modifications that need to be made to methods, the input data that creates the scenario needs to be generated and implemented first.

As stated in the introduction of this chapter, the network and requests are system-defining elements that are created using external data. A node information file states the node ID and its location in coordinates. This file can be generated by exporting the grid of nodes created in Section 3.5.2. The node ID is changed from an H3 ID to a numerical ID, ranging from 0 to 7840. An adjacency matrix of size 7840x7840 is then generated to define what nodes are connected with a link. The nodes that are connected have a value of 1 in the matrix and otherwise, the value will be 0. The distance matrix is the same size as the adjacency matrix, but as opposed to the adjacency matrix, this matrix is not sparse. For every node combination, the shortest distance is stored on the corresponding element of the matrix. This means this file contains a lot of information and is by far the largest input file. Lastly, the request file needs to be made. This contains every request and its required information, as stated in Section 4.1.1. The file is a compressed version of the dataframe that was used to perform analysis on, explained in Section 3.1. The request file needed to be modified later as well, this will be further elaborated in Section 4.3.1.

4.3. Request handling modification

The change in physical scenario was not the only thing that had to be changed in order for the model to represent the water taxi Rotterdam scenario. The provided model was built around a Mobility on Demand (MoD) system. This means only dynamically appearing requests are considered. In an MoD system, no knowledge of future requests is available so the model will naturally only process requests that are currently ongoing. This also means that requests that appear always want to be picked up as soon as possible, so there is no time for a vehicle to plan ahead. The system is by definition reactive.

The scenario involving the water taxis in Rotterdam, however, is not an MoD system. Here, requests can be made hours or even days ahead. Having knowledge of future requests on which an optimal planning can be made, drastically changes the outcome compared to a reactive MoD system. If all requests were known at the beginning of a day, an optimal planning could be made once, and the request would be served optimally. The difficulty of the water taxi scenario is that it features mixed requests: planned as well as dynamic requests. Planned reservations have to be included while on the other side, there should be room for dynamically appearing requests as well. Creating an entirely new planning with every new request appearing would computationally be very expensive and creating a new planning every certain time interval would exclude dynamic requests that have a pick-up time within that time interval.

Eventually, the compromise was made that a large part of the MoD system would stay in place, but with the addition of a short-term future planning. The usage of the MoD system means that an already good working method would not have to be completely rewritten. The addition of the short-term future planning means that some of the benefits of having knowledge of future requests could be obtained, while not paying the expense of constructing a completely new planning every timestep.

4.3.1. Reveal time

In order to facilitate the development towards a mixed request system, a new attribute should be added to requests. Previously, the requests were revealed to the system once the pick-up time had passed. The time the request was revealed, and the time the request wanted to be picked up were identical. Now, a distinction needs to be made between the time a request is revealed and the pick-up time. This is where the reveal time as a new attribute comes in. In the water taxi data provided by flying fish, the first interaction a user makes with the system is recorded. This time can be used as the reveal time for the simulation. The reveal time should always be earlier or equal to the pick-up time. The addition of the reveal time to the input request data is as simple as creating an extra column in the CSV file with the corresponding reveal time for every request.

4.3.2. Mixed Request handling

With the addition of the reveal time, the handling of the requests can now be altered. Instead of creating a batch of requests based on pick-up time. A batch is now created and introduced to the system according to their reveal time. But if only that is done, a large number of requests would always be in the system. This is caused by the large number of requests that are revealed hours or days before their pick-up time. We do not want to overload the system with requests that will not start in the near future. So a part of the request needs to be stored somewhere so that it can be introduced to the system at the appropriate time. The way that this is done, is visualized in Figure 4.5.

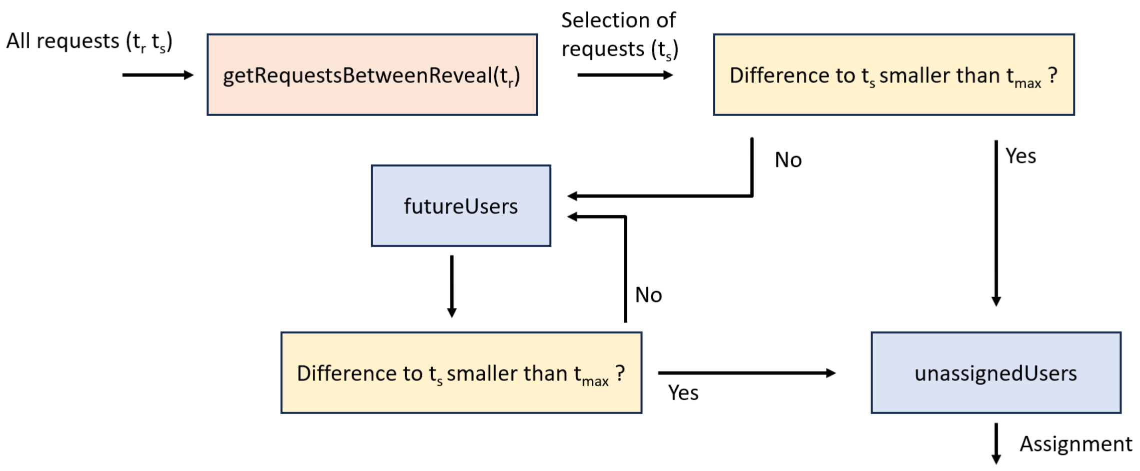

All requests (t_r t_s)
getRequestsBetweenReveal(t_r)
Selection of requests (t_s)
Difference to t_s smaller than t_{max} ?
No
Yes
futureUsers
No
Difference to t_s smaller than t_{max} ?
Yes
unassignedUsers
Assignment

Figure 4.5.: A modification on the method for generation of requests, where a distinction is made between current and future requests.

When a batch is created using the reveal time, the mixed request handling method differentiates the requests according to the duration until the pick-up time of the request. If this duration until pickup is larger than the threshold t_{max}, the request will be temporarily stored in a list called *futureUsers*. If the duration is shorter than t_{max}, the request gets directly put in the list *unassignedUsers* meaning that these requests will be included in the next matching round. The requests in the *futureUsers* list will be checked every timestep to see if they are suitable to be put in the *unassignedUsers* as well. The value of t_{max} has been set to 15 minutes because this limits computational efforts in the matching phase. However, a time advantage of 15 minutes before a request's pickup time already makes a great improvement.

4.3.3. Time windows

In Section 4.1.2 it was explained that an RTV graph takes the time windows of the requests into account. The time windows have a lot of influence on the performance of the system because they can make the problem harder or easier to solve. Large time windows create an environment where users are very flexible. Large delays are allowed so there is more room for vessels to take an optimal route. Tight time windows give less of this flexibility and decrease the delays recorded in the simulation.

When dealing with dynamic and planned requests, a distinction in time windows should be made. A planned request that has been confirmed by both sides and known some time beforehand, should not have the same allowed delay as a request wanting to be picked up right now. For a planned request, a pick-up delay of more than 3 minutes is really not desirable. For a dynamic request, getting to the location may even take more than that. If differentiation is made between the time windows of the types of requests, a low delay can be ensured for planned requests, while preventing dynamic requests from being rejected.

To assign the time windows to the different types of requests, a clear distinction needs to be made between the types as well. A threshold can be set into place to achieve this. When the duration until the pick-up time of the request is below this threshold, the request is considered dynamic. A duration above the threshold will result in a planned request. When this distinction is made, the corresponding time windows can be applied. In Section 4.5, the values for these time windows and threshold are specified.

4.4. Matching modification

The benefits of revealing future requests are not achieved if the matching phase cannot handle these future requests. Modification is needed for the method to make future assignments. If future requests are handled in an unmodified way, the RTV graph will not find a valid trip option until the point in time when cruising to that location will be just on time, or too late. The request will just get cycled through the matching phase until time window constraints are satisfied. Furthermore, improvements can be made to the objective function of the assignment ILP. Additions to this objective function give the opportunity to lay more emphasis on certain performance factors. Both subjects will be further explained in this section.

4.4.1. Future assignment

The scheduling of potential trips is determined in the RTV graph. With the current time as an input parameter, the RTV graph method will check the feasibility of the combinations of vehicles and requests. The time a vehicle leaves for a pick-up location has a great influence on whether the vehicle will arrive within the time window. Therefore, when constructing the RTV graph, it is assumed that the vehicle leaves instantly for the pick-up location. This assumption is true in a MoD scenario, but not in the mixed requests scenario. If a vehicle is assigned to a future pick-up, it will not always leave instantly. In order to facilitate assignments in the future, but still allow the RTV graph to check time windows for an assignment in the future, future RTV graphs need to be constructed. If RTV graphs for future time instances are created, the system is basically checking what trips are possible in future timesteps. These graphs can then be combined into one single graph, that contains trips for current and future assignments. The new matching method is visualized in Figure 4.6.

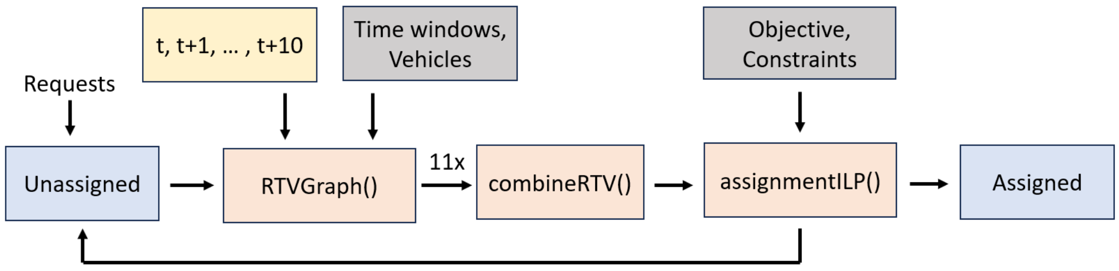


Figure 4.6.: A modification on the matching method. Here, future assignment is possible by providing future RTV graphs and combining them into one.

The assignment ILP still decides what trips result in the best output, but now it makes the decision over multiple timesteps. Because the RTV generation is computationally relatively expensive, it is decided to not look further into the future than 10 minutes. This new matching method means that vehicles can be assigned to requests up to 15 minutes beforehand. In most situations that is enough time to reach the destination and have spare time as well. The distribution of that spare time is where waiting strategies come in. Without this future assignment, the application of waiting strategies would not be possible.

4.4.2. ILP objective

With the addition of future assignments, the simulation model gets closer to the real-life scenario of the water taxis in Rotterdam. However, a few more improvements can be made in terms of the results of the simulation. One thing that stood out was that the number of shared requests was relatively low compared to the observed numbers in the data analysis. After investigation, it became clear that the problem may lie in the formulation of the objective function. The original objective function is displayed in Equation 4.2.

$$\text{maximize: } \sum_{i \in \mathcal{T}} r_i x_{ij} - \sum_{i \in \mathcal{T}, j \in \mathcal{V}} c^d \cdot d_{ij} x_{ij} \tag{4.2}$$

From Section 4.1.2 we learned that this objective function maximizes the number of served requests while minimizing the amount of delay produced. And it does its job very well. From Table 4.1 can be read that only an average delay of about 4 seconds was achieved while serving all requests. However, the objective is not very inclined towards ride-sharing. Ride-sharing does not have a very positive influence on delay and that is the only thing that is minimized by the objective. Something that is positively influenced by ride-sharing is distance minimization. When a vehicle can serve multiple requests in one trip this saves distance. In order to motivate the system to share more rides, this distance minimization should also be included. The formulation of the new objective function is shown in Equation 4.3.

$$\text{maximize: } \sum_{i \in \mathcal{T}} r_i x_{ij} - \sum_{i \in \mathcal{T}, j \in \mathcal{V}} c^d \cdot d_{ij} x_{ij} - \sum_{i \in \mathcal{T}, j \in \mathcal{V}} c^s \cdot s_{ij} x_{ij} \tag{4.3}$$

In this objective function, the amount of distance s_{ij} that a vehicle j travels when serving trip i is added to the minimization of the delay. It is multiplied by the parameter c^s which determines the weight of the distance part of the objective function. This weight has to be balanced with the weight for the delay minimization c^s. The result of the simulation with added distance minimization is shown in Table 4.1.

Table 4.1.: The results of the distance minimization tests. Where the request and delay objective is compared with the request, delay, and distance objective.

Objective	**Avg. pick-up delay [s]**	**Served requests**	**Shared requests**	**Distance empty [km]**	**Distance loaded [km]**
Request & delay	4.11	155	12	837	441
Requests, delay & distance	8.94	155	33	290	416

The results show a large increase in shared requests and a decrease in traveled distance. As a consequence, the average pick-up delay rises. This was expected since it is not the only aspect being minimized anymore.

Another aspect that has not been considered before is the usage of vessels. In the results shown in Table 4.1, all vessels are being used every time. This is not realistic because the data never shows that high number of vessel usage. The reason for all vessels to be used is because there is no cost associated with it. Just like the cost per meter traveled or per second of delay, there should also be a cost for the usage of an extra vehicle. This can again be accommodated by a modification of the objective function, as shown in Equation 4.4.

$$\text{maximize: } \sum_{i\in\mathcal{T}} r_i x_{ij} - \sum_{i\in\mathcal{T}, j\in\mathcal{V}} c^d \cdot d_{ij} x_{ij} - \sum_{i\in\mathcal{T}, j\in\mathcal{V}} c^s \cdot s_{ij} x_{ij} - \sum_{j\in\mathcal{V}} c^v \cdot v_j x_{ij} \tag{4.4}$$

In this objective function, the usage of extra vehicles is penalized. Before the execution of the ILP, the currently used vehicles are gathered and stored in a list. The variable v_j becomes 1 if the vehicle used in the assignment x_{ij} is not present in the used vehicle list. In this way, the penalty in the objective function is only included when a new vehicle is used. The size of the penalty can be determined by the parameter c^v. With an increasing parameter value, the number of used vehicles will decrease. The result of this addition to the objective function is shown in Table 4.2.

Table 4.2.: The results of the tests performed with different objectives. Where the request, delay, and distance objective is compared with the request, delay, distance, and vehicles objective.

Objective	**Avg. pick-up delay [s]**	**Served requests**	**Shared requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Requests, delay & distance	8.9	155	33	18	290	416
Requests, delay, distance, vehicles	18.2	155	47	8	341	412

The results show a clear drop in used vessels, as expected. As a result of this extra factor in the objective function, the distance traveled and pick-up delay increase again. Fewer vehicles will have to travel more distance to service the same amount of requests, and because the vehicles follow a tighter schedule, delays will also increase. The penalty on extra used vehicles also increases the number of rides shared. The reason for that is that ride-sharing now has a double benefit: less distance traveled as well as fewer vehicles used for the same number of trips.

4.5. Base case

In order to do experiments with different strategies and analyze how the results are affected, a clear benchmark needs to be set. This benchmark will be defined as the base case. This is the case where no extra strategies are applied to the model. The settings of this base case will be conserved through the entire testing phase. The specified strategy will be the only alteration of this case. The settings of the base case are summarized in Table 4.3.

By definition of the problem, there are 18 vessels available and the vessels have a capacity of 12 persons. The speed of the vessels is around 50 km/h. In the model, it is however reduced to 47 km/h to account for docking as well. Furthermore, in these settings, the dynamic request threshold is 5 minutes. This means that a request is considered dynamic if the pick-up time is less or equal to 5 minutes. A larger time window of 15 minutes for dynamic requests is defined,

Table 4.3.: The settings of the simulation model that define the base case.

Model setting	**Value**
Available vessels	18
Vessel capacity	12 pers.
Vessel speed	47 km/h
Dynamic request threshold	5 min
Dynamic request time window	15 min
Planned request time window	3 min
Planning horizon t_{max}	15 min
Number of RTV graphs	10
Objective function parameters:	
Requests	1
Delay	$-2.2e^{-5}$
Distance	$-2.3e^{-6}$
Vehicles	$-8e^{-3}$
Rebalancing strategy	None
Waiting strategy	Drive First

compared to 3 minutes for planned requests. As discussed in Section 4.4.1, due to computational limitations the number of RTV graphs is set to 10. Having a request revealed earlier than 15 minutes is not necessary. The objective function parameters are defined in a way so that plausible results are generated that are comparable with the results found in data analysis. Drive First is defined as the base waiting strategy and no rebalancing strategy is applied. The results of the base case are already shown in Table 4.2, where the objective function includes vehicle usage as well.

5. Waiting strategies

As stated in Section 2.2, waiting strategies aim to optimally distribute waiting time to a schedule. It does this by ordering the vehicle to wait at certain places and not at others. Literature can tell us some things, but the potential of these waiting strategies can only be fully validated by testing them in the simulation. In this chapter, it will first be explained how waiting is implemented into the simulation model. This is followed by the implementations and results of the different waiting strategies that are tested.

5.1. Implementation of waiting

Before any strategies can be implemented, the concept of waiting needs to be introduced to the model. Because the model previously served as a MoD simulator, waiting while a request is assigned is never done by the model.

After analysis of the model, it was decided that the best place to insert a waiting method is after the assignment and right before the fleet and request update. This is because an assignment is needed to determine if waiting is preferred, and it should happen before any route updates because otherwise, the vehicle would have already moved. The insertion of a waiting method is visualized in Figure 5.1.

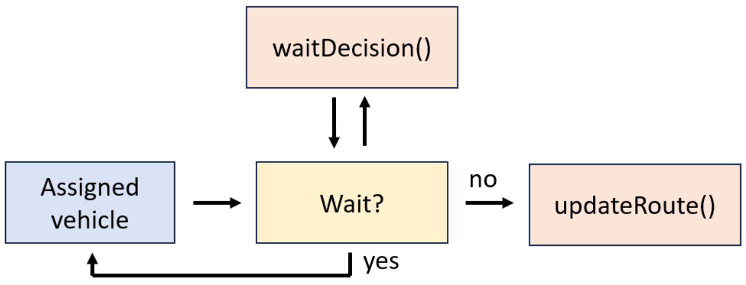


Figure 5.1.: The implementation of waiting on the model. A waiting method is inserted between the assignment of vehicles and route updates.

Here, the simulation model will only execute the route update if the vehicle does not have to wait. The route update method reads the path of the assigned vehicles and checks if vehicles have arrived at their target. When a node in their path is reached, it updates the vehicle's status accordingly and assigns a new target. The wait decision method decides if the vehicle can start with its route or not. By doing it in this way, the vehicle stays assigned but is not moving.

The assignment of the vehicle is not static. Throughout the whole simulation, if a vehicle is not servicing a customer, it can be reassigned if another better assignment appears. This is done by inserting the vehicles that are not performing service into the assignment. The same happens for requests that are assigned but not picked up. They are inserted into the RTV graph and ILP as if the previous assignment did not happen. If a better solution appears than the previous one, the assignment will be altered. This is why waiting could make a difference in some places. The distribution of vehicles is altered by a waiting strategy and this will result in different results.

5.2. Wait First

As we know from Section 2.2.2, Wait First will always try to wait whenever possible. In the model, this means that when a vehicle is assigned, it is checked if this vehicle is able t wait or not. A difference with the theoretical strategy, it is decided that the vehicle should wait until it can reach the opening of the time window instead of the last possible time. Otherwise, passengers would always have to wait the maximum amount which is of course not desirable. Wait First in this context means that the vehicle should wait until it reaches the opening of the time window just in time.

The main part of the Wait First strategy is the calculation of the departure time. It uses the earliest pick-up time and the time needed to travel to the location. The calculation is shown in Equation 5.1.

$$\text{Latest departure} = \text{Earliest pick-up} - \text{Duration} - 1 \text{ minute} \tag{5.1}$$

An extra minute is added as a buffer to make sure the vehicle will not arrive too late. This is because the simulation runs with timesteps of 1 minute. This means it is checked every minute if the latest departure has passed. If the latest departure had passed without an extra minute added, the vehicle would arrive too late. The extra minute makes sure the vehicle arrives in an interval of 1 minute too early and just in time. A visualization of the Wait First implementation of the model is displayed in Figure 5.2.

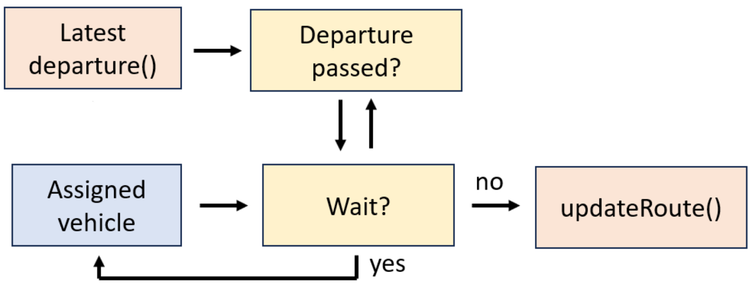


Figure 5.2.: Visualization of the waiting strategy implemented in the model.

5.3. Intensity

The intensity waiting strategy may be fully defined in the literature, but implementing this extensive strategy still has its challenges. First, the right information needs to be extracted from the simulation to serve as input for the strategy. Additionally, the strategy in literature assumes full knowledge about the entire planning is available, which is not the case in the simulation. Also, the tuning of this strategy plays a large role in its performance. Solutions found for these challenges will be presented in this section, starting with an in-depth analysis of the intensity measure.

5.3.1. The intensity measure

The intensity measure is really beneficial in the water taxi Rotterdam scenario. As stated in Equation 2.2. The intensity is defined by the average transition time for unrevealed requests in the historical set S. To understand what the intensity actually means, an example is created in Figure 5.3.

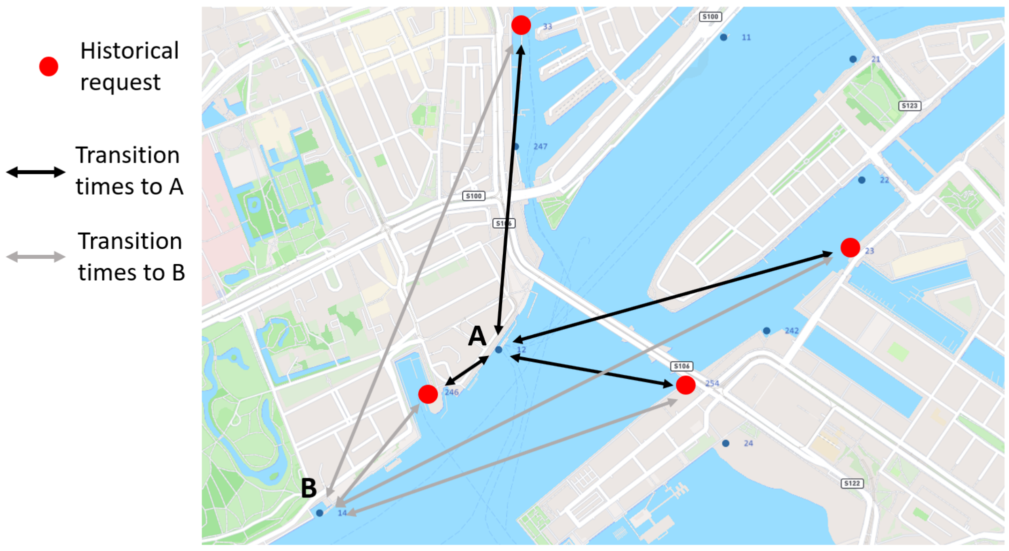


Figure 5.3.: Intensity explained by an example were the intensity of location A and location B are analyzed.

The figure shows a location A and a location B. For both locations, the intensity will be analyzed. There are four historical requests present in this example. The transition times are shown as arrows. It is clear that location A has shorter transition times to the historical requests. This is because that location has a more central position to the requests. Location B lies further away so the average transition time is longer. Because of this A has a higher intensity than B. In a way, the intensity measure will promote the locations that are the most central compared to the historical request available.

An important thing to note is that the intensity measure also has a planning horizon. In this case, the horizon is set to one hour. This means that requests that are recorded from the current time of the simulation until one hour into the future will be included. This makes the measure also time-dependent. In the morning, different requests may appear than in the afternoon for instance. In the end, the intensity measure is a way of valuing the centrality of a location compared to historical requests in the coming hour. Together with the slack time and the transition time to that location, a decision on waiting can be calculated.

5.3.2. Wait decision

The total value of a location is based on the intensity, the slack time, and the transition time to the location, as stated in Equation 2.3. How large of a role these factors play in determining the value is based on 3 tuning parameters. The way the value function is calculated there is directly used in this implementation of the simulation model. The way the waiting time is calculated, however, is different.

The waiting time calculation in Equation 2.4 will not be used in this implementation. That is because in the waiting time calculation, Vonolfen and Affenzeller (2016) assume that the entire planning is known. So the available total slack time can be distributed over multiple instances. In the water taxi case, only the next location is known. Using the same calculation as Equation 2.4, would result in very small and unrealistic wait times. Instead, a threshold-based wait decision is created. If the value of waiting is above the threshold, the vehicle waits as long as possible. Basically, it switches to WF for a moment. So the latest departure is calculated again and handled like explained in Section 5.2. The updated method for the intensity wait strategy is shown in figure Figure 5.4.

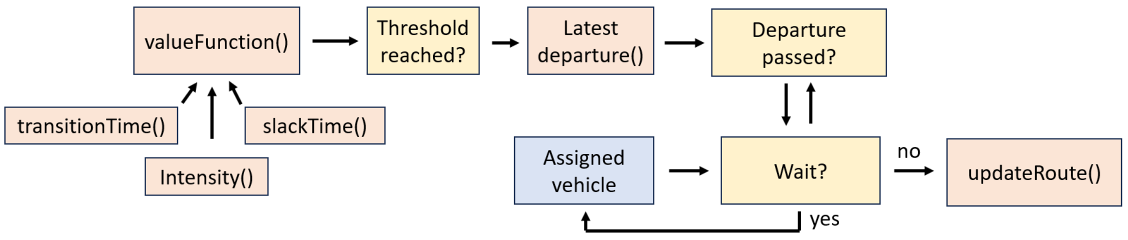


Figure 5.4.: The modified procedure for the intensity waiting strategy.

5.3.3. Tuning

The value function adds 3 tuning parameters in the form of weights that influence the importance of each part in the equation. On top of that, another tuning parameter is added in the form of the threshold that the value must surpass in order to trigger a waiting action. All of these parameters can be tuned to any number between 0 and 1. Tuning these parameters individually is not an option because the parameters are connected and do not function independently. The possible combinations of these 4 parameters and their results need to be investigated. 3 types of the most used tuning algorithms (Petro Liashchynskyi and Pavlo Liashchynskyi, 2019) are analyzed: grid search, random search, and the genetic or evolutionary algorithm.

Grid search The most structured type of parameter tuning is grid search. The combinations of parameters will be placed in a grid with a preferred precision. In Figure 5.5, a 2-dimensional grid is displayed. In this case, a four 4-dimensional grid would be needed. For every parameter, an evenly spaced range from 0 to 1 would be needed. The 4 ranges of the parameters then have to be combined to return all unique combinations. The advantage of this algorithm is that the complete solution space is searched evenly. And with a small enough precision, one could state that the entire solution space can be searched this way. A disadvantage of this method is that it takes a lot of samples to get an accurate result. A fine grid is needed for this to capture the different possible outcomes. Even if the parameters are increased by steps of 0.1, the amount of samples needed for that grid is $10^4 = 10.000$. Given that one sample would take 5 minutes of computation time, this very coarse grid would take more than a month to complete.

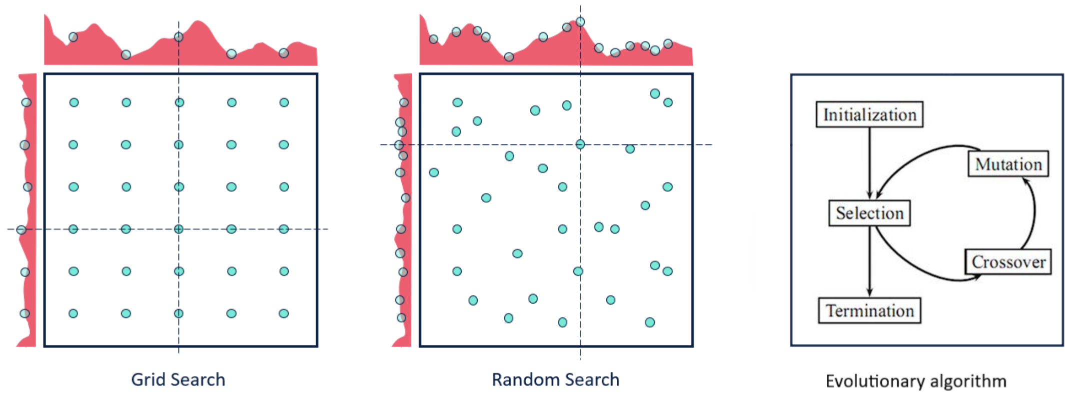


Figure 5.5.: Visualization of the different tuning algorithms.

Random search A way to deal with otherwise very long tuning durations is, for instance, by using random search. Instead of fixing the parameter options to a grid, the values are assigned randomly, with an upper and lower bound. In general, this leads to faster acceptable solutions (Zhigljavsky, 2012), but at the cost of less certainty that the most optimal parameter is found. In Figure 5.5 is illustrated how with the same low number of samples, a better solution can be found because of the randomization of parameters. Also, this algorithm allows one to search until a certain number of samples is reached, instead of completing a whole grid. In the current situation with relatively large run times, this extra convenience plays a role as well.

Evolutionary algorithm The last method is an evolutionary algorithm. As the name implies, this algorithm is derived from evolution in nature. The algorithm operates on the idea of natural selection. That a species will pass the genetics to a successive generation that provides the largest probability of survival. Similarly, an algorithm with the right settings can provide generations of solutions that will ultimately converge to an optimal solution. First, an initial population is needed. The best results from the population will be passed on to the next generation. Additionally, the best solutions can be recombined into new solutions, this is also called crossover. Also, a selection of solutions can have a randomly altered parameter in their solution as well, creating a new input. This is called a mutation. The size of the populations, the number of selected solutions, and the amount of crossover and mutation all play an important role in the efficiency of this algorithm. Even though this algorithm could outperform both grid search and random search, it either needs a large population or a lot of generations to converge to a solution (Mirjalili, 2019).

Because of the large number of samples needed for grid search and the evolutionary algorithm. The random search method is chosen as the tuning algorithm for this case. The implementation is shown in Figure 5.6. For every iteration, new random parameters are generated. Those parameters are inserted into the model and the model runs the simulation. The simulation results are used in a score calculation. The score is calculated in the same way as the objective function in Equation 4.4. The objective parameters from Table 4.3 are used. If this score is higher than the current best score, the best score is replaced. Otherwise, it is added to the other scores. The best parameters as a result of this tuning algorithm are shown in Table 5.1

The parameter controlling the portion of the transition time of the value α is set to a medium amount. This means the transition time is taken into account but has not too much influence on the value of the location. The intensity of the location, on the other hand, is given a lot of influence

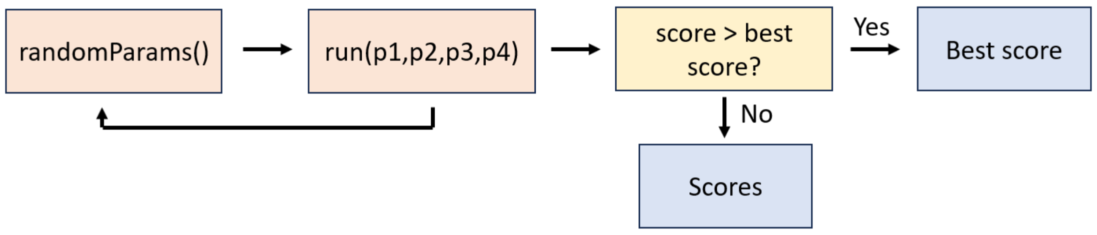


Figure 5.6.: The implementation of the random search algorithm to the simulation model.

Parameters	α	β	γ	ϵ
Values	0.385	0.891	0.064	0.589

Table 5.1.: The values of the parameters for the intensity waiting strategy, resulting from the random search tuning algorithm.

since its parameter β is very high. The parameter for the slack time γ is almost zero, meaning it has very little influence. The value of threshold ϵ is balanced. A higher threshold would lean more towards Drive First and a lower threshold would go in the direction of Wait First.

5.4. Results

Table 5.2.: The results of the waiting strategies

Waiting strategy	**Avg. pick-up delay [s]**	**Served requests**	**Shared requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Drive First	18.2	155	47	8	341	412
Wait first	32.5	155	65	9	215	408
Intensity	19.2	155	47	7	340	410

The results in Table 5.2 show the performance indicators of the simulations executed with the different waiting strategies. Drive First is the base case. It has the lowest average pick-up delay of all waiting strategies. On the other hand, it also results in the most distance traveled, which makes sense. When a vehicle always drives to the next location without thinking, it is inevitable that in some cases more distance is traveled than needed. Waiting naturally does not produce traveled distance, making this a more conservative approach in terms of distance. Wait First, however, does perform the least of all waiting strategies overall. It does perform the best in the traveled distance, but the need for an extra vehicle and the large increase in pick-up delay is not good for performance. It needs an extra vehicle because waiting at all times causes the vehicle distribution to be different. Scenarios may occur where all vehicles are waiting in remote locations, causing the usage of an extra vehicle. On top of that, it also causes more pick-up delay. In this scenario, always waiting results in more shared requests, contributing to the low distance traveled as well, but increasing the pick-up delay. The Intensity waiting strategy finds a middle ground between these two strategies for some performance factors. It still has a higher average delay than Drive First but not as high as Wait First. Also, the number of shared requests is identical to Drive First. Where the intensity stands out is the number of vehicles used. By waiting at the right locations it was able to reduce to 7 vehicles used. And even with that decrease in used vehicles, the amount of distance traveled is slightly reduced as well. Overall, this strategy scores the best and the increase in delay would in most cases outweigh the decrease in the use of vehicles and distance traveled.

6. Rebalancing strategies

As we know from the literature in Section 2.3, rebalancing is defined as the movement of idle vehicles to desired places of interest. It can reduce waiting times or fleet size by already being in places where requests emerge. This method can also be applied in different ways in the water taxi scenario. The knowledge of historical requests can help with determining the right rebalancing targets. This could be done by always rebalancing to a popular place, or rebalancing to locations where delays have occurred. More sophisticated methods could also be used to try and predict future request locations or generally popular locations and rebalance towards them. How these different techniques are implemented into the simulation model and how they perform, will be evaluated in this chapter.

6.1. Rebalance to location

Rebalancing in its simplest form is taking a popular location and making sure vehicles rebalance to that location when needed. It is a variant of the "rebalance to desired distribution" method elaborated by Pavone et al. (2012). Here the desired distribution is based on the fact that the most popular location should have a vehicle present at all times. A rebalancing action is very beneficial when it leads to a pick-up at the intended target with the rebalanced vehicle in a short amount of time. It can also be beneficial if the rebalancing action has caused a vehicle to have already moved closer to a pick-up location. Having vehicles rebalancing to the most popular location has the highest probability of a successful rebalancing action. This method requires a minimal amount of data analysis, only popular locations have to be extracted but further calculations are not necessary.

Rebalancing is implemented into the model right after the main assignment and before the fleet and requests update. The rebalancing assignment is an addition to the main assignment, where idle vehicles can be given a task if necessary. The procedure for rebalancing is displayed in Figure 6.1. First, a list of potential targets is defined. This is in this case the most popular location. Another list is created that contains the location ID of every location where a vehicle is located or will be in the future. Both of the lists are compared with each other. If a vehicle is already

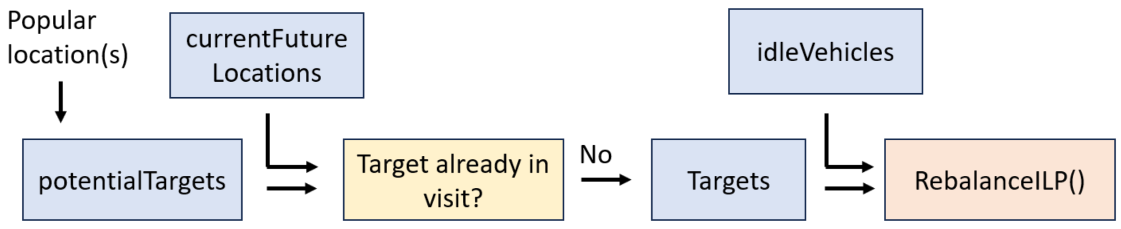


Figure 6.1.: The procedure for rebalancing to popular locations. It shows how potential rebalancing targets are filtered and inserted into the rebalancing ILP.

located on a potential rebalancing target, it would not make sense to send another one. Also if a vehicle has the potential target in its path already, sending another vehicle to it would result in 2 vehicles arriving there in the future. Therefore, it must be checked if a potential target is not already in the *currentFutureLocations* list. When it is not, the potential target becomes an actual target and serves together with the idle vehicles as input for the rebalancing ILP. The rebalancing ILP is very similar to the main assignment ILP. It is shown in Equation 6.1. The input $\mathcal{V}_{\text{idle}}$ is the set of idle vehicles and $\mathcal{T}$ is the set of targets as defined earlier. The ILP will find the optimal assignment where the least distance is traveled while rebalancing to all targets. Equation 6.1d in particular makes sure that the number of rebalancing actions is equal to the number of targets or the number of vehicles. The other constraints work the same as the main assignment ILP in Equation 4.1

$$\text{minimize} \sum_{i\in\mathcal{V}_{\text{idle}}, j\in\mathcal{T}} c^x_{ij} x_{ij} \tag{6.1a}$$

$$\text{subject to:} \sum_{i\in\mathcal{V}_{\text{idle}}} x_{ij} \leq 1 \qquad \forall j \in \mathcal{T} \tag{6.1b}$$

$$\sum_{j\in\mathcal{T}} x_{ij} \leq 1 \qquad \forall i \in \mathcal{V}_{\text{idle}} \tag{6.1c}$$

$$\sum_{v\in\mathcal{V}_{\text{idle}}} \sum_{r\in\mathcal{T}} x_{ij} = \min(|\mathcal{V}_{\text{idle}}|, |\mathcal{T}|) \tag{6.1d}$$

$$x_{ij} \in \mathbb{N} \qquad \forall i \in \mathcal{V}_{\text{idle}}, j \in \mathcal{T} \tag{6.1e}$$

The amount of targets can be an interesting parameter to investigate. Instead of rebalancing to the most popular location, a selection of the most popular locations can also be targeted. This increases the number of locations where potential requests can be served directly. It does however also increase the number of rebalancing actions and with that the traveled distance and the number of vehicles needed. To get more insight on the trade-off, tests with an increasing number of rebalancing targets were performed. The results are shown in Table 6.1. The 5 most popular locations of the water taxi scenario in Rotterdam are the following:

1. Hotel New York (terminal)
2. Leuvenhaven (center)
3. SS Rotterdam
4. Boompjes
5. Euromast

The number of times the locations are used as pick-up or drop-off location is shown in Figure 6.2. By far, the most requests emerge from the terminal at Hotel New York. The second most popular location is the dock that is closest to the city center. A famous ship that is now a Hotel and Restaurant is in third place and the last two locations are other places of interest in around the center of Rotterdam.

As expected, the number of rebalances increases with every target location added. The amount of distance traveled follows the same trend because more rebalancing generally means more distance traveled when the vehicle is empty. With 2 rebalancing targets, the vehicle distribution is so beneficial that only 7 vehicles are needed to fulfill all requests. When rebalancing to only the second most popular location, the location closest to the center, this decrease in used vehicles is observed as well. However, by not including the most popular location, the pick-up delay increases. Interestingly, the average pick-up delay does not decline steadily. When the 5 targets

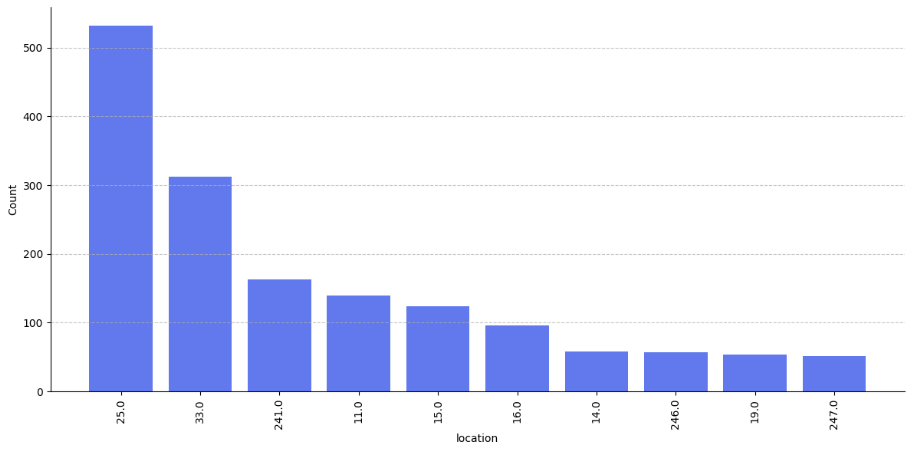


Figure 6.2.: The most visited docks of the water taxis in Rotterdam. The first 5 bars in the chart correspond to the 5 most used locations.

Table 6.1.: The results of the tests of inserting more rebalancing target locations. The rebalances performed column is added for more insight.

Targets	**Reb. performed**	**Avg. pick-up delay [s]**	**Served requests**	**Shared requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base	-	18.2	155	47	8	341	412
Top 1	33	17.9	155	47	8	348	409
Top 2	73	19.1	155	46	7	422	410
Top 3	106	16.2	155	46	8	531	410
Top 4	187	14.2	155	40	9	623	410
Top 5	250	15.6	155	39	10	717	409
Centre only	41	19.4	155	42	7	410	411

are selected, the delay experiences an increase in value. This is due to the fact that a rebalancing action does not mean that that action is successful. If a rebalancing action fails it can lead to more delay on top of the extra distance driven. If more rebalances are done, the chances of rebalancing actions being successful do not increase. In fact, if the added rebalancing actions are linked to less popular locations, the chances of a successful rebalancing action decline. Performing only a selected amount of rebalances can therefore have a better result than performing the most actions possible.

6.2. Rebalance based on delay

Another interesting strategy is the rebalancing strategy proposed by Alonso-Mora, Samaranayake, et al. (2017). They identify rebalancing targets not by popularity, but by the system performance per location. In their work, the pick-up location of every previous request that was rejected is turned into a rebalancing target. This made sure that locations that are hard to reach or usually do not fit into the schedule have a vehicle nearby. In the water taxi Rotterdam scenario, however, there are no rejected requests in the base case. So implementing this strategy would result in no rebalancing actions. Instead, another performance factor is used: the delay. A location performs badly when the request is eventually completed with a delay. These are the locations that are hard to reach or do not fit into the schedule. Rebalancing to these locations could prevent further delays in the future. The procedure for the rebalancing strategy is similar to the rebalance to location strategy shown in Figure 6.1. The ILP used is the same as the previous rebalancing ILP defined in Equation 6.1.

6.2.1. Delay threshold

A way the strategy can be influenced is by using a delay threshold. Of course, it is possible to rebalance to all locations that cause delays. But maybe it might be better to create a selection of locations where the delay is higher than a certain threshold. But to find the right threshold, tests need to be done to investigate the influence. The delay thresholds that are used are 0, 30, 60, 100, and 150 seconds. The results are shown in Table 6.2.

Table 6.2.: The results of the rebalancing based on delay tests for using different delay thresholds.

Threshold [s]	Reb. performed	Avg. pick-up delay [s]	Served requests	Shared requests	Vessels used	Distance empty [km]	Distance loaded [km]
Base	-	18.2	155	47	8	341	412
0	11	20.6	155	48	8	351	410
30	10	20.6	155	48	8	344	410
60	8	19.0	155	46	8	361	409
100	5	21.5	155	49	8	358	410
150	2	19.7	155	48	8	346	410

The delay rebalancing strategy seems to not perform as well as the location rebalancing strategy. On the positive side, the number of vehicles used does not increase. This is largely because the number of rebalances is very low. Even with a threshold of 0, meaning that all locations that cause delay are targeted, the number of rebalancing actions is only 11. This has two reasons. The first one is the simple fact that the number of trips that are delayed is not very high. In most cases, the number of delayed trips is around 30. Having this low number of potential rebalancing actions means that the actual rebalancing actions will also never be high. Secondly, it has to be noted that a potential rebalancing action is only selected if there is no vehicle already at the location. The terminal is often the source of a delay because it is the most busy. But because it is busy, a lot of times there will be a vehicle traveling to the location already. This explains the low number of rebalancing actions. On top of the low number of rebalancing actions, the actions performed are not very successful. Hence the increase in average pick-up delay.

6.3. Rebalance based on forecast

The first rebalancing strategy that uses historical requests to generate a target is a variant of the method proposed by Iglesias et al. (2017). Their method uses the requests for prediction of future requests. The predictions are then used to form rebalancing targets. In theory, this strategy should perform better than the previous strategies because it utilizes important information to base its decisions on. If the variant of this strategy also performs better in the water taxi scenario, will be tested in this section. The strategy requires additional effort before it can be tested. A prediction model needs to be in place as this strategy requires predicted requests. The strategy has its own ILP that needs to be set up. Also, parameters need to be tuned to get the best results.

6.3.1. Prediction

For the prediction of future requests Iglesias et al. (2017) use a Long Short Term Memory (LSTM) neural network. The neural network is trained on historical data and will output predicted requests. Because the construction and training of a neural network is out of the scope of this project, a prediction model needs to be imitated in order to test this rebalancing strategy. The simulation model normally has no knowledge of the request list for the full day, although it is present. It is used to pull the batches of requests and those are revealed to the system according to their reveal time. Instead of a prediction model, some of those requests can be pulled from the list and inserted as a prediction. In this way, an amount of future requests is revealed to the system. Then the strategy makes the decision if the location is turned into a rebalancing target or not.

Conservative assumptions need to be made to construct the prediction model. The two factors that are important are the percentage of future requests to be predicted and the prediction horizon. A prediction of 100% would mean that all future requests are being predicted and 0% means that no prediction is made. Eventually, the prediction percentage is set to 50%. In Section 7.7 this number is further explained and justified. The prediction horizon does not have to be long. In order to create a successful rebalancing action the action should not take place too long before the pick-up time. Otherwise, the vehicle would be doing another task by the time the pick-up takes place. Multiple prediction horizons will be tested to confirm this.

6.3.2. Rebalancing ILP

In addition to the prediction model, the ILP for rebalancing assignment needs to be updated as well. The ILP will be a bit different than the one from the literature in Equation 2.8. There are no waiting requests, no total demand, and no request assignment in the water taxi case. This is because pre-processing steps are done that are not done in the literature. The ILP is post-matching, so everything that relates to the request assignment can be ignored. The literature ILP is combined with the current rebalancing ILP and the result is shown in Equation 6.2.

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in \mathcal{V}, j \in \mathcal{T}} c^x_{ij} x_{ij} + \sum_{j \in \mathcal{T}} c^d d_j && && \text{(6.2a)}\\
\text{subject to:} \quad & \sum_{i \in \mathcal{V}} x_{ij} \leq 1 && \forall j \in \mathcal{T} && \text{(6.2b)}\\
& \sum_{j \in \mathcal{T}} x_{ij} \leq 1 && \forall i \in \mathcal{V} && \text{(6.2c)}\\
& \sum_{i \in \mathcal{V}} x_{ij} + d_j = 1 && \forall j \in \mathcal{T} && \text{(6.2d)}\\
& x_{ij}, d_j \in \mathbb{N} && \forall i \in \mathcal{V}, j \in \mathcal{T} && \text{(6.2e)}
\end{aligned}$$

What makes this different from the previous rebalancing ILP is that not every target needs to be served now. Instead, the target can be served or denied. In the objective function 6.2a the distance traveled and the denied targets are minimalized. The parameter c^d is added to add a weight to a denied request. Additionally, constraint 6.2d is added to make sure that a target is either served or denied. The rest of the constraints are the same as Equation 6.1. The extra parameter can be tuned as well. This will be done in the next chapter.

6.3.3. Tuning

There are 2 parameters that can be tested to see what gives the best results. The prediction horizon and the rejection distance. The prediction horizon tells something about how far ahead in the future a request will be predicted and provided to the system. The rejection distance is the minimum distance from the vehicle to the target where the rebalancing ILP will choose to deny that rebalancing target. The results for tests with the prediction horizon are shown in Table 6.3

Table 6.3.: The results of the rebalancing based on forecast tests with a different prediction horizon.

Horizon [km]	**Reb. performed**	**Avg. pick-up delay [s]**	**Served requests**	**Shared requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base	-	18.2	155	47	8	341	412
15	36	18.6	155	44	9	357	415
30	75	17.6	155	46	9	387	410
45	113	17.5	155	46	9	407	407
60	185	15.0	155	46	10	477	410

Around 30 minutes is a sweet spot where the average delay is low for the amount of distance traveled. Compared to a 45-minute horizon, the increase in distance is not worth the 0.1 decrease in delay. But even in this sweet spot, there is a vehicle extra used and the traveled distance is quite high. This is due to the number of rebalancing actions that were performed. These tests were done with a rejection distance of 1.5 kilometers. Maybe a lower threshold at a 30-minute horizon will result in a better outcome. With a lower threshold, more requests are denied because they are too far away. This could potentially save traveled distance and vehicle usage. The results of the test performed in investigating the rejection distance threshold are displayed in Table 6.4.

Table 6.4.: The results of the rebalancing based on forecast tests with a different rejection distance in the ILP.

Rejection dist. [km]	**Reb. performed**	**Avg. pick-up delay [s]**	**Served requests**	**Shared requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base	-	18.2	155	47	8	341	412
0.5	11	21.2	155	47	9	324	410
1	47	19.2	155	45	9	347	409
1.5	79	17.9	155	45	9	389	410
2	105	15.8	155	44	9	414	410
3	108	16.6	155	47	9	426	410

An increasing rejection distance obviously results in increasing rebalancing actions performed. The distances toward rebalancing targets are allowed to be larger so more optional targets are selected. The increase in rebalancing actions relates to the decrease in average pick-up delay and an increase in traveled distance, except for the last case. The last rejection distance results in an increased pick-up delay, but the distance increases as well. The sweet spot is therefore around the 2 kilometer rejection distance.

6.4. Rebalance based on intensity

In this section, a new rebalancing strategy will be discussed. It is constructed with the intensity measure from Section 5.3. This measure has already proven to provide a good valuation of locations with a waiting strategy. In the literature, this measure is however not used for rebalancing. The measure has numerous benefits. Most importantly, it utilizes the historical request data without the requirement of a prediction model. Creating and tuning an accurate prediction model can be a costly exercise. As we saw in the previous section. Moreover, the intensity measure is time-dependent. Because the intensity is calculated for the next hour, the time that the calculation is done plays a role as well, potentially increasing its performance. Additionally, the value of the intensity is not only based on the requests emerging from that location but also on the other locations around it. This is a beneficial characteristic for rebalancing because a rebalancing action where a vehicle moves closer to a future request, results also in a successful rebalancing action. The intensity measure in its original form does however not provide satisfying results so first a modification is made. On top of that, there are numerous parameters that can be tuned to get the best results. These subjects will be elaborated on in this section.

6.4.1. Local intensity

It was imagined that a rebalancing strategy based on intensity would provide a dynamic strategy that explores rebalancing actions that were not seen before. However, after some analysis, the original intensity measure is not as dynamic as expected. Because the intensity measure always

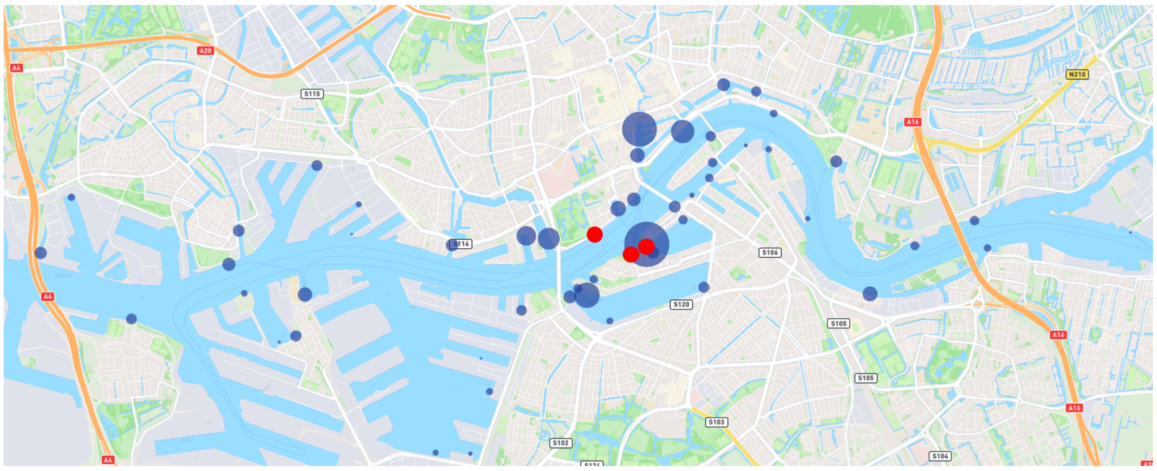


Figure 6.3.: The map of Rotterdam with the relative number of requests per location given by the size of the blue circle. The dominant places where the intensity was the highest throughout the entire day are displayed as red dots.

takes into account all of the historical requests in the next hour, the highest intensity locations are a bit nuanced. The highest intensity locations that were seen throughout the day were all located in the middle of the map, and mostly, the terminal. This can be seen in Figure 6.3.

This is because the sum of transition times to all historical requests will be the lowest in the middle. It makes sense that the overall best rebalancing target is somewhere in the middle. But it would also be interesting to see rebalancing targets caused by high intensity due to local requests only. This would give a more localized aspect to the measure. A location could become a target because the transition times for local requests are very low. This could give more variated results and a way to move away from only rebalancing to the center of the map. A visualization of local intensity is shown in Figure 6.4. A circle is the best representation to show that only local requests are included, but in reality, this is based on travel time. If only requests within the circles are included in the intensity calculation, the results for the highest intensity would be different and give a more local representation of its value as a rebalancing target. The size of this circle, or maximum travel time, needs to be determined. This parameter, among others, will be analyzed in the next section.

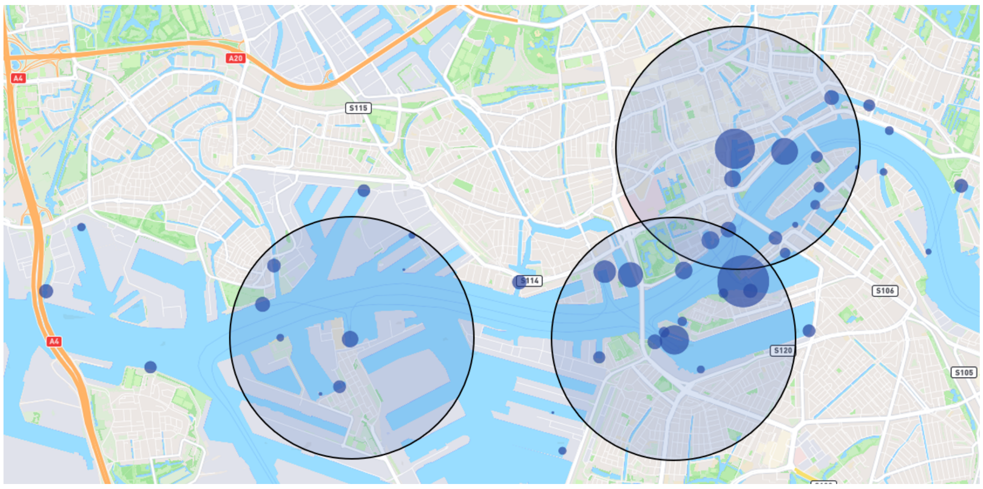


Figure 6.4.: An example of calculating the intensity by only accounting for local requests. Here displayed as a circle, in reality, based on travel time.

6.4.2. Tuning

The maximum travel time from a location to be included in the intensity calculation is not the only parameter that needs investigating. When building a strategy around this measure, a choice needs to be made when a location becomes a rebalancing target. Like the most 'popular' location, a strategy could be to rebalance to the highest intensity location. This would make sure that the highest intensity location at that time step is provided with an empty vehicle. The high intensity would mean that the location has a high probability of being close to an upcoming request. Having a vehicle stationed at that location seems like a good strategy. It then has to be determined what number of locations works best. Rebalancing to the top 2 or top 3 could also have good results. In this section, this parameter is investigated. Another way to implement the strategy is by setting rebalancing targets according to the value of the intensity. If the intensity value is higher than a certain threshold, a rebalancing target is set to that location. With this method, there is a large influence on how many rebalancing actions will be performed. And the targets can be limited to the highest intensities only. Both methods have their advantages, so a combination of both methods might also be interesting.

The maximum travel time for a request to be considered in the intensity calculation is tested first. With the travel times ranging from 1 minute to 4 minutes. With a 4-minute radius, more than half of the requests are included so this radius comes close to having no limit at all. It is expected that with a smaller radius, more variated rebalancing targets will be assigned. If this has a positive influence on the results, has to be tested. The results of the tests are shown in Table 6.5. Additionally, charts have been made to show the difference in variance with the different maximum travel times. This is displayed in Figure 6.5.

Table 6.5.: The results of the rebalancing based on intensity tests with a different search radius for intensity calculations.

Radius [min]	**Reb. performed**	**Avg. pick-up delay [s]**	**Served requests**	**Shared requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base	-	18.2	155	47	8	341	412
1	73	18.5	155	45	8	466	412
2	85	12.3	155	44	9	435	413
3	52	18.9	155	46	8	382	410
4	57	19.2	155	48	8	427	410

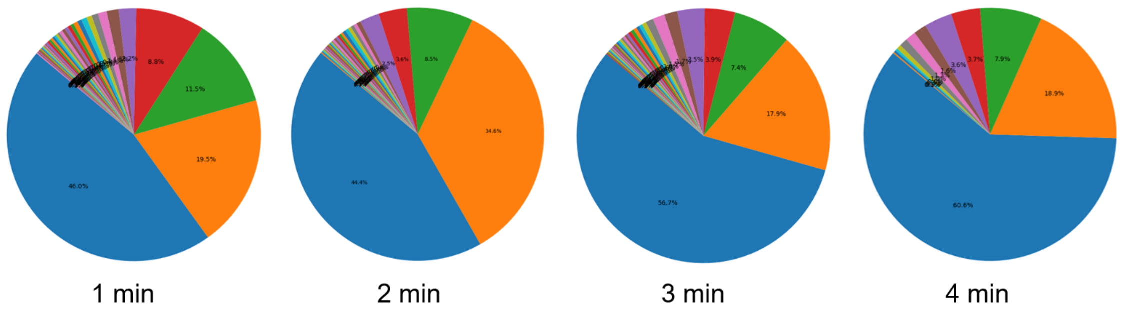


Figure 6.5.: The distribution of highest intensity locations throughout the day. A 1-minute radius shows a higher variance in locations than a 4-minute radius

The tests show that a radius of 2 minutes has the best average pick-up delay. Significantly better than a radius of one minute but this is also caused by the use of an extra vehicle. This is caused by the number of rebalances that is largest for a 2-minute radius. The scenarios with a radius of 3 or 4 minutes use a vehicle less than the 2-minute radius. This causes the pick-up delay to increase. These 2 scenarios resulted in fewer rebalancing actions because the rebalancing target was not switched as often. As a result, fewer vehicles were used and less distance was traveled.

The effect on the results caused by the number of target locations is investigated as well, with the number of targets ranging from 1 to 4. It is expected that the number of rebalances performed will increase proportionally with the added targets. The results are shown in Table 6.6.

As expected, the number of requests increases greatly when extra targets are added. It increases harder than expected because the second and third highest intensity locations tend to vary more, so more rebalancing actions are performed. With the increase in requests, also the number of vehicles increases. As a result of the extra vehicles and the rebalancing actions, the average

Table 6.6.: The results of the intensity rebalancing strategy tests with increasing rebalancing targets.

Targets	**Reb. performed**	**Avg. pick-up delay [s]**	**Served requests**	**Shared requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base	-	18.2	155	47	8	341	412
1	52	19.0	155	46	8	383	410
2	177	10.1	155	40	10	469	413
3	297	8.3	155	35	10	614	415
4	481	8.6	155	34	11	718	415

pickup delay can be decreased significantly. The distance, on the other hand, increases as well so it is again a trade-off between delay, and vehicles and distance. The average pick-up delay increases slightly with four rebalancing targets. It indicates that more unsuccessful than successful rebalancing actions are added when adding a fourth rebalancing target.

The last parameter that will be tested, is the intensity threshold. A low threshold will generate a large amount of targets and a high threshold only a select number. The intensity over the entire day is plotted to know what values to choose for the threshold. In Figure 6.6, not only the highest intensity but also the second, third, and fourth highest intensities are shown. The chosen thresholds to test are displayed as the red lines. The results of these tests are shown in Table 6.7.

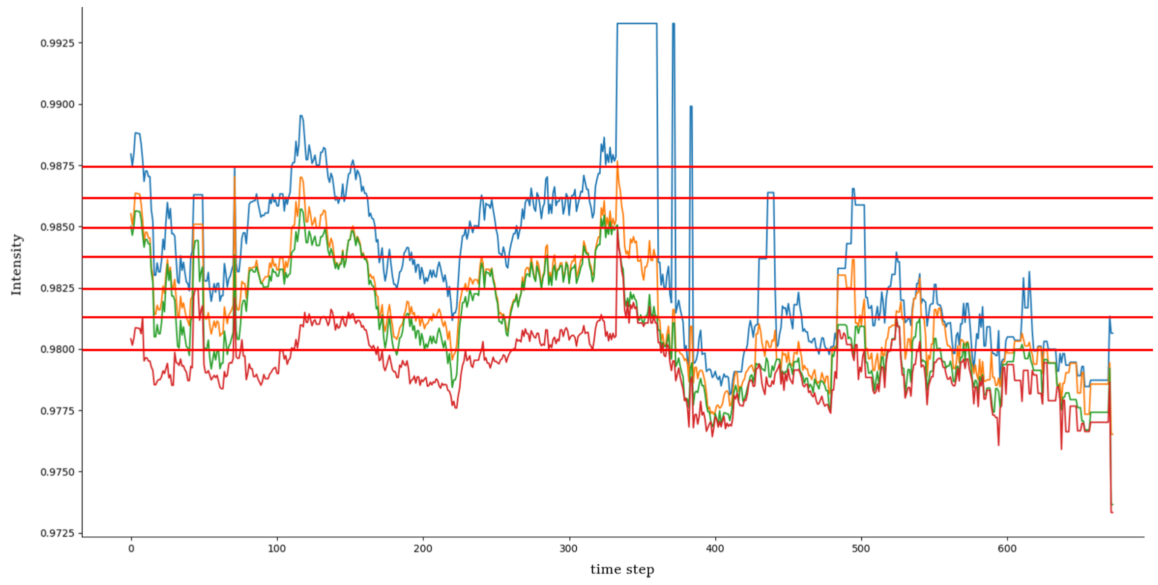


Figure 6.6.: The top 4 highest intensity values throughout the entire day. The intensity is calculated with a radius of 3 minutes. The thresholds that will be tested are displayed as the red lines.

With a decreasing threshold, rebalancing actions increase, as expected. It is again clear that when more rebalances are performed, the use of vehicles and distance traveled increases. The average pick-up delay decreases. This is due to successful rebalancing actions but also because of the fact that there are more vehicles to work with. The edge cases of the thresholds have less difference

Table 6.7.: The results of the tests of the intensity rebalancing strategy where the threshold is variated.

Targets	Reb. performed	Avg. pick-up delay [s]	Served requests	Shared requests	Vessels used	Distance empty [km]	Distance loaded [km]
Base	-	18.2	155	47	8	341	412
0.9875	6	19.3	155	47	8	347	412
0.98625	21	19.3	155	46	8	372	412
0.9850	50	16.2	155	42	11	386	411
0.98375	75	13.5	155	41	10	429	410
0.9825	128	9.5	155	36	10	482	415
0.98125	188	9.5	155	37	11	533	412
0.9800	322	9.4	155	37	12	607	411

than the middle cases. For the 2 highest thresholds, this is caused by the fact that the added rebalances are not that many, also they are not all successful rebalances since the delay remains the same. The last three thresholds also do not differ that much from each other in terms of pick-up delay. It illustrates the fact that adding more rebalancing actions does not necessarily mean more successful rebalances.

Every parameter has been tested separately and the relationships between the parameters and the results have been investigated. A smaller location search radius means more variety in rebalancing targets, and more target locations and a lower threshold mean more rebalancing actions that eventually lead to more traveled distance and vehicles, but less delay. What could be more interesting, is how the combination of all the different parameters can work together. To search for the best parameter combination, the previously tested parameters will now be tested for all combinations. 4 target parameters, 4 radius parameters, and 7 threshold parameters result in 112 combinations. The most interesting results of the combinations are shown in Table 6.8

Table 6.8.: The most interesting results from the parameter combinations for the intensity rebalancing strategy.

Parameters (radius, targets, threshold)	Reb. performed	Avg. pick-up delay [s]	Served requests	Vessels used	Distance empty [km]	Distance loaded [km]
Base	-	18.2	155	8	341	412
4 min, 1, 0.98625	6	17.8	155	8	359	412
1 min, 2, 0.985	165	11.6	155	8	509	412
2 min, 4, 0.9875	250	7.2	155	10	506	415

From all the combinations, there are 3 results extracted. Each of these results performs best in their own category. The first one is the most conservative, trying to keep as much of the results the same while decreasing the delay. The second one is minimizing the delay while keeping the number of used vehicles the same. And the last one is minimizing the delay without limitations.

When looking at the parameters, the first result is tuned conservatively. It has the highest intensity radius, meaning that the highest intensity location will vary the least. It rebalances to maximally one target at a time and it almost has the highest threshold. A threshold of one step higher could be possible but that did not lead to better results in this case. The delay is slightly decreased and the distance traveled is slightly increased. Results that can be expected when only 6 rebalancing actions have been performed. The parameters of the second result are a bit more bold. It has a small radius of 1 minute, which creates a lot of varying rebalancing targets. It also has the lowest threshold of the 3 results shown. To make sure the strategy does not create too many rebalancing actions, the maximum number of rebalancing targets is set to 2. This results in 165 rebalancing actions. This increases the distance traveled quite a bit, but also strongly decreases the pick-up delay. The number of vehicles used remains the same. The most extreme result is the last one. The combination of 4 rebalancing targets at a high threshold, means that lots of rebalancing actions will be performed, but only if the high threshold is reached. It results in the most rebalancing actions of all, which translates to the lowest pick-up delay as well. Contrary to the other results, it uses 10 vehicles. The use of these extra vehicles causes the traveled distance to be slightly less than the second result.

Depending on a company's preference, each of these parameter settings can be implemented. If a company wants to be more on the safe side and not invest too much extra distance traveled in return for a decreased pick-up delay, the first parameter settings can be chosen. If a company would not like to use an extra vehicle for rebalancing but is fine with extra distance, the second parameters can be used. Finally, if a company wants to minimize pick-up delay and is willing to use extra vehicles for that, the last parameter settings can be used.

6.5. Results

Table 6.9.: The most relevant results of the different rebalancing strategies summarized together.

Strategy	**Reb. performed**	**Avg. pick-up delay [s]**	**Served requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base	-	18.2	155	8	341	412
Location top 1	33	17.9	155	8	348	409
Location top 2	73	19.1	155	7	422	410
Location top 3	106	16.2	155	8	531	410
Delay	8	19.0	155	8	361	409
Forecast	105	15.8	155	9	414	410
Intensity 1	6	17.8	155	8	359	412
Intensity 2	165	11.6	155	8	509	412
Intensity 3	250	7.2	155	10	506	415

Rebalancing strategies overall, will increase the total traveled distance of the system. This makes sense because adding extra trips that do not directly serve customers adds extra empty distance by definition. The only way it can not add distance is if the vehicle would have driven that rebalancing distance anyway. In rare cases, the modified distribution of vehicles could result in less distance driven in total. The trend of adding distance is clearly visible in the results of the rebalancing strategies. The goal of the strategy is to trade the cost of added distance with an improvement in another performance factor. In most cases, this is done by reducing the average pick-up delay. In theory, this is also the main reason why a vehicle should rebalance: it can get closer to new potential requests, thereby reducing delays. Every strategy shows a decrease in pick-up delay, except for 2. The rebalancing strategy based on delay does not perform well in this environment. It is the only strategy that is not able to decrease the average pick-up delay and not improve other performance factors of the result. The other strategy that does not improve the delay is the rebalancing strategy based on the top 2 locations. This strategy, however, does improve an important performance factor: the number of vehicles used. By rebalancing to the right locations, this is the only rebalancing strategy that requires 7 vehicles to serve all requests. All rebalancing strategies show the same correlation between the number of rebalancing actions and the distance traveled. The intensity rebalancing strategy can perform the best in terms of average pick-up delay. When extra vehicles are allowed to be used, but also when 8 vehicles are used. The location top 3 strategy is the best contender after that but is still outperformed in terms of delay and distance. The forecast rebalancing strategy reduces the pick-up delay the most apart from the intensity strategies, but in order to do so it does need an extra vehicle. The strategy that was able to decrease the delay while adding the smallest amount of distance is the location top 1 rebalancing strategy.

7. Discussion

7.1. Reflection on all strategies

The results of waiting and rebalancing strategies can be compared with each other to show the characteristics of each strategy. The results are presented together in Table 7.1 and a visual representation in Figure 7.1. Waiting strategies, for instance, can result in better matches by slightly altering the vehicle distribution. This has advantages and disadvantages. An advantage is that no distance is added by waiting. This is also visible in the results, where the distance traveled is only decreased. A disadvantage of a waiting strategy is that results cannot be altered as aggressively as a rebalancing strategy can. By waiting at one spot or another spot in the route, there is only so much of a difference that it can make. Although, when correctly tuned, saving one vehicle usage is still a great result. Rebalancing strategies also prove to be far superior when they involve traveled distance minimization. This makes sense because adding extra trips with rebalancing will normally only increase the traveled distance. Consequently, this is also one of the main disadvantages of a rebalancing strategy. There is no way of implementing an effective rebalancing strategy without adding traveled distance, as is proven in the results. Rebalancing strategies can greatly influence pick-up delays, but pay their price for it. When implementing rebalancing strategies, tuning is important because rebalancing strategies usually have a sweet spot. With too few rebalancing actions the results are hardly influenced, but with too many not only the traveled distance shoots up, but also the number of vehicles required increases rapidly.

Table 7.1.: The results of waiting strategies and rebalancing strategies in one table.

Strategy	**Reb. performed**	**Avg. pick-up delay [s]**	**Served requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base		18.2	155	8	341	412
Waiting strategies						
Wait first		32.5	155	9	215	408
Intensity		19.2	155	7	340	410
Rebalancing strategies						
Location top 1	33	17.9	155	8	348	409
Location top 2	73	19.1	155	7	422	410
Location top 3	106	16.2	155	8	531	410
Delay	8	19.0	155	8	361	409
Forecast	105	15.8	155	9	414	410
Intensity 1	6	17.8	155	8	359	412
Intensity 2	165	11.6	155	8	509	412
Intensity 3	250	7.2	155	10	506	415

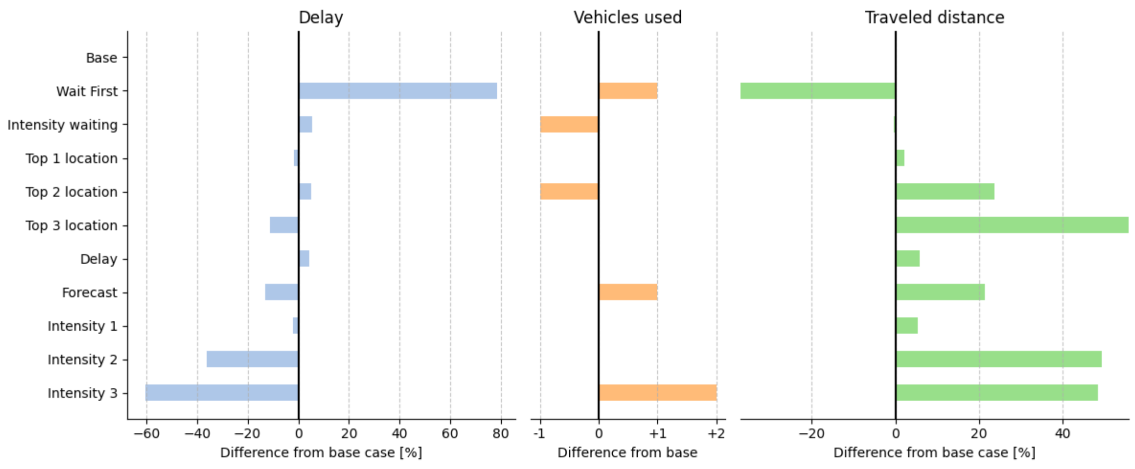


Figure 7.1.: The results of all strategies visualized in three graphs, displaying the difference in terms of delay, vehicles used, and traveled distance.

The different results from rebalancing and waiting strategies give a good insight into the characteristics of the strategies, but what could also be interesting is to look at how combinations of these strategies can influence the results. Strategies with different characteristics could balance each other out or could improve their performance on specific factors even more. The first combination is done with the Wait First strategy and all of the rebalancing strategies. The results are displayed in Table 7.2.

Table 7.2.: The most relevant results of the different rebalancing strategies in combination with the Wait First strategy.

Strategy	**Reb. performed**	**Avg. pick-up delay [s]**	**Served requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base		18.2	155	8	341	412
Rebalancing strategies & Wait First						
Location top 1	38	25.1	155	8	258	408
Location top 2	58	25.5	155	8	277	406
Location top 3	101	22.1	155	9	324	406
Delay	12	31.1	155	9	257	407
Forecast	78	23.5	155	10	223	403
Intensity 1	8	32.4	155	8	230	407
Intensity 2	156	16.6	155	9	308	412
Intensity 3	249	13.1	155	10	302	412

It is clearly visible that the Wait First strategy has its impact on the results compared to Drive First as a base. The characteristic of Wait First was that the pick-up delay was increased, but the traveled distance was decreased. This characteristic is combined with the results of the rebalancing strategies. For every rebalancing strategy, the delay has increased and the distance traveled is decreased. However, since the rebalancing strategies did the exact opposite, this results in some interesting output. Every rebalancing strategy now results in a lower traveled distance than the base case. So even though the number of rebalancing actions may be high, the traveled distance only decreases. On the other side, a lot of the pick-up delays are higher than the base case. Only the last two rebalancing strategies have a lower delay, but these strategies use more vehicles. A combination of Wait First and a rebalancing strategy could be useful when the distance traveled needs to be decreased, but a high increase in pick-up delay, like Wait First only, is not desired. The results for the intensity wait strategy and the rebalancing strategies are displayed in Table 7.3.

Table 7.3.: The most relevant results of the different rebalancing strategies in combination with the intensity waiting strategy.

Strategy	**Reb. performed**	**Avg. pick-up delay [s]**	**Served requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base		18.2	155	8	341	412
Rebalancing strategies & Intensity						
Location top 1	27	19.1	155	7	354	408
Location top 2	78	15.2	155	8	402	406
Location top 3	99	16.3	155	8	454	406
Delay	9	21.1	155	8	339	407
Forecast	135	10.0	155	11	373	403
Intensity 1	6	19.2	155	7	346	407
Intensity 2	163	13.3	155	8	506	412
Intensity 3	256	7.6	155	10	493	412

The combination of strategies is less one-sided than Wait First combined with the rebalancing strategies. In pick-up delay, there is not a general decrease or increase observed for all strategies. The combination sometimes results in more delay and sometimes also in less. In 2 combinations, the number of vehicles could be reduced to 7, but it also needs 8 vehicles for the top 2 locations rebalancing strategy, where previously 7 were needed. Overall, with the combinations that increase the delay, the distance is reduced and vice versa. The combination of rebalancing strategies with the intensity strategy could be beneficial when a vehicle less is needed and the minimization of pick-up delay is preferred above the distance minimization.

7.2. Sensitivity analysis

Sensitivity analysis can be an important tool in evaluating the stability, credibility, and comprehensiveness of an analytical study. A sensitivity analysis examines how variations or uncertainties in different parameters, assumptions, or methodologies affect the outcomes of the model or study. This section delves into a systematic exploration of the sensitivity of key variables or assumptions within this study, shedding light on their impact and the resulting implications on the results and conclusions.

7.2.1. Base case sensitivity

The first sensitivity tests will be done on the objective parameters of the base case. The base case, with its Drive First strategy and no rebalancing, is a good place to start to check if the objective function behaves as expected. When a certain parameter is increased, this means that part of the function has a higher priority while making decisions. The objective function, as constructed in Section 4.4.2 is displayed again in Equation 7.1.

$$\text{maximize: } \sum_{i \in \mathcal{T}} r_i x_{ij} - \sum_{i \in \mathcal{T}, j \in \mathcal{V}} c^d \cdot d_{ij} x_{ij} - \sum_{i \in \mathcal{T}, j \in \mathcal{V}} c^s \cdot s_{ij} x_{ij} - \sum_{j \in \mathcal{V}} c^v \cdot v_j x_{ij} \tag{7.1}$$

In this objective function c^d, c^s, and c^v represent the the objective parameters of the delay, the distance, and the vehicles, respectively. Each of them allows giving more or less priority to its corresponding factor. Increasing the c^d for instance would mean that a higher cost is associated with the delay that occurs during the simulation. A higher cost means that when maximizing the objective function, there is a higher priority on minimizing the delay because this will increase the overall solution value. Decreasing the parameter would result in the opposite, where the priority on minimizing the delay is decreased because it has less effect on the overall solution value. Using this mechanic, the emphasis of the objective function can be shifted towards every factor of the objective function.

To validate the model, it can be tested if the results of the simulation match the expected results when a shift in emphasis is applied to the objective function. When the emphasis is shifted towards more priority for delay, it is expected that a decrease in delay is observed in the results of the simulation. More choices will probably be made in favor of delay reduction so this should be visible in the results. It is also expected that a slight increase in the other factors can be observed because a trade-off will have to be made somewhere. Contrary, when the emphasis is moved away from a certain factor it is expected that this factor will decrease in the simulation results and that the other factors will slightly increase. To check these expectations, tests are performed where the emphasis is moved towards and away from the delay, the distance, and the used vehicles. The emphasis is created by increasing or decreasing the specific parameter value by 10%. The results are shown in Table 7.4.

Table 7.4.: The different results for scenarios where the objective function is modified to have more or less emphasis on delay, distance, and vehicles used.

Scenario	**Avg. pick-up delay [s]**	**Served requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base	18.2	155	8	341	412
Delay +	18.1	155	7	366	410
Delay -	17.5	155	9	334	411
Distance +	17.6	155	9	326	411
Distance -	18.1	155	7	366	410
Vehicles +	24.2	155	7	357	409
Vehicles -	18.2	155	8	341	412

Looking at the results, one column that stands out is the vessels used column. The vessels that are used in the scenarios vary a lot, also for the scenarios where the emphasis on delay and distance is varied. It means that the number of vessels used is sensitive to small changes in the objective function. For instance, when the delay parameter is increased, the average delay decreases as expected. However, the number of vehicles decreases as well. This is opposite to what was expected as well because more emphasis on delay would usually mean that other factors like vessels used or distance traveled would increase, not decrease. When the delay parameter is decreased, an extra vehicle is used. This is against expectations as well. As a result of the extra vehicle needed, the average delay decreases because the extra vehicle at hand can help solve the problem with less delay and traveled distance.

The same phenomenon can be observed with the shifts in emphasis on distance as well. Interestingly, the scenarios where the distance parameter is varied produce almost the exact opposite result as the results when the delay parameter was varied. This can be explained by the fact that these two parts of the result always have the most obvious trade-off. When observed delay decreases, this is almost always paired with an increase in traveled distance, and vice versa. This is because a decrease in delay can be directly linked to an assignment that has traded some of its distance toward delay. The use of an extra vehicle would not be traded that easily for a decrease in delay or distance, only if that value builds up to a large amount, which typically happens when the current number of vehicles cannot serve the demand anymore. The direct trade-off between distance and delay induces the fact that an increase in the delay parameter causes the same results as a decrease in the distance parameter. The same holds for the opposite. Here, the results are not identical but they are similar.

The variation of the vehicle parameter has some interesting results as well. When more priority is given to vehicles in the objective function, a vehicle less is used just like in the 'Delay +' and 'Distance -' scenarios. However, significantly more average delay is observed. More distance is traveled than in the base case but less than in the other two scenarios where the number of vehicles used is reduced. When the vehicle parameter is reduced, the results are identical to the base case. Probably no decisions during the simulation were altered because of this parameter reduction.

The sensitivity observed in vehicle usage can be explained by looking at how the objective function is used. It must be noted that the assignment is not done by solving one big ILP. Instead, every time step a small ILP is solved that determines the assignment of 2 or 3 requests on average. This means, that the ILP only maximizes the solution value of those currently available requests. What total results are observed at the end of the simulation is definitely influenced by the objective function, but it does not directly control it. The total simulation output at the end is the result of 43200 ILPs that have been solved in sequence. Changing the objective function of all those ILPs can have unanticipated results. The input of the ILPs are always the requests, but also the state and distribution of the fleet. This distribution of the fleet is completely dependent on all previous assignments. If one assignment changes at the beginning of the simulation, this single change in distribution can cascade into an entirely different distribution of vehicles 2 hours later for example. Moreover, by changing one objective function, all of the assignment ILPs now have a different preference in making decisions. The result is a completely different scenario. Changing a parameter of the objective function can lead to a new sequence of assignments that form the solution. With that in mind, it makes sense that there is more variance in the results than first was expected. The number of vehicles used is especially susceptible to the vehicle distribution because if at a certain busy moment, a vehicle is in the right spot, it could serve a request that otherwise would need an extra vehicle. Increasing the delay parameter may trigger a long sequence of different assignments that ultimately lead to a result where not only the delay is decreased, but also a vehicle less can be used as a result of the changed distribution.

7.2.2. Fixed vehicle usage

So there is an explanation for the variance in results, especially in terms of vehicle usage. But still, the differences in the vessels used make it hard to compare the rest of the results with each other. The number of vessels used has a large influence on the delay and distance as well. This is visible from the results in Table 7.4. The places where 9 vehicles were used have the lowest delay and the lowest distance traveled compared to the other results. The highest delay was observed when 7 vehicles were used and this also resulted in relatively higher distances traveled. In summary, the results were more dependent on the number of vehicles that were used, than on the actual change in parameter. To counter this, the number of vehicles can be fixed. With the same number of vehicles available every time, the results can be better compared with each other because there is no disruption caused by the number of vehicles used. The results of tests where the parameters of delay and distance were variated and the number of vehicles used is fixed to 8, can be observed in Table 7.5.

Table 7.5.: The different results for scenarios where the objective function is modified to have more or less emphasis on delay and distance. Now the number of vehicles used is fixed to 8.

Scenario	**Avg. pick-up delay [s]**	**Served requests**	**Vessels used**	**Distance empty [km]**	**Distance loaded [km]**
Base	14.4	155	8	338	412
Delay +	13.2	155	8	328	413
Delay -	18.5	155	8	356	412
Distance +	15.0	155	8	343	412
Distance -	13.2	155	8	328	413

It can be seen from the results that the base scenario now has a lower average pick-up delay and a lower traveled distance. This is because the ILP has one less factor to optimize. Because the number of vehicles is fixed, there is always the same number of vehicles available. The system will not try to avoid using extra vehicles by trading delay or distance. Instead, it is purely focused on minimizing the delay and traveled distance. This results in a better performance. The results are now more in line with the expectations. When there is an emphasis on delay, the average delay decreases. When the delay is not a priority anymore, the average delay increases. The base case provides a clear intermediate result compared to these variations. What is interesting is that the distance traveled decreases with the delay as well. So when the delay parameter is decreased, not only does the average pick-up delay increase compared to the base case but also the traveled distance. An increase in the delay parameter results in a better overall performance and vice versa. Just like before the number of vehicles was fixed, the 'Delay +' case is identical to the 'Distance -' case. Also, the 'Delay -' and 'Distance +' cases are similar. The direct trade-off between delay and distance is still in place, so increasing one parameter yields the same results as decreasing the other. This means that when the distance parameter is increased this results in an increase in traveled distance, opposite to the expectation. This result can again be explained by the fact that decisions may decrease distance in the short term, but change the vehicle distribution in a way that in the long term, more distance is traveled in total.

7.2.3. Strategy sensitivity

With more reliable results when the number of vessels used is fixed, the sensitivity of strategies can be tested with varying objective parameters. Before, we have only tested the base case, which applies Drive First without rebalancing. It would be good to know how other strategies influence the outcome when the objective function is altered. Normally, it would be expected that with slightly altered objective parameters, the effect of the strategies will roughly remain the same. But the tests on the base case have already provided unexpected results so testing the strategies as well is very relevant.

The tests will be performed with the same fixed number of vehicles as before. This will make sure that results are comparable with each other and that the results are not disturbed by the differences in vehicle usage. The strategies that will be tested are: Wait First, 3 variants of rebalancing to a popular location, and rebalancing based on delay. These strategies are chosen because they do not require tuning and can be directly implemented. Strategies that include intensity, for instance, have to be tuned for every variation of the objective function. This would take too much time to do for every variation so these strategies are skipped during the sensitivity analysis. The strategies are tested on the base scenario where the number of vehicles is fixed to 8. The results are displayed in Table 7.6 In addition to the values, the change in percentage compared to the base case is displayed as well. This will help to determine if the results of the strategies are comparable when switching to different objective parameter scenarios.

Table 7.6.: The results of different strategies when no parameter is increased but the number of vehicles is fixed to 8.

Strategy	Avg. pick-up delay [s]	Difference [%]	Total distance	Difference [%]
Base	14.4	0	750	0
Wait First	26.8	+86	604	-19
Top 1 location	10.5	-27	794	+6
Top 2 location	12.9	-10	797	+6
Top 3 location	14.8	+3	894	+19
Delay	14.6	+2	752	0

It is clear that the waiting strategy still behaves the same. The average delay increases and the traveled distance decreases, the same effect as observed earlier. The rebalancing strategies behave similarly, but there are some differences. There is still the same trade-off between delay decrease and distance increase. The delay rebalancing strategy is still not able to produce a lower average pick-up delay but it does increase the distance traveled. The difference is in the rebalancing based on location strategies. In the original tests, the more rebalancing targets were added, the more the delay was reduced. This also resulted in the distance traveled increasing more with every target added. When the number of targets was increased too much, the delay could not decrease anymore, but instead, increases again while still adding extra distance traveled. This effect occured because vehicles would have their hands full with constant rebalancing, so on average, vehicles arrive later at the pick-up point. The same effect is seen here with 8 fixed vehicles, but the threshold has shifted up. Now, with only two rebalancing targets, the delay decrease is less than with one target, and the distance is still increased. With three targets, the delay is even higher than the base case. As a result of this test, the 'Top 1 location' strategy would be the best choice, which is different from the original rebalancing results.

With the results of the default scenario in mind, tests can be done on these strategies while variating the objective function parameters. The same scenarios will be tested where the parameter for the delay will be slightly increased and decreased, also the parameter for distance will be slightly increased and decreased. These tests are again performed while fixing the number of used vehicles to 8. The results are shown in Table 7.7. The percentual difference of the strategies compared to the base case is also visualized in Figure 7.2.

The graph provides a clear overview of how the strategies are affected by changes in the objective function. Looking at the difference in distance compared to the base case, the different scenarios provide very similar results. The results also align with the original waiting and rebalancing results. Wait first has a large decrease in traveled distance. With increasing numbers of rebalancing

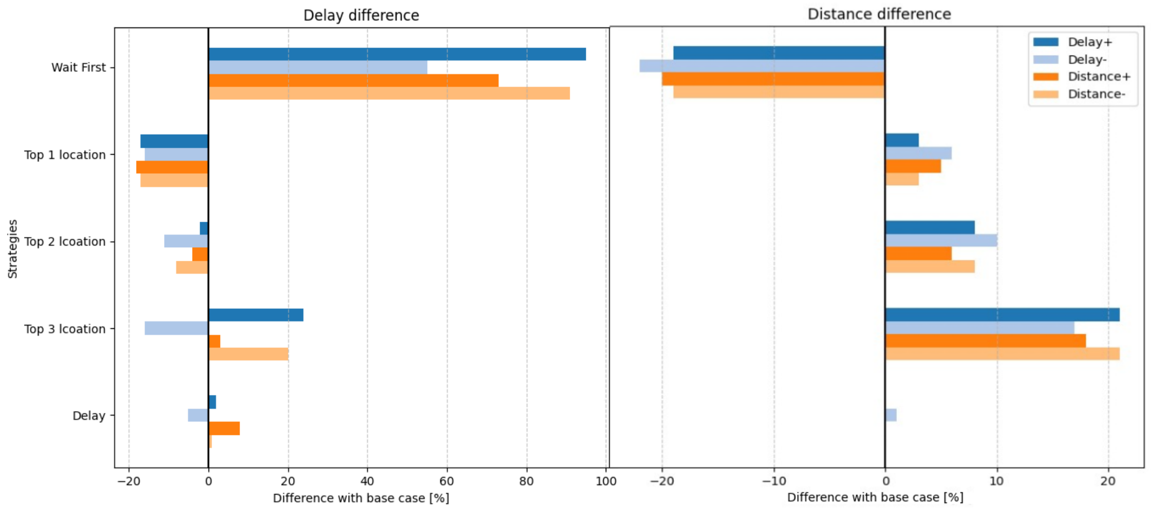


Figure 7.2.: The results of the sensitivity analysis performed with different strategies. The results are grouped per strategy, the colors stand for the different strategies.

targets, the distance traveled also increases. The delay strategy rebalances relatively few vehicles so the added distance is also minimal. For most scenarios, the delay difference observed is similar. With a large increase in delay for the Wait First strategy for instance. The top 1 location strategy has the most decrease in delay, and this reduction becomes smaller with the top 2 location strategy. For the top 3 location strategy, the 'Delay-' scenario behaves differently compared to the rest of the scenarios. Instead of increased delay, the scenario experiences a reduction in delay that is greater than the one before. Also, with the delay rebalancing strategy, the 'Delay-' scenario shows a reduction in average pick-up delay. This has also not been observed in the original rebalancing results.

The reason why this 'Delay-' scenario behaves differently compared to the rest can again be explained by the change in vehicle distribution that is caused by the different objective functions. Changes in vehicle distribution that occur when the delay parameter is reduced, resulting in a scenario where the rebalancing strategy to 3 locations works really well compared to other scenarios. This effect is not in line with the reduction of the delay parameter, because it should lead to less attention to delay, which should increase it. Therefore, this behavior is unfortunately not predictable so there is no way to potentially compensate for it.

The behavior is not predictable, but it does not mean it can avoided. As explained before, the 'random' change that the results can have when different objective scenarios are applied, can be explained by the fact that a large number of ILPs are solved in sequence during the simulation. For every time step, an assignment is made that is dependent on an ILP to make the best decision with the current information. The problem is that the information is not sufficient. A decision that is going to be made, can only base its logic on effects 10 minutes into the future. This is a problem because the effect of one decision can have negative implications multiple hours later. This is why changing an objective parameter can have these counteractive effects, all due to the lack of information when creating the assignment. This is combined with the fact that one different decision can cascade into an entirely different scenario in terms of vehicle distribution. It can be partially solved, however, when more information about future implications is known. This can be achieved by implementing a planning-based matching method that steps away from the MoD assignment structure. With a planning-based matching method, the effect that one

Table 7.7.: The results of different strategies when no parameter is increased but the number of vehicles is fixed to 8.

Scenario	**Strategy**	**Avg. pick-up delay [s]**	**Difference [%]**	**Total distance**	**Difference [%]**
Delay+	Base	13.2	0	742	0
	Wait First	25.7	+95	601	-19
	Top 1 location	10.9	-17	765	+3
	Top 2 location	12.9	-2	799	+8
	Top 3 location	16.4	+24	896	+21
	Delay	13.5	+2	744	0
Delay -	Base	18.5	0	769	0
	Wait First	28.7	+55	599	-22
	Top 1 location	15.5	-16	818	+6
	Top 2 location	16.4	-11	844	+10
	Top 3 location	15.6	-16	896	+17
	Delay	17.6	-5	774	+1
Distance +	Base	15.0	0	755	0
	Wait First	25.9	+73	606	-20
	Top 1 location	12.3	-18	783	+5
	Top 2 location	14.4	-4	815	+6
	Top 3 location	15.5	+3	909	+18
	Delay	16.2	+8	736	0
Distance -	Base	13.2	0	741	0
	Wait First	25.2	+91	598	-19
	Top 1 location	10.9	-17	765	+3
	Top 2 location	12.2	-8	801	+8
	Top 3 location	15.8	+20	905	+22
	Delay	13.3	+1	742	0

assignment has on future planning is directly calculated. Different assignments made by a change in objective function now have knowledge about its implications in the future. As a result, changes in decisions will only be made when this benefits the overall output according to the planning. More on the effects of an increased planning horizon will be explained in the next section.

7.3. Limitations of a short planning horizon

An important factor in the simulation model in terms of computational expense, is the planning horizon. Unfortunately, the short horizon of 15 minutes does not provide a look at the long-term future. The fact that the simulation model was based on MoD made it hard to come up with another solution than a short time planning. The transformation of the model from MoD to a mixed request system was a harder task than anticipated. It took more effort than implementing the strategies and the data analysis combined. First of all, because it was a large model and a full understanding was needed to change such an intrinsic aspect of it. And second, even with keeping a lot of the MoD matching intact, the number of changes needed was already very significant.

After some consultation with Flying Fish, it became clear that they use a suggestion-based algorithm where the complete future planning is known. When an inquiry is made, the system will check where the request can be inserted without it clashing with other reservations. The exact algorithm could also not be provided because this is the intellectual property of Flying Fish and is not to be shared with third parties, understandably. From the discussion about it, however, it was clear that this was an entirely different approach than the existing MoD model. Where the Flying Fish method was based on a secure definitive planning, the MoD method was based on current-time optimality without looking at a single step in the future. Completely transforming the entire request handling and matching phase would almost mean building a new simulator. Since the goal of this research was to investigate waiting and rebalancing strategies using the simulator, the decision was made to keep the MoD matching structure and try to modify it so that it can handle reservations.

It was explained in Section 4.4 how a short-term future planning can be created while keeping the same matching structure. This was done by creating RTV graphs for future timesteps and merging the results into one multi-timestep RTV graph. This works great because there is complete freedom in which visit to choose, and the best combination of visits is determined by the ILP. The downside is that creating RTV graphs is computationally expensive, particularly when there are lots of requests to assign. The planning needs to be modifiable, otherwise, the benefit of a waiting strategy is lost. This means that requests that are already assigned, keep being added to the request pool to check if a better option is available. Setting a longer planning horizon does not only increase the number of RTV graphs that need to be generated, it also makes the request pool bigger because requests appear there for a longer duration. Consequently, increasing the planning horizon increases the computational time exponentially. Hence, a short horizon of 10 minutes was chosen. It is far enough into the future to receive some of the benefits, but short enough for acceptable run times.

The short planning horizon has implications for the results of the simulation. Naturally, there are fewer options to choose from in the matching phase. But most importantly, for example, in the scenario in the previous section, there is no way of knowing what implications an assignment has in the long term. Some decisions may look like good choices at the moment, but looking ahead an hour, for instance, other choices may be more beneficial overall. If the matching phase was planning-based, or a longer planning was implemented, future implications could be taken into account. This would significantly improve the overall solution. Another improvement would come from waiting strategies. Because a request is assigned at a maximum of 10 minutes beforehand, the vehicle can only wait consciously for 10 minutes maximum. In this situation, a vehicle could be idle for 30 minutes, and when it is assigned a request it can then make the choice where to wait. With a longer planning horizon, The vehicle could make the decision at the beginning, deciding where to wait for the full 40 minutes. By not having this future information, the effect of waiting strategies is reduced. Unfortunately, the simulation faces these limitations but with the available equipment and time in this research, a compromise had to be made.

7.4. Test data selection

All of the tests in this research have been performed on the same data set. This data set resembles one day of the week of data that was received. The used day for tests was a Friday, a day that behaves average compared to the rest of the week. One could say that only testing strategies on a single day results in inaccurate results. While that is true, there are clear reasons why this decision was made. First of all, there was only one week of data provided, from Monday to Sunday. Clearly visible in Figure 3.4 is that weekdays are composed quite differently than weekend days. The largest difference is the total number of requests per day. On a Tuesday, about half of the request requests are received compared to Saturday. Another important factor is that during weekdays, the majority of requests consist of passengers that use the water taxi for their commute. This results in a peak of requests in the late afternoon and relatively high activity in the early morning. During the weekend, on the other hand, the peak of requests occurs in the middle of the day. Friday behaves a bit like a mix of the two. It shares features of both scenarios. Therefore, if an average day had to be picked, it would be Friday.

It does, however, still not explain why the strategies were not tested on all days. Apart from the computational time of tests that would be around 7 times higher, there is another important factor why all days should not be tested. For some strategies, there is a requirement for historical requests. The intensity waiting strategy, the intensity rebalancing strategy, and the rebalancing based on forecast strategy all need historical requests to work. It is important that the historical requests give a good representation of the average day, but should not be the same. The intensity measure, for instance, should not contain requests that have yet to show up on that day. That would mean that 14% of requests in the historical request set are requests that will show up that day. This would give an unrealistic predictability to the intensity measure which is not preferred. This explains the importance of an independent 'training' data set.

When strategies that use the intensity measure are implemented for water taxis, it would naturally be better to use a larger historical data set. When enough data is available, a different set can be used for every day of the week. Or even differentiate it more according to the season. From meetings with Flying Fish, it became clear that there exists a large difference in requests for the summer or winter period. Using an accurate data set and tuning the strategy accordingly can increase the performance of the strategies greatly. When keeping in mind that in practice, strategies can be used and tuned for specific days, focusing on one average day in this research is acceptable, given the fact that more data was not available.

7.5. Objective parameters

The ILP objective function plays a large role in the matching phase. The objective function has large control over what specific performance factors should be emphasized. The parameters of the function determine what is prioritized. Assigning numbers to these parameters is hard for multiple reasons. The most important reason is that not everything is quantifiable. A method of assigning the right parameters is by basing it on real-life costs. The distance traveled is the easiest example. The fuel consumption of the water taxis of 1.6 L/km could be used to determine a price per kilometer of around 3 euros. But not everything is as easily calculated. The number of used vehicles, for instance, is already harder. The cost of an extra vehicle does not only contain the driver's wage but also the fact that an extra vehicle is needed in the fleet. This depends on the initial purchase, storage, and maintenance costs. Quantifying delay is even harder. What would be the cost of 20 seconds delay? or 5 minutes? Maybe the company places a lot of value on punctuality, or maybe it doesn't care about delay as long as the number of vehicles is minimized. In other words, basing objective parameters on real-life costs is hard and can be different for every company.

Another way to do this without bothering too much with cost calculations is by tuning the parameters until reasonable results appear. This can be a time-consuming task as well because small changes in parameters can cause large disruptions in the solution. Also, every parameter has an influence on the portion of other parameters so changing one should always be done while taking the other parameters into account. Eventually, a parameter combination results in values that are relatable to the observed values in data analysis. There is not much insight from the data into the delays or distance traveled. But it is very clear how many vehicles are used. Therefore this was the main performance indicator that was aimed to match the data. In the end, it might have been better to give less priority to kilometers and more to delay. This would also result in less ride-sharing which would fit the data better. However, it must be noted that finding the perfect objective function was not the goal. Instead, the aim was to find a reliable benchmark so that the differences in results could be compared with each other.

The fact that all performance indicators are minimized at the same time, may have led to inconsistent results. The vehicles used especially disrupt the comparison of the results. Because in some situations fewer vehicles can be used, the average pick-up delay and traveled distance shoot up because of this result. This makes comparing the strategies or differently-tuned strategies with each other more difficult. Fixing one factor, for instance, the number of used vehicles, and minimizing the rest of the factors could result in better comparable output. However, in real life, the minimization of all the factors makes sense. A taxi company will try to minimize these factors all at once. So doing it this way brings the result closer to reality. Although, admittedly, for pure comparison of strategies, fixing the number of vehicles would have been better.

7.6. Simulation limitations

The goal of a simulation is to create a representation of reality. Generally, the representation never matches the true reality because taking into account every aspect of it is very hard. Also, in this case, simplifications are made compared to reality. One of them is that a constant speed of vessels is assumed. The model received already worked with this constant speed and it is not modified to handle different speeds. A water taxi will not travel as fast through smaller waterways inside the city as it will on open water. Also, with variable speeds, the time lost due to docking could be simulated as well. These features have not been implemented because their influence on the results would have been minimal. On top of that, creating accurate speed profiles throughout the map and while docking would have consumed time that otherwise was used for testing strategies. Another simplification made is that skippers are assumed to be always on duty. In reality, the skippers have breaks that would influence the schedule. These breaks are hard to implement because a long-term schedule is not present. In the end, adding these breaks could result in extra vehicles used during those breaks. Also, during meetings with Flying Fish, it became clear that skippers will normally not wait at a location for long and return to the terminal for a cup of coffee if there is time for that. Adding these practical implications would consume a lot of time and even then the realisticness of it could be questionable. Because in reality, breaks are planned at convenient times that follow from the long-term schedule. A skipper may only go for coffee when he hasn't had one for the last hour, and this could differ from person to person as well. Summarized, the possible implementations would require an amount of effort that would probably not be worth it in terms of creating more realistic results. Therefore the decision was made to not include these and spend the time and effort elsewhere.

7.7. Prediction model assumptions

For the rebalancing strategy based on a forecast, a prediction model was needed. Creating and training an entire prediction model would take too much time and would need a large amount of data, which was not available. Instead, a prediction model could be imitated. However, some assumptions needed to be made to construct this prediction model. The decision was made to create a simple prediction model imitator. One that predicts a percentage of unrevealed future requests. A very complicated and maybe more accurate imitator could be constructed as well but that was not the goal of the research. Instead, it will be analyzed how a rebalancing strategy performs if some of the requests could be predicted.

But how many of the requests should then be predicted? How accurate are prediction models nowadays? X. Song, Kanasugi, and Shibasaki (2016) states the following. "*Given a person's observed mobility history data, they find that the deep LSTM can predict his future movements and transportation mode with over 80% accuracy*". This means that by using LSTM, accuracy percentages as high as 80% accuracy were observed. Another study shows the maximum potential of transport-related prediction models. In the research of C. Song et al. (2010), they state "*The limits of predictability Π_{max} is obtained by measuring the entropy of the human mobility sequence considering both the randomness and the temporal correlation of human movement. They analyze 50,000 users' mobility and find that there is a potential 93% predictability*" This means that the potential of prediction technology is even greater and that those numbers might be reached in the future. However, this high accuracy is only reached with very well-trained models. It is safer to assume a conservative value because too high prediction assumptions may lead to unrealistic results. Eventually, a value of 50% is chosen. This is less than stated in the literature, but still high enough so that a strategy using a prediction can be tested. Another important factor that can be taken into account, is reservations. In the water taxi case scenario, there are a significant number of requests known beforehand. This is because reservations can be made more than a day before pick-up. So in reality, there is already more knowledge about the future available. A percentage of 50% does not look far-fetched when taking this knowledge into account.

8. Conclusion

In this chapter, the main results will be summarized and the research questions that were constructed at the start of this report will be answered. Furthermore, the contributions of this work will be mentioned and the recommendations for future research will be elaborated.

8.1. Conclusions

In conclusion, this research has investigated the performance of different strategies that in theory could improve the performance of dynamic transport systems. The performance can be improved by reducing delays, reducing traveled distance, or reducing the number of vehicles used. Every dynamic transport problem is different, so the real benefits of strategies can only be evaluated by experiments. Consequently, the research question that was formulated at the start of this study, is the following:

How can the operational performance of a water taxi service be improved while maintaining a positive customer experience?

To answer this question, the subquestions need to be answered first.

- What are the system characteristics and requirements of the Rotterdam water taxi system?

The characteristics of the water taxi system in Rotterdam have been extracted by analyzing the data of the system. Not only the number of requests per minute, hour, or week has been found, but also the properties of those requests like start and end location, number of passengers, and pick-up time have been extracted. With this information, an extensive request file could be generated that includes all important information regarding requests. Another important system characteristic is the network of the taxi system. Unlike road vehicles, the network of water taxis is not bound by roads. Water taxis have a lot of flexibility in routing and every path from one location to another is unique. The problem of quantifying this flexibility is overcome by constructing a grid-like network. In this network, the shortest paths can be calculated that resemble the paths taken by the water taxis in the real-life scenario. Using this feature together with the observed duration of trips, a generalized speed of the water taxis can be calculated. These characteristics together give a clear overview of the system and the features found can be used directly in a simulation model for the execution of experiments.

- How can the proposed methods be evaluated under different scenarios of interest?

The experiments that need to be performed have to be done by using a simulation model. This simulation is able to reproduce a water taxi scenario. The simulation handles the incoming requests and matches them to the available vessels. When an entire day is simulated, the performance of the system can be evaluated by looking at specific indicators. In this case, the performance indicators are traveled distance, pick-up delay, and number of vehicles used. These indicators are all properties of the system that an operator wants to minimize. To what extent every indicator is reduced at the cost of increasing other indicators can be a preference of the operator. To create a clear benchmark for comparing the results, a base case is created. The preferences in the base case are adjusted in a way that matches the real-life scenario. Having a simulator model, the performance indicators, and a base case, the strategies can be evaluated and compared with each other in an accurate way.

- In what way can waiting strategies reduce the required fleet size and total traveled distance?

In theory, waiting strategies can reduce traveled distance and reduce the required fleet size by waiting at the right places. To find out if this is also the case for a water taxi scenario, experiments have been performed where these strategies were implemented into the simulation model. From the results of the experiments, it can be concluded that waiting strategies can have a significant impact on the performance of the water taxi system. Depending on the preference of the operator, different strategies can be used. Wait First has proven to be able to drastically decrease the traveled distance. The trade-off here is that the pick-up delay increases a lot as well as needing an extra vehicle. The intensity waiting strategy, on the other hand, does not have that big trade-off. It only increases the pick-up delay by a little and has the same amount of traveled distance, while needing a vehicle less to serve all requests. This strategy improves the system performance overall so this is a good strategy for the water taxi scenario.

- In what way can rebalancing strategies decrease customer wait time and reduce the required fleet size?

Rebalancing strategies improve pick-up delays, but they use distance and vehicles to produce the rebalancing actions. If this is also the case for the water taxi scenario, has been determined with experiments where the strategies were implemented into the simulation model. The results of the experiments show that every rebalancing strategy produces extra traveled distance. This distance increases with an increasing number of rebalancing actions. On the other side, most of them are able to reduce the average pick-up delay of the system. Only the rebalancing strategy based on delay is not able to produce a better outcome than when no strategy is applied. The other strategies all have a similar trade-off. When the pick-up delay is decreased, the traveled distance is increased. The intensity strategy is able to produce the best trade-off. Where the increase in distance results in the most delay decrease. If the goal is to reduce the number of vehicles used, rebalancing to the top 2 most popular locations is a good option. This strategy was able to use a vehicle less with only a slight increase in delay and an increase in distance traveled. Depending on the preference of the operator, the best-performing strategy can be implemented.

- How can these methods be integrated successfully?

Waiting strategies and rebalancing strategies are different methods, with each having its own characteristics. What could be even more interesting than implementing the strategies apart from each other, is the combination of waiting and rebalancing. Rebalancing, for instance, was not able to reduce the distance traveled. Instead, it only increased the distance because rebalancing actions require the vehicle to move. This effect could be compensated by combining a rebalancing strategy with Wait First. The results of the combination experiments show that, indeed, the strategies can compensate for each other. The overall performance of the system is not affected by the combination of 2 strategies. The solution only leans more towards a certain performance indicator. This means that the combination of strategies can lead to more nuanced results and gives the operator more flexibility in modifying the outcome of the system.

- What is the benefit of proposed methods under different scenarios of interest?

The benefit of a method is dependent on the preference of the outcome of the system. If there is an interest in improving all performance factors of the system without any form of trade-off, then the methods do not impose a benefit. However, some methods, like the Intensity waiting strategy do come very close to this interest, where there is a small trade-off and a relatively large benefit.

If the interest is prioritized in decreasing the delay, this can be done with rebalancing strategies and if the interest is prioritized to decrease traveled distance, the Wait First strategy is a good choice. When there is a higher interest in decreasing delay, for instance, the objective function can be adapted to reflect this interest. In Section 7.2 it is explained, however, how these changes in objective function may have unexpected effects. The sensitivity analysis performed there resulted in the conclusion that the performance of strategies can be sensitive to small changes in the objective function. This sensitivity can be decreased by increasing the planning horizon, a solution to many of the model's limitations, but also a large task was outside the scope of this research, but an interesting topic for future research.

These answers combined together form the answer to the main research question. The operational performance can be improved by implementing rebalancing and waiting strategies. According to the preference of the operator, a larger sacrifice in distance traveled can be made by choosing certain rebalancing strategies. Furthermore, this can be combined with waiting strategies to compensate for some of the extra distance, or to potentially save on a vehicle that would otherwise be used.

8.2. Contributions

This work has made several contributions to the field of research on dynamic transport systems. They are listed below.

- By answering the research question, this work has connected the theory of rebalancing and waiting strategies with a mixed-request waterborne environment. Mixed requests involve dynamic on-demand requests as well as reservations that are made in advance.
- A simulation model has been created that can handle these mixed requests simulations and evaluate the results. The model is formulated in a flexible way that enables further extensions, meaning that even more waiting or rebalancing strategies can be added and evaluated. It also has the ability to combine waiting and rebalancing strategies without interference.
- A new rebalancing strategy has been constructed. This strategy uses the intensity component found in the intensity waiting strategy. This intensity is then further modified to better fit a rebalancing strategy. The strategy has multiple tuning possibilities to return the best results for various scenarios. This strategy is not limited to this case only. It can be applied in every dynamic vehicle routing problem as long as historical requests are available.

8.3. Recommendations

One of the recommendations for future work is finding a way to assign requests further in the future. Currently, the requests can only be assigned 10 minutes beforehand. The limiting factor is the number of RTV graphs that need to be generated for every timestep. As explained in Section 7.3, increasing the planning horizon requires more RTV graphs and larger ones. If a way can be found to avoid these multitudes of RTV graphs, a longer planning horizon may be possible. Future research can be focussed more on creating an assignment method that accounts for dynamic and planned requests in a more efficient way, allowing for a longer planning horizon. Flying Fish mentioned that they use a method that is suggestion-based. Implementing a method that uses the same principle might be a solution to this problem. With a longer planning horizon, requests can be observed earlier which could result in a better overall assignment. As explained in Section 7.2, a longer planning horizon will also improve the robustness of the model because decisions are not based anymore on short-term optimality. Small changes in the objective function would have a smaller and more predictable effect on the results. Moreover, the true potential of waiting strategies could be observed because a vehicle could be assigned to a request for a long amount of time.

Another aspect that could be improved is testing the model on more data. In this work, only one week of data was available. This dataset was recorded in a time that was not a very busy period for the water taxi service. Another dataset may become available in the future. This dataset, as opposed to the current one, would be recorded in a busy period. It would be interesting to see how the model and the strategies would behave in a different scenario. Also, the extra set of data could be used to train the strategies more, resulting in a strategy that can be tuned for multiple scenarios.

Lastly, the optimization problem formulation could be improved in future work. The objective function is now based on single values that have been determined at the start of the experimental phase. The objective function was kept the same to see how the different strategies could influence the outcome. This objective function, however, influences the most what the outcome would be. When a rebalancing strategy adds distance and reduces delays, maybe the objective function could be adapted to prioritize the minimization of traveled distance to compensate for this. This would mean an entirely new tuning procedure for every different strategy. Also, more time could be spent on determining the most realistic values for the objective function weights. A more extensive analysis could create an accurate scoring function to quantitatively rank the strategies. Because now the choice of a strategy is still very dependent on the preference of the operator.

To sum up, this thesis has presented a comprehensive analysis of waiting and rebalancing strategies for urban waterborne transport, highlighting their potential to improve the efficiency of water transportation systems. The findings of this research contribute to the research on innovative solutions for urban mobility and provide a basis for further exploration and implementation of these strategies in real-world contexts.

A. Paper

(See next page)

Waiting and rebalancing strategies in an urban waterborne transport environment

Daan van Gisbergen, Bilge Atasoy, Breno Alves Beirigo

***Abstract*— This paper explores waiting and rebalancing strategies in urban waterborne transport, with a focus on the enhancement of the water taxi system in Rotterdam. To start, the existing literature is analyzed and relevant waiting and rebalancing strategies are highlighted. Based on the characteristics of the water taxi system in Rotterdam, different strategies are developed and tested in an enhanced simulation environment that can handle both on-demand requests and reservations. By performing simulations with various strategies, important findings are achieved. With waiting strategies, the distance traveled by empty vehicles can be reduced by 37%, or the number of vessels is reduced without a major trade-off in delay or traveled distance. On the other hand, by using rebalancing strategies, the average pick-up delay can be decreased by 36% while using the same number of vessels. Overall, this paper provides valuable insights into the challenges and opportunities of waterborne mobility and the strategies that can be used to optimize its efficiency and effectiveness.**

I. Introduction

Urban waterborne mobility, the movement of people and goods through waterways in cities, has been gaining attention as a sustainable transportation alternative (Tanko and Burke, 2017). With the increasing population density in cities and the need for efficient transportation options, the use of waterways is becoming more important. Waterborne mobility offers numerous advantages over traditional modes of transportation, including reduced traffic congestion and faster travel times. In recent years, many cities have implemented initiatives to develop and promote waterborne mobility (Cheemakurthy, 2017). This has resulted in the establishment of new waterway transport systems, such as ferries and water taxis, as well as the revitalization of existing waterways. These initiatives aim to reduce the negative impacts of transportation on the environment.

One of those initiatives is a project done by Flying Fish (*Flying Fish* 2023) in Rotterdam. They have modernized the water taxi system in multiple ways. Flying Fish has developed a water taxi Operations System that handles all position and motion data, as well as incoming requests. The system successfully services thousands of people on busy days.

A. Problem formulation

Efficiency improvements in dynamic transport systems are an ongoing pursuit. Waiting times experienced by passengers often result from inefficiencies caused by supply and demand imbalances within the system. For transport companies, the challenge lies in enhancing their responsiveness while minimizing added costs. Numerous methods have been devised to boost system performance without altering the existing equipment.

Two notable approaches to improve efficiency are rebalancing and waiting strategies. A rebalancing strategy seeks to optimally distribute idle vehicles across the service area, facilitating quicker responses to new requests as the required vehicles are already nearby. Conversely, waiting strategies aim to make the most of slack time in the route—periods when one customer has been served, and the next request is not immediate. By waiting strategically, a vehicle can serve more requests within the same timeframe. In this study, various variants of these strategies will be examined and compared.

To assess the impact of these strategies within the context of the Rotterdam water taxi scenario, a simulation is essential. This simulation should incorporate spatial data from the Rotterdam water taxi service area and mirror real-world request handling as closely as possible. This research encompasses the creation of such a simulation, followed by the testing of selected waiting and rebalancing strategies using this simulation. To guide these efforts, the research question is defined:

How can the operational performance of a water taxi service be enhanced while ensuring a positive customer experience?

To address this question, the existing literature will be reviewed, data provided by Flying Fish will be analyzed to define system characteristics, and a simulation model will be established. Subsequently, the strategies will be implemented into the model, allowing for their analysis and the drawing of conclusions.

B. Existing Literature

1) Waiting strategies: Techniques that can be used to improve serviceability and reduce the required fleet size in vehicle routing problems (VRP) or pickup and delivery problems (PDP) are waiting strategies. In most dynamic transportation systems, vehicles face slack time during their service. This occurs when a request is completed and the vehicle does not immediately have to service another request. In a dynamic environment, waiting during some moments might be more beneficial than others. This has to do with possible incoming requests that require rescheduling that could lead to a better solution. Waiting strategies aim to distribute this time throughout the service optimally. In this section, some waiting strategies will be elaborated.

a) Drive First: The Drive First (DF) strategy requires a vehicle to always drive if possible. In other words, to leave a location at the earliest departure time. Let i be a location with a customer with time window $[a_i, b_i]$. For a vehicle traveling towards location i, let $\underline{A}_i$ be the earliest possible arrival time and $\overline{A}_i$ be the latest possible arrival time. When DF is applied, the scheduled arrival time A_i will be equal to $\underline{A}_i$. The vehicle does wait at the arrived location i if the location is not open for service yet. The waiting time at location i before the service is equal to $max\{0, a_i - \underline{A}_i\}$. The waiting time after the service is always zero. The DF strategy is commonly used when transporting letters or parcels, for instance, see Benyahia and Potvin (1998), Gendreau et al. (2006). This is because these situations are static vehicle routing problems. Drive First is the only appropriate strategy for a static vehicle routing problem because, without dynamism, the earliest possible arrival is always preferred.

b) Wait First: Opposite to the DF strategy, Wait First (WF) requires a vehicle to always wait at the current location as long as possible. It means that the vehicle leaves the location at the latest possible departure time. For the same customer location i, with a time window $[a_i, b_i]$ and a vehicle with earliest possible arrival time $\underline{A}_i$ and latest possible arrival time $\overline{A}_i$, the arrival time A_i will always be the latest possible $\overline{A}_i$. The location is always serviced immediately and the vehicle is allowed to wait at the current location. For the WF strategy, the earliest possible departure time $\underline{D}_i$ from location i and the latest possible departure time $\overline{D}_i$ is introduced as well. The waiting time at location i is then given by $max\{0, \overline{D}_i - b_i\}$. WF has the advantage over DF in that it has the potential to build shorter routes. By waiting longer it can incorporate changes in requests and has a longer period to optimize routing. The disadvantage is that WF requires more vehicles to service the same set of locations. This has to do with the fact that WF concentrates a lot of the waiting time in the beginning, leaving too little time in the end. It is therefore unlikely that the vehicle is able to service requests that appear after it leaves its starting position. This requires more vehicles to complete the service.

c) Intensity waiting strategy: The intensity strategy was proposed by Vonolfen and Affenzeller (2016). Their strategy, applied on a PDPTW, moves away from the dependence on stochastic knowledge because of the fact that stochastic knowledge may require intensive pre-processing steps to result in high-quality predictions (Ferrucci, Bock, and Gendreau, 2013). Their strategy introduces a new variable on which decisions can be based: the intensity measure, which is why this strategy will be called the *intensity* strategy. This measure aims at removing the restrictions that a stochastic model imposes.

The intensity measure is based on the transition time of historical requests. A service request is defined as $s \in S$ where S is the total set of service requests. The request s has a reveal time r_s, a pick-up location l_s^p, and a drop-off location l_s^d. The transition time is the total amount of time that is between the decision time, and the beginning of service on the next location. Potential waiting time due to time windows is included in this transition time. The transition time is given by the following formula.

$$\begin{aligned} TransTime(l_1, l_2, t) &= dist(l_1, l_2) \\ &\quad + max\{0, o_{l_2} - (t + dist(l_1, l_2))\} \end{aligned} \tag{1}$$

Where $dist(l_1, l_2)$ is the distance between the two locations. The current time is denoted as t and the opening time of the time window of location l_2 is denoted as o_{l_2}. The derivation for the intensity measure is displayed in Equation 2.

$$Intensity(l, t) = 1 - \frac{\sum_{\{s \in S : r_s > t\}} \frac{TransTime(l, l_s^p, t)}{h}}{|\{s \in S : r_s > t\}|} \tag{2}$$

The intensity is defined by the average transition time for unrevealed requests in the historical set S. The transition time is normalized by the planning horizon h. The intensity of a location is high if the averaged transition time to potential future requests l_s^p is low. This results in a high intensity if there are nearby historical requests, and a low intensity corresponds to historical requests that are far away.

The *intensity* strategy will make the decision on whether to wait at a given location R_i in a given route R or to move on to the next location R_{i+1} at time t_i based on three factors:

- Transition time
- Intensity
- Slack time

These 3 factors are combined into a single value v to make the decision if the vehicle should wait at the given location R_i. The calculation is displayed in Equation 3.

$$\begin{aligned} v(R_i, t_i) = \alpha' \cdot \frac{TransitionTime(R_i, R_{i+1}, t_i)}{(h - t_i)} \\ + \beta' \cdot \frac{Intensity(R_i, t_i)}{Intensity(R_i, t_i) + Intensity(R_{i+1}, t_{i+1})} \\ + \gamma' \cdot \frac{Slack(R_i, t_i)}{(h - t_i)} \end{aligned} \tag{3}$$

The Slack function returns the slack time, which is the maximum time the vehicle is permitted to wait without violating time windows. The values for transition time and slack time are normalized by the time remaining in the planning horizon. The intensity is divided by the sum of the current location's intensity and the intensity of the next location. This results in all factors having a range of [0,1]. Each of the 3 factors is weighted by the tuning parameters α', β', γ'. Making room to give some factors more importance than others. The value function gives a number to the value of waiting at a certain location during a certain point in time. This value is then further used to calculate the waiting time with. An extra threshold parameter ϵ is used for this. The exact equation can be found in the work of Vonolfen and Affenzeller (2016).

2) Rebalancing strategies: Vehicle rebalancing is described as the movement of idle vehicles to preferred locations within the service area. A vehicle is considered idle when it is not actively serving a customer or moving toward a customer. Through rebalancing, waiting time for customers can be decreased and serviceability can be increased, as well as the utilization of the fleet. Different strategies for rebalancing have been developed in numerous studies. A selection of strategies will be evaluated in this section. Some of them are more comprehensive than others.

a) Rebalance to desired distribution: When no information about future demand is available, static rebalancing strategies like presented by Pavone et al. (2012) are needed. Their idea is that there is some desired distribution of vehicles $v_i^d(t)$ over nodes $i \in \mathcal{N}$. Where $\mathcal{N}$ is the set of nodes in the service area. This distribution can be anything the user wants to insert and can be, for instance, the result of an empirical study. Pavone et al. (2012) demonstrate a possible distribution where all idle vehicles, also called excess vehicles $v_i^{\text{excess}}(t)$ are distributed equally over all nodes. They do this by defining the set of all vehicles as V and all customers waiting at node i is defined as $\sum_i c_i(t)$. The desired distribution is then given by Equation 4.

$$v_i^d(t) = \left\lfloor \frac{V - \sum_j c_j(t)}{n} \right\rfloor \quad \text{for each node i} \tag{4}$$

Where n is the number of nodes in the service area. This equation ensures that the desired number of vehicles stationed at a node adapts to the number of excess vehicles in the service area. This desired distribution can then be inserted in the linear program that aims to optimize this transition to the desired distribution. The linear program LP is given by:

$$\begin{aligned} &\text{minimize} \sum_{i,j} \tau_{ij}\text{num}_{ij} && \text{(5a)}\\ &\text{subject to: } v_i^{\text{excess}}(t) + \sum_{i \neq j}(\text{num}_{ji} - \text{num}_{ij}) \geq v_i^d(t) && \text{(5b)}\\ &\qquad\qquad\qquad\qquad\qquad\qquad \forall i \in \mathcal{N} &&\\ &\qquad \text{num}_{ij} \in \mathbb{N} \qquad \forall i,j \in \mathcal{N}, i \neq j && \text{(5c)} \end{aligned}$$

Where τ_{ij} represents the costs to travel from node i to node j and num_{ij} is the number of vehicles that travels from node i to node j. This linear program minimizes the rebalancing costs while making sure that every node in the service area has the desired number of excess vehicles.

b) Rebalance to failed request locations: Rebalancing the idle vehicles of the system to locations where requests have been rejected is another form of rebalancing without knowledge of future demand. It is proposed by Alonso-Mora, Samaranayake, et al. (2017), who use this strategy in an on-demand high-capacity ride-sharing scenario. Although this method does not explicitly incorporate predicted demand, it keeps in mind that a location with a failed request has the possibility that another request in that area can not be fulfilled with the current system state. It would improve serviceability to send an idle vehicle to that location to await future requests.

Alonso-Mora, Samaranayake, et al. (2017) define the set of undefined requests $\mathcal{R}_{ko}$ and idle vehicles $\mathcal{V}_{idle}$. Also, $\tau_{v,r}$ which is the travel time between a vehicle $v \in \mathcal{V}_{idle}$ and failed request $r \in \mathcal{R}_{ko}$. The variable $y_{v,r} \in \mathcal{Y}$ indicates the individual assignment of rebalancing vehicle v to failed request r. The linear program to optimally rebalance the idle vehicle looks as follows:

$$\begin{aligned} &\text{minimize} \sum_{v \in \mathcal{V}_{idle}} \sum_{r \in \mathcal{R}_{ko}} \tau_{v,r} y_{v,r} && \text{(6a)}\\ &\text{subject to: } \sum_{v \in \mathcal{V}_{idle}} \sum_{r \in \mathcal{R}_{ko}} y_{v,r} = \min(|\mathcal{V}_{idle}|, |\mathcal{R}_{ko}|) && \text{(6b)}\\ &\qquad 0 \leq y_{v,r} \leq 1 \qquad \forall y_{v,r} \in \mathcal{Y} && \text{(6c)} \end{aligned}$$

Where the $|...|$ denotes the number of elements of a set. This linear program produces the shortest rebalancing trips. Constraint b makes sure that the amount of rebalancing trips is either equal to the amount of failed requests, or to the number of idle vehicles. Whatever is less. The inputs $\mathcal{V}_{idle}$ and $\mathcal{R}_{ko}$ are the result of the batch assignment that was computed before the rebalancing phase. The strategy described here is reactive. Meaning that the system only reacts to current states and not to future states. It recognizes where the system has failed and reacts to it by sending vehicles over. It does not, however, do anything to prevent the same problem from happening in the long-term future.

c) Rebalancing based on forecast: The strategy proposed by Iglesias et al. (2017) is the first strategy being discussed here that incorporates forecasting of demand. In their research, they compute their predictions using a Long Short-Term Memory (LSTM) neural network (Hochreiter and Schmidhuber, 1997). At a given time t_0 the available vehicles are observed and stored in the set $\mathcal{S}$. Also the outstanding customer demand λ_{ij0} is obtained. These 2 inputs can be observed in the current system state. Next, the future customer demand $\hat{\Lambda}_{t_0,t_0+T_{forward}}$ for the next $T_{forward}$ time steps is predicted. This is done using a prediction model f, trained with historical data θ_t. With these inputs the problem 7 can be solved.

Iglesias et al. (2017) define $w_{ijt} \in \mathcal{W}$ as the decision variable denoting the number of outstanding customers who wish to travel from i to j, and are picked up at time t. The slack variable $\mathcal{D} = \{d_{ijt}\}_{ijt}$ denotes the number of predicted customers that remain unsatisfied. c_{ijt}^w and c_{ijt}^d are the cost related to letting an outstanding customer wait, and the cost for not servicing a predicted customer. Similarly, the cost for rebalancing is c_{ijt}^y and it is related to the distance that is traveled.

To solve the optimal rebalancing strategy, a mixed-integer linear program (MILP) is used which is shown below.

$$\text{minimize} \sum_{(ijt)} c^y_{ijt} y_{ijt} + c^w_{ijt} w_{ijt} + c^d_{ijt} d_{ijt} \tag{7a}$$

$$\text{subject to: } x_{ijt} + d_{ijt} - w_{ijt} = \hat{\lambda}_{ijt} \qquad \forall i,j \in \mathcal{N}, t \in \mathcal{T} \tag{7b}$$

$$\sum_{j \in \mathcal{N}} x_{ijt} + y_{ijt} - x_{ijt-\tau ij} - y_{ijt-\tau ij} = s_{it} \qquad \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{7c}$$

$$\sum_{t \in \mathcal{T}} w_{ijt} = \lambda_{i,j,0} \qquad \forall i,j \in \mathcal{N} \tag{7d}$$

$$x_{ijt}, y_{ijt}, w_{ijt}, d_{ijt} \in \mathbb{N} \quad \forall i,j \in \mathcal{N}, t \in \mathcal{T} \tag{7e}$$

The objective function in Equation 7a aims to minimize the costs related to rebalancing, customers waiting, and predicted customers rejected. The constraints ensure that requests are either assigned or rejected and that the flow of vehicles is constrained. This MILP is more extensive than the LPs we have seen so far. It does not only include future predicted requests in the rebalancing decision-making, but it also allows tuning the parameters.

II. Methods

A. System characteristics

Decisions made in a specific system can cause great improvements, but deteriorate the solution in another system. This is why it is important to analyze the problem thoroughly before methods can be applied. The system characteristics of water taxi Rotterdam consist of dock locations, demand, trip statistics like durations and ride-sharing, and the network. These characteristics have to be extracted from the data that is provided by Flying Fish. The steps that have been taken to do this are described in this section.

1) Data structure: The data provided by Flying Fish is a copy of the water taxi database for the period of one week. Every interaction with the system, change in planning, or update of estimated times is stored in this database. The data was shared as a MySQL dump file. This was an unfamiliar file and not easy to work with. To overcome that, the data was first parsed to a CSV file and after that, constructed as a Pandas DataFrame using Python. In the process, the data has been compressed in a way that a request only takes up one row of the DataFrame. The attributes of a request are shown below.

- request ID
- Inquiry time
- Number of passengers
- Start location ID
- End location ID
- Start time
- End time
- Fleet ID
- Type

This structured data made acquiring the information needed to construct the system characteristics possible.

2) Locations and demand: Watertaxi Rotterdam offers its service to 248 locations in and around Rotterdam. The locations are specified with the name, the ID number, and the coordinates. In one week, 54 locations were used. The coordinates can be used to project the locations on a map. The demand per location is needed to determine how active locations are during the day and how they differ. Some locations will have demand over the whole day while others have spikes in demand during the morning and evening. The average demand over the whole week per location is visible in Figure 1

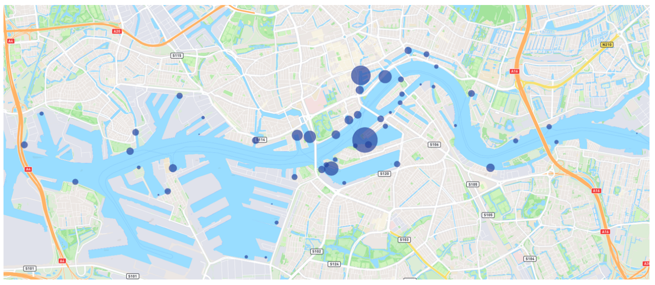


Fig. 1. The locations used by the watertaxis, where the size of the circle is proportional to the number of requests emerging from the location.

3) Ride-sharing: Ride-sharing is a way to merge multiple requests into one trip. To know how many of the rides are shared, a clear distinction needs to be made. The data received was only structured in requests. To see whether some of these requests were merged into one ride needed extra investigation. A ride is labeled as shared once the travel interval of successive requests for a single taxi overlaps with each other. This results in 119 requests that have been shared over one week. Compared to all 1232 requests in that week, this results in a ride-sharing percentage of about 10%.

4) Network: The water taxi system can be seen as a large network. Where the docks are nodes and the paths between the docks are links. If we can create a virtual model of this network, calculations can be made to reveal more properties about trips. Moreover, if there is going to be a simulation of the water taxi operation, a network can serve as an input for that simulation as well.

Boats on the river can choose their path and are not bounded by roads. To simulate the feature of having the freedom of choosing an undetermined path in a bounded area, a grid-like network is needed. This grid will cover the area over which the taxi will operate, in this case, the water. Docks will have to be assigned to nodes on this grid and also the smaller waterways should be reachable using this grid size. The nodes should therefore be spaced close enough together to provide an accurate representation of the real world. When the operational area is determined, a grid can be generated over this area. It was decided to use the Hexagonal Hierarchical Spatial Index, or H3, developed by Uber (Brodsky, 2018). The H3 grid generates points, and between these points, links can be generated to create the network, as is shown in Figure 2.

The key information we would like to extract from the

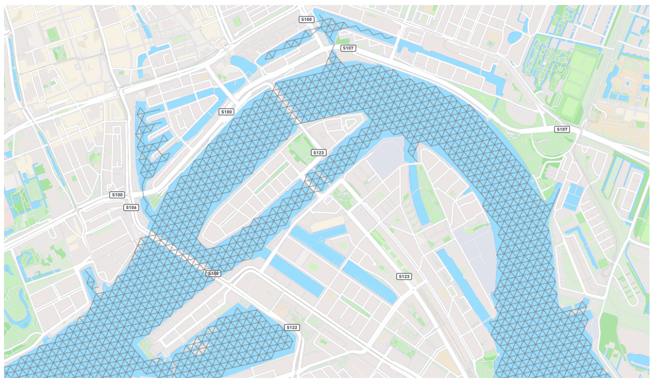


Fig. 2. The network is shown as links. A center node has a link with each of its six neighboring nodes.

network is how much distance a vessel travels from one specified dock to another. This is useful information because it can ultimately lead to an estimation of the duration of trips. To find the distance, paths must be generated that travel across the network. Using network calculations, the shortest paths between docks can be found. The paths generated by using the week of data are visualized in Figure 3.

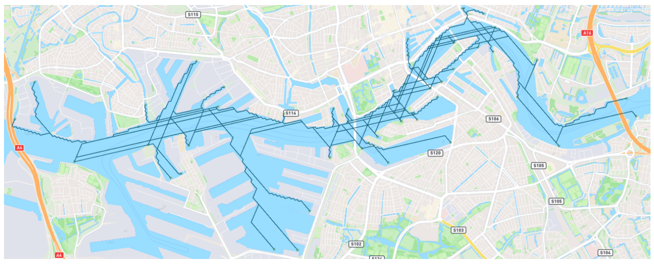


Fig. 3. The generated shortest paths for all recorded trips, projected on the map.

B. Experimental setup

With the relevant data extracted, an environment can now be set up to test the strategies. To get relevant and comparable results from the experiments, the experimental setup needs to be close to reality. However, the computational expense of the experiments can not be too long because otherwise the tuning and testing of strategies can not be completed within the project's time.

A working simulation model was provided by Breno Alves Beirigo. This model was already applied to simulate a taxi service in Manhattan, New York. The model is used for simulating a Mobility on Demand (MoD) service. In this service, a request is always dynamic, meaning that the pick-up time is immediate. Since the water taxi service handles dynamic requests as well as planned requests, the structure for handling and assigning the requests needs modification. The most important parts of the model are the request handling structure, the matching structure, and the assignment ILP. These 3 parts will be briefly explained, followed by the benchmark settings.

1) Request handling: The requests that enter the system are selected according to their reveal time. This is the time that corresponds to the moment when a new request makes its first interaction with the system. The request then gets checked if the duration until pick-up is longer or shorter than a certain threshold t_{max}. If it is shorter, the request is directly inserted into the *unassignedUsers* group, meaning that it is ready for assignment in the next round. If the duration is longer, the request is stored temporarily until the threshold is reached. This ensures that the system is not overloaded with requests that are planned for the far future. The procedure is also shown in Figure 4.

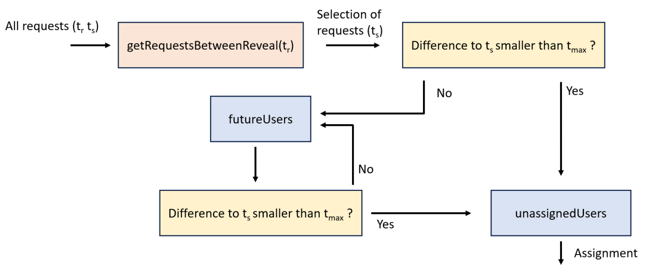

All requests (t, t_s)
getRequestsBetweenReveal(t_r)
Selection of requests (t_s)
Difference to t_s smaller than t_max ?
No
Yes
futureUsers
No
Difference to t_s smaller than t_max ?
Yes
unassignedUsers
Assignment

Fig. 4. The method for generation of requests, where a distinction is made between current and future requests.

2) Matching: The requests that are ready for assignment will first be, together with the time, inserted as input for an RTV graph. The RTV graph is introduced by Alonso-Mora, Wallar, and Rus (2017). The RTV graph works as a pre-processing method for the final assignment ILP. In the graph, vehicles that can reach requests given the time constraints are paired. Also, requests that could be served together are paired as well. This leaves Vehicle-Request pairs (VR) and Request-Request (RR) pairs. In the next step, the pairs are connected if possible. In this step, trips will be created. These consist of the VR pairs and additional combinations that can be made. If a vehicle can serve both requests of an RR pair, then this will become a trip as well for instance. An RTV graph for the current time step and the future 10 time steps are made and the graphs are merged into one. This enables assignment in future timesteps, resulting in a short-term planning. The procedure is visualized in Figure 5.

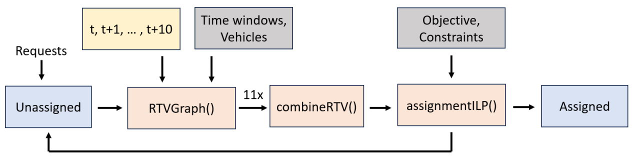


Fig. 5. A modification on the matching method. Here, future assignment is possible by providing future RTV graphs and combining them into one.

3) Assignment ILP: At the end of the matching phase, the definitive choice of which vehicle will serve which request will be made by the assignment ILP. The RTV graph has ensured that this optimization problem is as small as possible, resulting in a quick solution. The input for the ILP is the combined RTV graph that includes all the feasible trips, together with the constraints and the objective function of the problem. The ILP is described below.

$$\begin{aligned}
\text{maximize: } & \sum_{i\in\mathcal{T}} r_i x_{ij} - \sum_{i\in\mathcal{T},j\in\mathcal{V}} c^d \cdot d_{ij} x_{ij} \\
& - \sum_{i\in\mathcal{T},j\in\mathcal{V}} c^s \cdot s_{ij} x_{ij} - \sum_{j\in\mathcal{V}} c^v \cdot v_j x_{ij} && \text{(8a)} \\
\text{subject to: } & \sum_{j\in\mathcal{V}} x_{ij} \leq 1 \qquad \forall i \in \mathcal{T} && \text{(8b)} \\
& \sum_{i\in\mathcal{T}} x_{ij} \leq 1 \qquad \forall j \in \mathcal{V} && \text{(8c)} \\
& x_{ij} \in \mathbb{N} \qquad i \in \mathcal{T}, j \in \mathcal{V} && \text{(8d)}
\end{aligned}$$

In this ILP, the number of requests served is maximized. Additionally, the produced delay d, the traveled distance s and the number of vehicles used v are minimized. The last three each have a weight c added so that the priority of each attribute in the solution can be modified.

4) Base case: In order to do experiments with different strategies and analyze how the results are affected, a clear benchmark needs to be set. This benchmark will be defined as the base case. This is the case where no extra strategies are applied to the model. The settings of this base case will be conserved through the entire testing phase. The settings of the base case are summarized in Table I.

TABLE I
THE SETTINGS OF THE SIMULATION MODEL THAT DEFINE THE BASE CASE.

Model setting	**Value**
Available vessels	18
Vessel capacity	12 pers.
Vessel speed	47 km/h
Dynamic request threshold	5 min
Dynamic request time window	15 min
Planned request time window	3 min
Planning horizon t_{max}	15 min
Number of RTV graphs	10
Objective function parameters:	
Requests	1
Delay	$-2.2e^{-5}$
Distance	$-2.3e^{-6}$
Vehicles	$-5e^{-3}$
Rebalancing strategy	None
Waiting strategy	Drive First

C. Proposed strategies

1) Waiting strategies: Before any strategies can be implemented, the concept of waiting needs to be introduced to the model. Because the model previously served as a MoD simulator, waiting while a request is assigned is never done by the model. After analysis of the model, it was decided that the best place to insert a waiting method is after the assignment and right before the fleet and request update. This is because an assignment is needed to determine if waiting is preferred, and it should happen before any route updates because otherwise, the vehicle would have already moved. The insertion of a waiting method is visualized in Figure 6. Here, the simulation model will only execute the

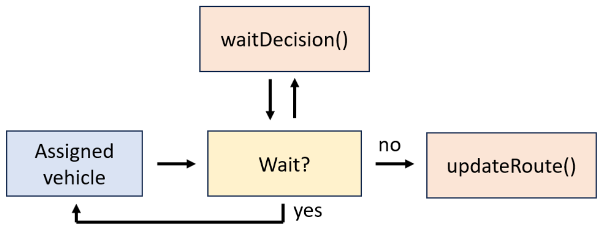


Fig. 6. The implementation of waiting on the model. A waiting method is inserted between the assignment of vehicles and route updates.

route update if the vehicle does not have to wait. The wait decision method decides if the vehicle can start with its route or not. By doing it in this way, the vehicle stays assigned but is not moving. The wait decision method will change according to what strategy is being used. With drive first, the wait decision will always answer no. With Wait First, the latest departure is calculated first. If the current time is before the latest departure, the answer of the wait decision method will be yes. Otherwise, it will be no. For the intensity waiting strategy, the value of waiting will be calculated as described in Equation 3. If this value is higher than a certain threshold, the vehicle will wait. If it is lower than the threshold, the vehicle will follow its normal route procedure.

2) Rebalancing strategies: Rebalancing is implemented into the model right after the main assignment and before the fleet and requests update. The rebalancing assignment is an addition to the main assignment, where idle vehicles can be given a task if necessary. The procedure for rebalancing is displayed in Figure 7. First, a list of potential targets is defined. This list is dependent on what rebalancing strategy is used. Another list is created that contains the location ID of every location where a vehicle is located or will be in the future. Both of the lists are compared with each other.

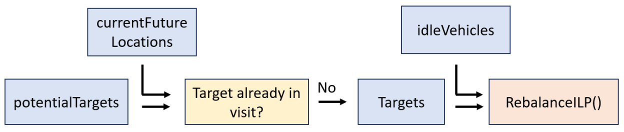


Fig. 7. The procedure for rebalancing. It shows how potential rebalancing targets are filtered and inserted into the rebalancing ILP.

If a vehicle is already located on a potential rebalancing target, it would not make sense to send another one. Also if a vehicle has the potential target in its path already, sending another vehicle to it would result in 2 vehicles arriving there in the future. Therefore, it must be checked if a potential target is not already in the $currentFutureLocations$ list. When it is not, the potential target becomes an actual target and serves together with the idle vehicles as input for the rebalancing ILP. The rebalancing ILP, shown in Equation 9, is very similar to the main assignment ILP.

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in \mathcal{V}_{\text{idle}}, j \in \mathcal{T}} c^x_{ij} x_{ij} && && \text{(9a)}\\
\text{subject to:} \quad & \sum_{i \in \mathcal{V}_{\text{idle}}} x_{ij} \leq 1 && \forall j \in \mathcal{T} && \text{(9b)}\\
& \sum_{j \in \mathcal{T}} x_{ij} \leq 1 && \forall i \in \mathcal{V}_{\text{idle}} && \text{(9c)}\\
& \sum_{v \in \mathcal{V}_{\text{idle}}} \sum_{r \in \mathcal{T}} x_{ij} = \min(|\mathcal{V}_{\text{idle}}|, |\mathcal{T}|) && && \text{(9d)}\\
& x_{ij} \in \mathbb{N} && \forall i \in \mathcal{V}_{\text{idle}}, j \in \mathcal{T} && \text{(9e)}
\end{aligned}$$

The input $\mathcal{V}_{\text{idle}}$ is the set of idle vehicles and $\mathcal{T}$ is the set of targets as defined by the strategy. The ILP will find the optimal assignment where the least distance is traveled while rebalancing to all targets.

In the experiments, 4 different rebalancing strategies will be tested. Rebalancing to location is implemented by inserting a fixed set of potential targets into the procedure every time step. Rebalancing based on delay is achieved by generating potential targets based on a certain delay threshold. Rebalancing based on forecast is done by using a prediction model imitation. The predicted requests are used as potential rebalancing targets. Additionally, a new strategy is created where rebalancing is based on the intensity measure from the intensity waiting strategy explained in paragraph I-B.1.c. In this strategy, potential rebalancing targets are generated by observing what location has the highest intensity. There is also an intensity threshold involved where no potential targets are generated if the intensity of that location is too low.

III. Results

A. Waiting strategies

The results in Table II show the performance indicators of the simulations executed with the different waiting strategies. Drive First is the base case. It has the lowest average pick-up delay of all waiting strategies. On the other hand, it also results in the most distance traveled, which makes sense. When a vehicle always drives to the next location without thinking, it is inevitable that in some cases more distance is traveled than needed. Waiting naturally does not produce traveled distance, making this a more conservative approach in terms of distance. Wait First, however, does perform the least of all waiting strategies overall. It does perform the best in the traveled distance, but the need for an extra vehicle and the large increase in pick-up delay is not good for performance. The Intensity waiting strategy finds a middle ground between these two strategies for some performance factors. It still has a higher average delay than Drive First but not as high as Wait First. Where the intensity strategy stands out is the number of vehicles used. By waiting at the right locations it was able to reduce to 7 vehicles used. And even with that decrease in used vehicles, the amount of distance traveled is slightly reduced as well. Overall, this strategy scores the best and the increase in delay would in most cases outweigh the decrease in the use of vehicles and distance traveled.

B. Rebalancing strategies

Rebalancing strategies overall, will increase the total traveled distance of the system. This makes sense because adding extra trips that do not directly serve customers adds extra empty distance by definition. The goal of the strategy is to trade the cost of added distance with an improvement in another performance factor. In most cases, this is done by reducing the average pick-up delay. The rebalancing strategy based on delay does not perform well in this environment. It is the only strategy that is not able to decrease the average pick-up delay and not improve other performance factors of the result. The other strategy that does not improve the delay is the rebalancing strategy based on the top 2 locations. This strategy, however, does improve an important performance factor: the number of vehicles used. All rebalancing strategies show the same correlation between the number of rebalancing actions and the distance traveled. The intensity rebalancing strategy can perform the best in terms of average pick-up delay. The location top 3 strategy is the best contender after that but is still outperformed in terms of delay and distance. The forecast rebalancing strategy reduces the pick-up delay the most apart from the intensity strategies, but to do so it does need an extra vehicle. The strategy that was able to decrease the delay while adding the smallest amount of distance is the location top 1 rebalancing strategy.

C. Combinations

The different results from rebalancing and waiting strategies give a good insight into the characteristics of the strategies, but what could also be interesting is to look at how combinations of these strategies can influence the results. Strategies with different characteristics could balance each other out or could improve their performance on specific factors even more. The first combination is done with the Wait First strategy and all of the rebalancing strategies. The results are displayed in Table III. It is clearly visible that the Wait First strategy has its impact on the results compared to Drive First as a base. The characteristic of Wait First was that the pick-up delay was increased, but the traveled distance was decreased. This characteristic is combined with the results of the rebalancing strategies. For every rebalancing strategy, the delay has increased and the distance traveled is decreased. However, since the rebalancing strategies did the exact opposite, this results in some interesting output. Every rebalancing strategy now results in a lower traveled distance than the base case. So even though the number of rebalancing actions may be high, the traveled distance only decreases.The results for the intensity wait strategy and the rebalancing strategies is displayed in Table IV.

The combination of strategies is less one-sided than Wait First combined with the rebalancing strategies. In pick-up delay, there is not a general decrease or increase observed for all strategies. The combination sometimes results in more delay and sometimes also in less. In 2 combinations, the

TABLE II

THE RESULTS OF WAITING STRATEGIES AND REBALANCING STRATEGIES

Strategy	Reb. performed	Avg. pick-up delay [s]	Served requests	Vessels used	Distance empty [m]	Distance loaded [m]
Base		18.2	155	8	341	412
Waiting strategies						
Wait first		32.5	155	9	215	408
Intensity		19.2	155	7	340	410
Rebalancing strategies						
Location top 1	33	17.9	155	8	348	409
Location top 2	73	19.1	155	7	422	410
Location top 3	106	16.2	155	8	531	410
Delay	8	19.0	155	8	361	409
Forecast	105	15.8	155	9	414	410
Intensity 1	6	17.8	155	8	359	412
Intensity 2	165	11.6	155	8	509	412
Intensity 3	250	7.2	155	10	506	415

TABLE III

THE MOST RELEVANT RESULTS OF THE DIFFERENT REBALANCING STRATEGIES IN COMBINATION WITH THE WAIT FIRST STRATEGY.

Strategy	Reb. performed	Avg. pick-up delay [s]	Served requests	Vessels used	Distance empty [m]	Distance loaded [m]
Base		18.2	155	8	341	412
Rebalancing strategies & Wait First						
Location top 1	38	25.1	155	8	258	408
Location top 2	58	25.5	155	8	277	406
Location top 3	101	22.1	155	9	324	406
Delay	12	31.1	155	9	257	407
Forecast	78	23.5	155	10	223	403
Intensity 1	8	32.4	155	8	230	407
Intensity 2	156	16.6	155	9	308	412
Intensity 3	249	13.1	155	10	302	412

TABLE IV

THE MOST RELEVANT RESULTS OF THE DIFFERENT REBALANCING STRATEGIES IN COMBINATION WITH THE INTENSITY WAITING STRATEGY.

Strategy	Reb. performed	Avg. pick-up delay [s]	Served requests	Vessels used	Distance empty [m]	Distance loaded [m]
Base		18.2	155	8	341	412
Rebalancing strategies & Intensity						
Location top 1	27	19.1	155	7	354	408
Location top 2	78	15.2	155	8	402	406
Location top 3	99	16.3	155	8	454	406
Delay	9	21.1	155	8	339	407
Forecast	135	10.0	155	11	373	403
Intensity 1	6	19.2	155	7	346	407
Intensity 2	163	13.3	155	8	506	412
Intensity 3	256	7.6	155	10	493	412

number of vehicles could be reduced to 7, but it also needs 8 vehicles for the top 2 locations rebalancing strategy, where previously 7 were needed. Overall, with the combinations that increase the delay, the distance is reduced and vice versa.

IV. DISCUSSION

1) Sensitivity analysis: Sensitivity analysis examines how changing key variables or assumptions impacts outcomes. It systematically tests how adjustments in objective parameters, like delay or distance, influence the results. Tests on the base case's parameters show surprising results, such as an increase in the delay parameter reducing delay but unexpectedly affecting the number of vehicles used. Fixing the number of vehicles used to 8 helps create more comparable scenarios, leading to more consistent patterns in results. Different strategies respond differently to changes in objective parameters. While some align with expectations, others exhibit unpredictability, notably in the 'Delay-' scenario, indicating the unpredictable nature of the simulation's outcomes. The sensitivity analysis reveals unexpected outcomes due to the simulation's complex interplay of decisions and their cascading effects. Implementing a planning-based matching method might mitigate this unpredictability by enhancing decision-making and foreseeing future implications more accurately. Creating a longer prediction horizon can have additional benefits as well, as discussed in the next section.

2) Limitations of a short planning horizon: The 15-minute planning horizon, a result of the model's foundation in MoD (Matching on Demand), restricted the ability to consider long-term future planning. Transitioning from MoD to a mixed request system proved to be challenging, as it required a profound understanding of the model and involved significant modifications. Despite retaining aspects of MoD matching, the changes were substantial. The consulting with Flying Fish revealed their suggestion-based algorithm with complete future planning, distinct from the MoD approach, which was centered on current-time optimality. Converting the entire request handling and matching process would have essentially necessitated building a new simulator, which conflicted with the research's primary goal of exploring waiting and rebalancing strategies within the existing simulator. Therefore, the decision was made to preserve the MoD matching structure and create short-term future planning through RTV graphs, a computationally expensive process. The planning horizon was kept at 10 minutes to balance computational time and benefits. This short planning horizon limits matching options and hinders the consideration of long-term implications, which could have been addressed with planning-based matching or a longer planning horizon. Additionally, the effectiveness of waiting strategies is reduced due to the limited foresight. These limitations were acknowledged as compromises given the available resources and research constraints.

3) Test data selection: This decision to use only one day as test data, although limiting, was influenced by several factors, including the availability of data for just one week and the substantial differences in demand patterns between weekdays and weekends. Additionally, some strategies require historical requests, which must be representative of an average day but not identical to it, to avoid unrealistically high predictability. The use of a more extensive dataset, tuned to specific days or seasons, is recommended for practical implementations. Given the limited data available for this research, focusing on one average day for testing is considered acceptable, with the possibility of more refined datasets for specific days in the future.

4) Objective parameters: Assigning numerical values to the tuning parameters of the objective function is complex due to factors that are not easily quantifiable. An approach is to link parameters to real-life costs, such as distance traveled, but not all parameters can be as straightforwardly calculated. An alternative approach is to iteratively tune these parameters until reasonable results are achieved, although this can be time-consuming and requires careful consideration of how adjustments affect each other. The primary performance indicator chosen was the number of vehicles used. The goal was to create a reliable benchmark for comparing results, not to find the perfect objective function.

5) Simulation limitations: The goal of a simulation is to represent reality, although it never perfectly matches real-life complexity due to inherent challenges. This simulation makes certain simplifications compared to reality. For example, it assumes a constant speed for water taxis, even though they naturally have varying speeds depending on waterway size and docking times. These features were omitted because their impact on results would be minimal, and the task of modeling detailed speed profiles was time-consuming. Additionally, the simulation assumes that skippers are always on duty, neglecting breaks that occur in reality, as modeling breaks without a long-term schedule is complex and may not significantly enhance realism. In summary, these potential additions would demand significant effort and likely not justify the marginal improvements in realism, so they were excluded in favor of other priorities.

6) Prediction model assumptions: For the rebalancing strategy reliant on a forecast, a full-fledged prediction model was impractical due to time and data constraints. Instead, a simplified model was used to imitate predictions for a fraction of future requests, with a conservative estimate of 50%. While recent research has demonstrated high accuracy in prediction models, these rates require well-trained models. Considering existing reservations in the water taxi scenario, the 50% prediction rate was considered a reasonable compromise for testing a strategy with predictive capabilities.

V. Conclusion

In conclusion, this research explored strategies for enhancing the performance of dynamic transport systems, with a focus on reducing delays, traveled distance, and the number of vehicles used.

The analysis of system characteristics, the creation of a simulation model, and the evaluation of waiting and rebalancing strategies have collectively provided insights into answering this question. Waiting strategies demonstrated a significant impact on system performance, with strategies like Wait First effectively reducing traveled distance but at the cost of increased pick-up delay and an additional vehicle. On the other hand, the intensity waiting strategy improved overall system performance with minimal trade-offs. Rebalancing strategies, while increasing traveled distance, succeeded in reducing average pick-up delays, with the "rebalancing to the top 2 most popular locations" strategy offering a favorable trade-off by using one less vehicle. Furthermore, the combination of waiting and rebalancing strategies demonstrated their compatibility and the potential to fine-tune system performance without compromising the overall effectiveness.

In summary, this research provides a solution to the main research question: operational performance can be enhanced through the implementation of rebalancing and waiting strategies. The choice of strategy can be tailored to the operator's preferences, whether prioritizing reduced distance or minimizing delays, and a combination of these strategies offers a nuanced approach to optimizing system performance.

References

Alonso-Mora, Javier, Samitha Samaranayake, et al. (Jan. 2017). "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment". In: *Proceedings of the National Academy of Sciences* 114, p. 201611675. DOI: 10.1073/pnas.1611675114.

Alonso-Mora, Javier, Alex Wallar, and Daniela Rus (Sept. 2017). "Predictive routing for autonomous mobility-on-demand systems with ride-sharing". en. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC: IEEE, pp. 3583–3590. ISBN: 978-1-5386-2682-5. DOI: 10.1109/IROS.2017.8206203. URL: http://ieeexplore.ieee.org/document/8206203/ (visited on 01/30/2023).

Benyahia, I. and J.-Y. Potvin (1998). "Decision support for vehicle dispatching using genetic programming". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 28.3, pp. 306–314. DOI: 10.1109/3468.668962.

Brodsky, Isaac (June 2018). *H3: Uber's hexagonal hierarchical spatial index — uber blog*. URL: https://www.uber.com/en-NL/blog/h3/.

Cheemakurthy, Harsha (2017). *Urban waterborne public transport systems: An overview of existing operations in world cities*. eng. KTH Royal Institute of Technology. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-220525 (visited on 02/28/2023).

Ferrucci, Francesco, Stefan Bock, and Michel Gendreau (2013). "A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods". In: *European Journal of Operational Research* 225.1, pp. 130–141. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2012.09.016. URL: https://www.sciencedirect.com/science/article/pii/S0377221712006807.

Flying Fish (2023). URL: https://www.flying-fish.tech/case-studies/watertaxi-operations-system (visited on 03/17/2023).

Gendreau, Michel et al. (2006). "Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries". In: *Transportation Research Part C: Emerging Technologies* 14.3, pp. 157–174. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2006.03.002. URL: https://www.sciencedirect.com/science/article/pii/S0968090X06000349.

Hochreiter, Sepp and Jürgen Schmidhuber (Dec. 1997). "Long Short-term Memory". In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Iglesias, Ramon et al. (Sept. 2017). *Data-Driven Model Predictive Control of Autonomous Mobility-on-Demand Systems*. arXiv:1709.07032 [cs, stat]. URL: http://arxiv.org/abs/1709.07032 (visited on 02/08/2023).

Pavone, Marco et al. (May 2012). "Robotic load balancing for mobility-on-demand systems". In: *International Journal of Robotic Research - IJRR* 31, pp. 839–854. DOI: 10.1177/0278364912444766.

Tanko, M. and M. I. Burke (Jan. 2017). "Transport innovations and their effect on cities: the emergence of urban linear ferries worldwide". en. In: *Transportation Research Procedia*. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016 25, pp. 3957–3970. ISSN: 2352-1465. DOI: 10.1016/j.trpro.2017.05.483. URL: https://www.sciencedirect.com/science/article/pii/S2352146517307901 (visited on 02/28/2023).

Vonolfen, Stefan and Michael Affenzeller (Jan. 2016). "Distribution of waiting time for dynamic pickup and delivery problems". en. In: *Annals of Operations Research* 236.2, pp. 359–382. ISSN: 1572-9338. DOI: 10.1007/s10479-014-1683-6. URL: https://doi.org/10.1007/s10479-014-1683-6 (visited on 01/30/2023).

Bibliography

(Apr. 2014). URL: https://networkx.org/.

Agatz, Niels et al. (Dec. 2012). “Optimization for dynamic ride-sharing: A review”. en. In: *European Journal of Operational Research* 223.2, pp. 295–303. ISSN: 03772217. DOI: 10.1016/j.ejor.2012.05.028. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221712003864 (visited on 03/02/2023).

Alonso-Mora, Javier, Samitha Samaranayake, et al. (Jan. 2017). “On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment”. In: *Proceedings of the National Academy of Sciences* 114, p. 201611675. DOI: 10.1073/pnas.1611675114.

Alonso-Mora, Javier, Alex Wallar, and Daniela Rus (Sept. 2017). “Predictive routing for autonomous mobility-on-demand systems with ride-sharing”. en. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC: IEEE, pp. 3583–3590. ISBN: 978-1-5386-2682-5. DOI: 10.1109/IROS.2017.8206203. URL: http://ieeexplore.ieee.org/document/8206203/ (visited on 01/30/2023).

Benyahia, I. and J.-Y. Potvin (1998). “Decision support for vehicle dispatching using genetic programming”. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 28.3, pp. 306–314. DOI: 10.1109/3468.668962.

Beyer, Hans-Georg and Hans-Paul Schwefel (Mar. 2002). “Evolution strategies – A comprehensive introduction”. In: *Natural Computing* 1.1, pp. 3–52. ISSN: 1572-9796. DOI: 10.1023/A:1015059928466. URL: https://doi.org/10.1023/A:1015059928466.

Brodsky, Isaac (June 2018). *H3: Uber’s hexagonal hierarchical spatial index — uber blog*. URL: https://www.uber.com/en-NL/blog/h3/.

Chang, Yu Sang, Yong Joo Lee, and Sung Sup Brian Choi (Oct. 2017). “Is there more traffic congestion in larger cities? -Scaling analysis of the 101 largest U.S. urban centers-”. en. In: *Transport Policy* 59, pp. 54–63. ISSN: 0967-070X. DOI: 10.1016/j.tranpol.2017.07.002. URL: https://www.sciencedirect.com/science/article/pii/S0967070X17304626 (visited on 02/28/2023).

Cheemakurthy, Harsha (2017). *Urban waterborne public transport systems: An overview of existing operations in world cities*. eng. KTH Royal Institute of Technology. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-220525 (visited on 02/28/2023).

Duan, Leyi et al. (Feb. 2020). “Centralized and decentralized autonomous dispatching strategy for dynamic autonomous taxi operation in hybrid request mode”. en. In: *Transportation Research Part C: Emerging Technologies* 111, pp. 397–420. ISSN: 0968-090X. DOI: 10.1016/j.trc.2019.12.020. URL: https://www.sciencedirect.com/science/article/pii/S0968090X19306710 (visited on 03/01/2023).

Ferrucci, Francesco, Stefan Bock, and Michel Gendreau (2013). “A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods”. In: *European Journal of Operational Research* 225.1, pp. 130–141. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2012.09.016. URL: https://www.sciencedirect.com/science/article/pii/S0377221712006807.

Flying Fish (2023). URL: https://www.flying-fish.tech/case-studies/watertaxi-operations-system (visited on 03/17/2023).

Gendreau, Michel et al. (2006). “Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries”. In: *Transportation Research Part C: Emerging Technologies* 14.3, pp. 157–174. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2006.03.002. URL: https://www.sciencedirect.com/science/article/pii/S0968090X06000349.

Gu, Yewen and Stein W. Wallace (Oct. 2021). “Operational benefits of autonomous vessels in logistics—A case of autonomous water-taxis in Bergen”. en. In: *Transportation Research Part E: Logistics and Transportation Review* 154, p. 102456. ISSN: 1366-5545. DOI: 10.1016/j.tre.2021.102456. URL: https://www.sciencedirect.com/science/article/pii/S1366554521002209 (visited on 01/30/2023).

Hochreiter, Sepp and Jürgen Schmidhuber (Dec. 1997). “Long Short-term Memory”. In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Iglesias, Ramon et al. (Sept. 2017). *Data-Driven Model Predictive Control of Autonomous Mobility-on-Demand Systems*. arXiv:1709.07032 [cs, stat]. URL: http://arxiv.org/abs/1709.07032 (visited on 02/08/2023).

Inman, J (1849). “The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes”. In: *London: R & J Rivington*.

Kooti, Farshad et al. (2017). “Analyzing Uber’s Ride-Sharing Economy”. In: *Proceedings of the 26th International Conference on World Wide Web Companion*. WWW ’17 Companion. Perth, Australia: International World Wide Web Conferences Steering Committee, pp. 574–582. ISBN: 9781450349147. DOI: 10.1145/3041021.3054194. URL: https://doi.org/10.1145/3041021.3054194.

Kretschmann, Lutz, Hans-Christoph Burmeister, and Carlos Jahn (2017). “Analyzing the economic benefit of unmanned autonomous ships: An exploratory cost-comparison between an autonomous and a conventional bulk carrier”. In: *Research in Transportation Business & Management* 25, pp. 76–86. ISSN: 2210-5395. DOI: https://doi.org/10.1016/j.rtbm.2017.06.002. URL: https://www.sciencedirect.com/science/article/pii/S2210539516301328.

Laar, Paul (Mar. 2022). *Rotterdam: A historical perspective for the future*. URL: https://portusonline.org/rotterdam-a-historical-perspective-for-the-future/.

Liashchynskyi, Petro and Pavlo Liashchynskyi (2019). “Grid search, random search, genetic algorithm: a big comparison for NAS”. In: *arXiv preprint arXiv:1912.06059*.

Lo, Jenny and Steve Morseman (2018). “The Perfect uberPOOL: A Case Study on Trade-Offs”. en. In: *Ethnographic Praxis in Industry Conference Proceedings* 2018.1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1559-8918.2018.01204, pp. 195–223. ISSN: 1559-8918. DOI: 10.1111/1559-8918.2018.01204. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/1559-8918.2018.01204 (visited on 02/03/2023).

Milakis, Dimitris, Bart van Arem, and Bert van Wee (Jan. 2017). “Policy and society related implications of automated driving: A review of literature and directions for future research”. en. In: *Journal of Intelligent Transportation Systems* 21.4, pp. 324–348. ISSN: 1547-2450. DOI: 10.1080/15472450.2017.1291351. URL: https://www.sciencedirect.com/org/science/article/pii/S1547245022002341 (visited on 03/01/2023).

Mirjalili, Seyedali (2019). “Genetic algorithm”. In: *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pp. 43–55.

Mitrović-Minić, Snežana and Gilbert Laporte (Aug. 2004). “Waiting strategies for the dynamic pickup and delivery problem with time windows”. en. In: *Transportation Research Part B: Methodological* 38.7, pp. 635–655. ISSN: 0191-2615. DOI: 10.1016/j.trb.2003.09.002. URL: https://www.sciencedirect.com/science/article/pii/S0191261503001073 (visited on 02/15/2023).

Pavone, Marco et al. (May 2012). “Robotic load balancing for mobility-on-demand systems”. In: *International Journal of Robotic Research - IJRR* 31, pp. 839–854. DOI: 10.1177/0278364912444766.

Saranow, Jennifer (2006). “Carpooling for Grown-Ups — High Gas Prices, New Services Give Ride-Sharing a Boost; Rating Your Fellow Rider”. en. In.

Soltani, Ali et al. (2015). “Travel Patterns of Urban Linear Ferry Passengers: Analysis of Smart card Fare Data for Brisbane, Queensland, Australia”. In: *Transportation Research Record* 2535.1, pp. 79–87. DOI: 10.3141/2535-09. eprint: https://doi.org/10.3141/2535-09. URL: https://doi.org/10.3141/2535-09.

Song, Chaoming et al. (2010). “Limits of Predictability in Human Mobility”. In: *Science* 327.5968, pp. 1018–1021. DOI: 10.1126/science.1177170. eprint: https://www.science.org/doi/pdf/10.1126/science.1177170. URL: https://www.science.org/doi/abs/10.1126/science.1177170.

Song, Xuan, Hiroshi Kanasugi, and Ryosuke Shibasaki (2016). “Deeptransport: Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level”. In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI’16. New York, New York, USA: AAAI Press, pp. 2618–2624. ISBN: 9781577357704.

Stiglic, Mitja et al. (2016). “Making dynamic ride-sharing work: The impact of driver and rider flexibility”. In: *Transportation Research Part E: Logistics and Transportation Review* 91, pp. 190–207. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2016.04.010. URL: https://www.sciencedirect.com/science/article/pii/S1366554515303033.

Tanko, M. and M. I. Burke (Jan. 2017). “Transport innovations and their effect on cities: the emergence of urban linear ferries worldwide”. en. In: *Transportation Research Procedia*. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016 25, pp. 3957–3970. ISSN: 2352-1465. DOI: 10.1016/j.trpro.2017.05.483. URL: https://www.sciencedirect.com/science/article/pii/S2352146517307901 (visited on 02/28/2023).

Thompson, Robert, Richard Burroughs, and Tiffany Smythe (2006). “Exploring the connections between ferries and urban form: Some considerations before jumping on board”. In: *Journal of Urban Technology* 13.2. Publisher: Routledge _eprint: https://doi.org/10.1080/10630730600872021, pp. 25–52. DOI: 10.1080/10630730600872021. URL: https://doi.org/10.1080/10630730600872021.

van Gisbergen, D (2023). “Waiting and rebalancing strategies in an urban waterborne transport environment”. In.

Vonolfen, Stefan and Michael Affenzeller (Jan. 2016). “Distribution of waiting time for dynamic pickup and delivery problems”. en. In: *Annals of Operations Research* 236.2, pp. 359–382. ISSN: 1572-9338. DOI: 10.1007/s10479-014-1683-6. URL: https://doi.org/10.1007/s10479-014-1683-6 (visited on 01/30/2023).

Watertaxi-Rotterdam (2023). *Over ons - Watertaxi Rotterdam*. URL: https://www.watertaxirotterdam.nl/over-ons (visited on 03/17/2023).

Zhigljavsky, Anatoly A (2012). *Theory of global random search*. Vol. 65. Springer Science & Business Media.