Maria Lundin Brenna, Amalie Omholt Ellestad &
Anna Sofie Lunde

An Adaptive Large Neighbourhood Search Heuristic for a Dial-a-Ride Problem with Real-Time Disruptions

Master's thesis in Industrial Economics and Technology Management
Supervisor: Professor Henrik Andersson
June 2022

Maria Lundin Brenna, Amalie Omholt Ellestad & Anna Sofie Lunde

An Adaptive Large Neighbourhood Search Heuristic for a Dial-a-Ride Problem with Real-Time Disruptions

Master's thesis in Industrial Economics and Technology Management
Supervisor: Professor Henrik Andersson
June 2022

Norwegian University of Science and Technology
Faculty of Economics and Management
Dept. of Industrial Economics and Technology Management

Preface

This thesis concludes our Master of Science at the Department of Industrial Economics and Technology Management at the Norwegian University of Science and Technology. The thesis is written during the spring semester of 2022. It is a continuation of our specialisation project within the course *TIØ4500 - Managerial Economics and Operations Research, Specialization Project* during the fall semester of 2021.

We want to thank our supervisor Professor Henrik Andersson for his valuable guidance throughout the project. In addition, we would like to express our sincere gratitude towards the administration behind the Ruter Aldersvennlig Transport service, who have provided us with helpful insights into the service and domain knowledge. A special thanks goes to Heidi Aakre Juvik for inspiring discussions on the utilisation of optimisation research in the Ruter Aldersvennlig Transport service and access to valuable data.

Trondheim, June 2022
Maria Lundin Brenna, Amalie Omholt Ellestad, Anna Sofie Lunde

Abstract

This thesis explores the dial-a-ride problem (DARP) faced by Ruter Aldersvennlig Transport (RAT), a door-to-door transportation service for the elderly in Oslo. The DARP concerns the design of vehicle routes and schedules to meet several requests for passenger transportation between specified origins and destinations. A subset of the requests is known beforehand, while others are revealed throughout the day, thus making the problem dynamic. Further, the service may be subject to other disruptions such as delays, cancellations, and no-shows. Therefore, utilising operational research to create and maintain efficient routes in case of unforeseen events is of great interest for RAT.

A literature study is conducted to obtain a broader understanding of existing literature relevant to the thesis. The study mainly focuses on literature regarding the dynamic DARP and disruption management in general. It is revealed that existing literature on the disruptive DARP is somewhat limited, focusing mainly on the arrival of new user requests. Consequently, our contribution is the implementation of disruption management for a wide range of disruption types.

The proposed solution method consists of four main parts: a greedy construction heuristic, an adaptive large neighbourhood search (ALNS) heuristic, a greedy insertion heuristic, and a disruption updater. Additionally, a realistic simulation framework is developed to generate disruptions. Firstly, the greedy construction heuristic creates an initial route plan for the requests that have arrived before the beginning of the operational day. The ALNS is then used to improve the route plan. Next, the simulator starts to generate disruptions. In the case of a new request, the greedy insertion heuristic tries to quickly insert the request into the current route plan. The other disruption types are first handled by the disruption updater, which adapts the current route plan to include the disruption. The ALNS is then used to improve the route plan. The solution method and simulator are run iteratively throughout the operational day.

To best mirror the real-life problem, historical request data provided by RAT has been used to generate test instances and develop the proposed simulation framework. Through a computational study, these are used to provide valuable insights into the problem. Results show that the proposed solution method successfully solves all instances within a reasonable time. Additionally, it handles disruptions in a timely manner, with a fast response time to the customer. The heuristic is also shown to serve more requests and better reduce the impact of delays compared with simpler solvers. The work may thus serve as a basis for further research on the disruptive DARP.

Furthermore, managerial insights are obtained by analysing the value of implementing different policies. Specifically, these policies involve adjusting the pick-up time of rejected requests, implementing an order deadline on new requests, changing the vehicle fleet size, and implementing a morning and afternoon shift. Several of the policies are shown to reduce the rejection rate and provide more cost-effective routes, thus providing RAT with valuable insights relevant to their decision-making.

Sammendrag

Denne oppgaven utforsker dial-a-ride problemet (DARP) til Ruter Aldersvennlig Transport (RAT), en dør til dør transporttjeneste for eldre i Oslo. DARP dreier seg om å designe ruteplaner for transport av passasjerer mellom ulike start- og endelokasjoner. En andel av transportforespørslene er kjent på forhånd, mens andre kommer i løpet av dagen. Problemet er derfor dynamisk. Det kan også oppstå andre uforutsette hendelser, som forsinkelser, kanselleringer og kunder som ikke møter opp. Å bruke optimering til å konstruere og opprettholde effektive ruteplaner ved slike hendelser, er derfor av høy interesse for RAT.

For å få en bredere forståelse av problemet, presenteres en oversikt over tidligere publisert litteratur relatert til emnet. Litteraturen som blir presentert omhandler først og fremst det dynamiske DARP eller håndtering av uforutsette hendelser generelt. Det fremkommer at eksisterende litteratur på det dynamiske DARP først og fremst fokuserer på håndtering av nye transportforespørsler. Andre uforutsette hendelser blir sjeldent diskutert. Vårt bidrag til litteraturen er derfor å modellere og løse et DARP som inkluderer mange ulike typer uforutsette hendelser.

Den foreslåtte løsningsmetoden består av fire hoveddeler: en grådig konstruksjonsheuristikk, en adaptive large neighbourhood search (ALNS) heuristikk, en grådig innsettingsheuristikk og en hendelsesoppdaterer. I tillegg har vi konstruert et realistisk simuleringsrammeverk som genererer de uforutsette hendelsene. Først brukes den grådige konstruksjonsheuristikken til å lage en initiell ruteplan for de forespørslene som har kommet inn før klokken 10:00 samme dag. Så brukes ALNSen til å forbedre ruteplanen. Simulatoren begynnere så å generere hendelser. Ved nye forespørsler prøver den grådige innsettingsheuristikken å raskt legge forespørselen inn i ruteplanen. Andre hendelser håndteres først av hendelsesoppdatereren, som oppdaterer ruteplanen til å inkludere hendelsen. ALNSen brukes så til å forbedre ruteplanen. Løsningsmetoden og simulatoren kjøres iterativt gjennom hele den operasjonelle dagen.

For å best etterligne det virkelige problemet brukes historisk data fra RAT til å generere testinstanser og utvikle simuleringsrammeverket. Gjennom en beregningsstudie blir disse brukt til å skaffe verdifull innsikt om problemet. Resultatene viser at den foreslåtte løsningsmetoden klarer å løse alle testinstanser innen rimelig tid. I tillegg håndterer den effektivt uforutsette hendelser og har en kort responstid til kunden. Sammenlignet med enklere løsningsmetoder klarer heuristikken å håndtere flere transportforespørsler samt å bedre redusere konsekvensen av forsinkelser. Arbeidet kan dermed tjene som grunnlag for videre forskning rundt håndtering av uforutsette hendelser i DARP.

Videre gir oppgaven beslutningstøtte til RAT gjennom å analysere verdien av å implementere ulike strategier. Strategiene omfatter endring av avviste kunders hentetidspunkter, implementering av en bestillingsfrist, endringer i antall busser og innføringen av formiddags- og ettermiddagsskift. Flere av strategiene resulterer i redusert antall avvisninger og mer kostnadseffektive ruter. De bidrar derfor med verdifull innsikt relevant for RAT tjenesten.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

By 2050 it is projected that one in six people in the world will be over the age of 65, up from one in 11 in 2019 (United Nations and Social Affairs, 2019). The shift in the age of the population is referred to as the age wave and is a result of increased life expectancy, ageing of the baby boom generation, and declining birth rates in recent decades. To ensure the well-being of the elderly, society must facilitate for an active and social lifestyle, for instance, through sufficient transportation options. While some elderly drive their own car or can use ordinary public transport, many have a health state which makes this problematic. As a result, dial-a-ride (DAR) services have emerged as a common alternative to ordinary public transport.

The dial-a-ride problem (DARP) concerns the design of route schedules to meet several requests for passenger transportation between specified origins and destinations. One of the most common examples of the DARP arises in door-to-door transportation of the elderly and disabled. In light of the upcoming age wave and the current focus on sustainable transportation, dial-a-ride (DAR) services are becoming increasingly popular. As the demand for such services increases, service providers are no longer able to produce time schedules and vehicle routes manually (Molenbruch et al., 2017). Consequently, operational research on DAR services is necessary to develop effective planning systems.

The background for this thesis is a DAR service operated by the public transportation provider Ruter, named Ruter Aldersvennlig Transport (RAT). RAT offers door-to-door transportation for people over the age of 67 in Oslo and is a part of Ruter's commitment to create sustainable mobility and freedom of movement for all age groups. The purpose of the service is to provide a flexible and convenient transportation mode for the elderly, thereby facilitating an active and social lifestyle. Consequently, their main priority is to serve as many requests as possible, while maintaining a high degree of customer satisfaction. To achieve this, effective routing and scheduling of requests are needed.

Rides with the RAT service can be ordered both up to a week in advance or on-demand. The latter results in the need for frequent replanning of existing route plans. Additionally, RAT may be subject to other types of unforeseen events such as cancellations, customer no-shows, traffic congestion, vehicle breakdowns, and more. Even though potential delays are one of the key reasons customers do not use the RAT service (Epinion, 2021), RAT's replanning in the case of such events is limited. Therefore, utilising operational research to create and maintain efficient routes in case of unforeseen events is of great interest for RAT.

The purpose of this thesis is to understand and solve the DARP faced by RAT, thereby providing them with managerial insights relevant to their decision-making. Through the carried out work, two major contributions are made. The first is the development of a solution method for realistic instances of the problem. This includes the design of an initial route plan at the start of the day, and the re-optimisation of routes in response to disruptions. The proposed solution method combines a greedy insertion heuristic with an adaptive large neighbourhood search (ALNS) to effectively respond to disruptions in the form of new requests, delays, cancellations, and customer no-shows. To the best of the authors' knowledge, existing literature on the disruptive DARP does not consider the wide range of disruptions studied in this thesis. The second contribution is the development of a realistic simulation framework to assess the solution method's performance on real-life instances.

This thesis is written in collaboration with Ruter. Through the collaboration, we have been provided with historical request data for the RAT service. This has been used to generate the test instances and develop the proposed simulation framework.

The thesis is organised as follows: Chapter 2 establishes the background and motivation of the thesis, including a description of the concept of DAR services and how RAT operates. Next, Chapter 3 presents a literature study on the DARP intending to provide a broader understanding of existing literature relevant to the thesis. A problem description of the specific DARP faced by RAT is provided in Chapter 4, followed by a mathematical formulation in Chapter 5. Chapter 6 presents the proposed solution method, and Chapter 7 presents the simulation framework used to generate disruptions. Further, Chapter 8 describes the implementation and generation of the test instances, which are used in the computational study in Chapter 9. Finally, Chapter 10 provides concluding remarks for the report before possible areas of future research are presented in Chapter 11.

Chapter 2

Background

This chapter establishes the background and motivation of this thesis. Section 2.1 discusses the upcoming age wave and how it leads to an increased interest in on-demand transportation services. Next, challenges related to the transport of the elderly are presented in Section 2.2, while Section 2.3 describes the concept of DAR services. Finally, Section 2.4 describes how RAT operates and addresses some of its challenges. All sections in this chapter are based on sections from the RAT report by Brenna et al. (2021).

2.1 Age Wave

The age wave refers to the shift in the age of the population. By 2030, the number of older people in Norway is projected to surpass the number of people under 15 for the first time. Figure 2.1 shows the historical and projected number of people across different age groups in the period 1980 - 2100. In 2100, about one-third of the population will be over 67 years, compared with today's number of one-sixth (Statistisk sentralbyrå, 2020, 2022). The age wave is a result of increased life expectancy, ageing of the baby boom generation, and declining birth rates in the recent decades (Aguiar and Macário, 2017).

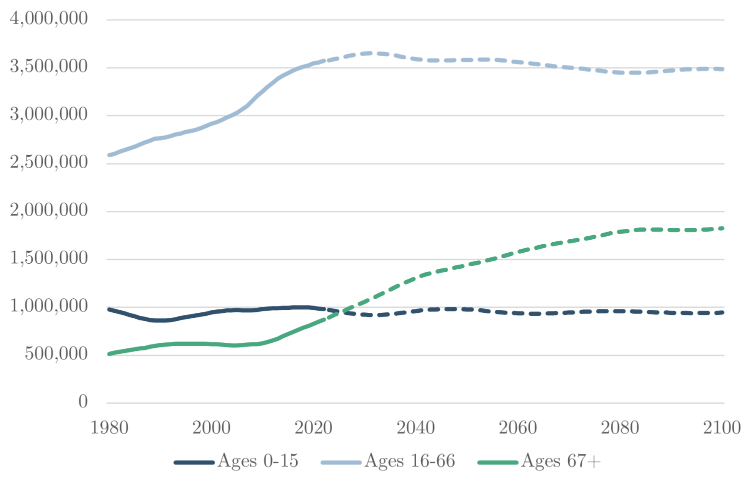


Figure 2.1: Historical and projected number of people in age groups 0-15 (dark blue), 16-66 (light blue), and 67+ (green) in the period 1980 - 2100. Projected numbers in dotted lines. (Statistisk sentralbyrå, 2020, 2022).

The population is ageing, but as the elderly are healthier and can stay in their homes longer, more people are "ageing in place". While "ageing in place" is often desirable for the elderly and economically beneficial for the welfare state of Norway, there are some related challenges. "Ageing in place" can be associated with social isolation and loneliness. Maintaining an active and social lifestyle by regularly undertaking out-of-home activities is essential to reduce social isolation. An active and social lifestyle can provide health benefits such as increased quality of life and maintained physical mobility (Nordbakke et al., 2020). However, older people are not a homogeneous group, and not everyone has the same abilities to undertake out-of-home activities due to reduced health and physical limitations. For some, staying active and mobile is a challenge, and lack thereof can cause decreased quality of life. Hjorthol et al. (2011) state that without sufficient transportation options, one will feel old and isolated, and this itself can cause further reduction in quality of life.

2.2 Transport of Elderly

Transport of the elderly differs from other age groups due to differences in travel patterns, requirements, and health states. A study from New Zealand (O'Fallon and Sullivan, 2003) shows that the elderly (aged 65 and over) tend to undertake fewer and shorter trips per day compared with younger adults (ages 25-59). Additionally, the number of travels to educational institutions and workplaces is reduced after retirement. Consequently, the elderly tend to travel during different hours than the other age groups. While younger age groups tend to travel before 9:30 and after 15:00, the elderly mostly travel between 9:30 and 15:00 (O'Fallon and Sullivan, 2003).

The travel purposes of the elderly can be divided into basic activities and lifestyle activities (Hjorthol et al., 2011). The former include activities such as grocery shopping and visits to health services, while the latter may include visits to friends and leisure activities. These differ from basic activities as they are not crucial, but they still have social and recreational purposes and are therefore essential to maintain a high quality of life.

For some elderly, there are several challenges with using ordinary public transportation. The most significant challenges are related to reduced health and physical limitations, such as difficulties walking to and from transportation stops. Consequently, it is essential that the walking distance to and from the stop is not too long and feels safe. Other concerns include the availability of seats and room for potential rolling walkers and wheelchairs. Boarding and disembarking are other worrying areas as there may exist a gap between the vehicle and the ground (Hjorthol et al., 2011).

Due to the mentioned challenges and the independence and flexibility a private car offers, it is often the preferred mean of transportation. However, for some, driving is not an alternative due to physical limitations, feelings of incompetence, or uncertainty when driving. In such cases, family assistance can be an alternative mean of transportation, but there are also challenges related to this. The elderly are often reluctant to ask friends and family due to the concern of being a burden. Hence they instead turn to more expensive alternatives such as taxi (Gilhooly et al., 2002).

From an environmental and sustainable point of view, all citizens, including the elderly, should be encouraged to use public transportation instead of taxis or private cars. A public transportation system that is safe, easily accessible, and meets the needs of all groups is necessary to achieve this.

2.3 Dial-a-Ride Services

DAR services have become a standard public transportation alternative for those who cannot use ordinary public transportation. A DAR service is a demand-responsive door-to-door transportation service. DAR services remove several obstacles by being door-to-door, such as long and unsafe walking distances to the transportation stops. The vehicles used are often modified towards the needs of the passengers, e.g., low-floored buses with extra room for rolling walkers and wheelchairs.

Most DAR services operate with ride-sharing, namely that several passengers are transported by the same vehicle simultaneously even though their origins and destinations may differ. Without ride-sharing, a DAR service becomes more similar to a regular taxi service. As mentioned, this is not beneficial from an economic or environmental point of view. A decreasing degree of ride-sharing leads to increased costs per trip, as illustrated in Figure 2.2.

Another consequence is an increased number of cars on the roads, creating more congestion and emissions. As public transportation has a high degree of ride-sharing, it is one of the most economical and environmental means of transportation. However, ride-shared DAR services are good alternatives in those cases where traditional public transportation is not accessible.

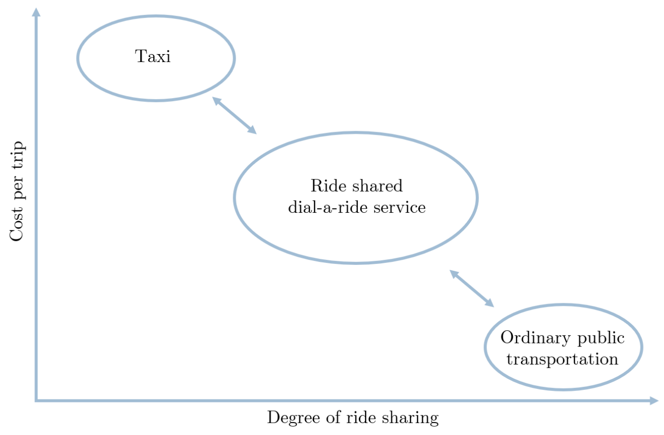


Figure 2.2: Illustration of the relationship between cost per trip and degree of ride-sharing.

In Norway, several DAR services exist, and Table 2.1 presents some of these. Ruter Aldersvennlig Transport (RAT) and AtB 67plus are services specifically for the elderly, while the other services are available for all age groups. Smart Transport i Distriktene differs from the other services as it considers not only the transportation of people, but also goods. As this thesis is written in collaboration with RAT, the remainder of the thesis focuses on this service.

Table 2.1: Overview of DAR services in Norway.

Name of Service	Operational Area
Ruter Aldersvennlig Transport	Oslo
AtB 67pluss	Trondheim
Smart Transport i Distriktene	Folldal
VKT Bestilling	Vestfold og Telemark
Flexx	Østfold

2.4 Ruter Aldersvennlig Transport

Ruter Aldersvennlig Transport (RAT), also referred to as "Rosa Busser", is a door-to-door transportation service offered to elderly people over 67 years old in Oslo, Norway. The service is operated by Ruter, which is responsible for all public transportation in Oslo and Akershus.

2.4.1 About the Service

The RAT pilot project started in 2017, but it has gone through several adaptations. In the beginning, the service was a fixed-route service, but after receiving feedback from customers, the service changed to an on-demand service. As of today, RAT operates within the six districts shown in Figure 2.3, namely Nordre Aker, Sagene, Vestre Aker, Ullern, Alna, and Bjerke. Additionally, the RAT service drives to Maridalen, Solemskogen, and Sørkedalen.

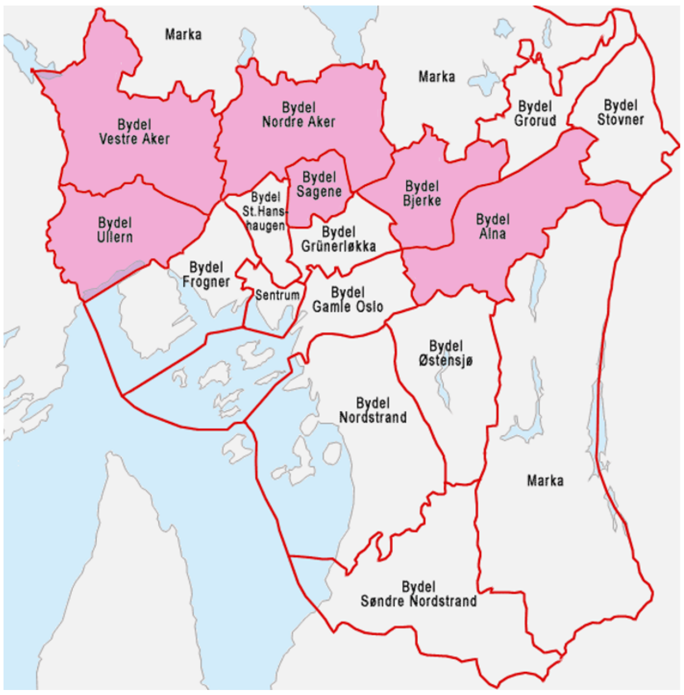


Figure 2.3: Map over the districts in Oslo. The areas RAT operates in are highlighted in pink. Adapted from source: Eidstuen (2011)

The RAT service operates from Monday to Saturday between 10:00 and 18:00. Rides can be ordered both on-demand and up to a week in advance by calling or through the app RuterBestilling. Visits to family and friends and health care appointments are often carefully planned, and transport is therefore ordered minimum a day in advance. Spontaneous trips, such as a ride home from the grocery store due to weather changes or spontaneous social activities, are often on-demand requests (Epinion, 2021).

When ordering, the customer gives either a preferred pick-up or drop-off time and their desired pick-up and drop-off locations. The customer then receives feedback on whether RAT can accept the request. If accepted, the customer also receives an initial pick-up time. If rejected, RAT may propose an alternative pick-up time in dialogue with the customer. RAT sends out text messages to confirm the pick-up time and inform about potential changes, such as delays. Then, the vehicle arrives, picks up the customer, and transports it to its drop-off location. Figure 2.4 shows the process of using the RAT service.

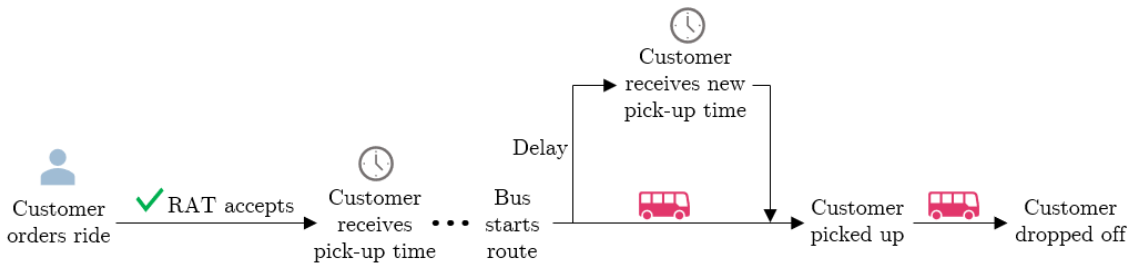


Figure 2.4: The user-interaction process of using a DAR service.

Ruter operates 16 minibuses for the RAT service. Figure 2.5 shows one of the RAT buses. The buses are low-floored with 15 seats, one wheelchair seat, and room for rolling walkers and other aids. In addition to adapted vehicles, the bus drivers assist with boarding, finding a seat, and disembarking to make the experience more seamless for the elderly. The bus driver also helps carry heavy shopping bags from the bus to the door if needed.

Figure 2.5: A minibus used by the RAT service. Source: Utviklingssenter for sykehjem og hjemmetjenester (2020)

2.4.2 Handling of Unforeseen Events

As with all transportation services, RAT may be subject to unforeseen events. Such events include new ride requests, cancellations, customer no-shows, traffic congestion, vehicle breakdowns, and more. Depending on the type of disruption, RAT can choose to modify the existing route plan thereafter.

RAT does not have a deadline for when customers can cancel their trips. They distinguish between cancels, which are cancellations made up to an hour before the planned pick-up time, and late cancels, which are cancellations made the last hour before the pick-up time. No-shows occur when a user does not show up at the pick-up location. In the case of both cancellations and no-shows, RAT removes the corresponding locations from the route plan. Due to the limited time for replanning, no-shows and late cancels are not likely to affect the route plan any further. Cancels, however, might free up time in the route plan that can be used to accommodate new requests.

As mentioned, RAT is an on-demand service. Therefore, they often experience disruptions in the form of new requests. In general, RAT aims to serve as many requests as possible. Consequently, if a bus is available to serve a new request, the request is always accepted. As of today, the demand is higher than what RAT is able to serve, hence some requests are rejected. To have a higher degree of flexibility in their replanning, RAT operates with a pick-up window of 15 minutes. This means that a customer is picked up between five minutes before the agreed pick-up time and 10 minutes after. Hence, the pick-up times of existing requests in the route plan may be slightly shifted to accommodate the new request.

Events such as traffic congestion, employees who are late for work, increased service times, and vehicle problems might all result in delays. As of now, RAT does not re-optimise the route to account for delays. If the bus is delayed beyond the pick-up window, the customer is informed about the delay and is given a new pick-up time. In case of extensive delays, RAT might have to cancel trips.

2.4.3 Goals and Challenges

The RAT service is a part of Ruter's commitment to create sustainable mobility and freedom of movement for all age groups. The service is currently subsidised by Oslo Kommune, as a part of their project to create a more age-friendly city. Consequently, the service's primary goal is to offer a flexible and customised transportation mode for the elderly to help them be active, independent, and mobile. As earlier mentioned, the elderly often have difficulties using ordinary public transportation. Therefore, a DAR service such as RAT is beneficial to achieve both Ruter's and Oslo Kommune's commitments.

The RAT service is currently not profitable and dependent on subsidies. Currently, the price equals the price of an "honnørbillett" on Ruter's regular public transportation modes. To become more profitable, the cost per trip should be decreased. RAT defines the cost per trip as the hourly hiring price per bus divided by the total number of passengers transported per bus per hour. A lower cost per trip might be achieved through optimised routes with a higher degree of ride-sharing and effective boarding and disembarking. However, there is a trade-off between effectiveness and quality of service. The elderly value that the trip feels safe and not rushed and that bus drivers assist them and understand their needs (Nordbakke et al., 2020). Consequently, reduced quality of service can lead to fewer using the service. As there is a connection between the number of trips and the cost per trip, a substantial reduction in quality of service would be disadvantageous for the profitability of the service.

To increase the number of requested trips, RAT must continue to address its customers' needs and preferences. Overall, the customers are satisfied with the RAT service. They say that the service gives them a more active life, increased independence, and increased self-esteem. However, they also call attention to some challenges related to the service. According to a customer survey by Epinion (2021), extended operational hours and areas are desirable. Further, the survey states that potential delays are one of the key reasons customers do not use the RAT service. Although the cause of a delay might be beyond RAT's control, it results in customer complaints and dissatisfaction. Consequently, reducing the impact of delays might be beneficial to retain current customers and acquire new ones.

Chapter 3

Literature Study

For several decades, the dial-a-ride problem (DARP) has been an active research field. This chapter aims to give a broader understanding of existing literature relevant to this thesis. Section 3.1 briefly explains how the literature search has been conducted. A description of the DARP, its applications, and its taxonomy are provided in Section 3.2. The objective function of the DARP varies greatly based on its application. Thus this section also elaborates on different objectives for the DARP. Additionally, common restrictions and solution methods used for the DARP are presented. As the DARP presented in this thesis is subject to several disruptions, Section 3.3 discusses the characteristics of disruption management and its applications to the DARP. Finally, 3.4 presents an overview of the reviewed literature and Section 3.5 gives a motivation for the thesis. Section 3.2 is based on Section 3.2 in the report on RAT by Brenna et al. (2021).

3.1 Literature Search Strategy

Our literature search strategy is influenced by the surveys of Ho et al. (2018) and Nasri et al. (2021). These sources provide an overview of the key features of the DARP and have served as a starting point in the search for relevant literature. Search for DARP-related terms, such as DARP + Dynamic, have been conducted on Google Scholar to find subsequent literature. The key terms used in conjunction with DARP are presented in Table 3.1.

Table 3.1: Key terms used in conjunction with the term DARP for literature review.

General Terms	**Objective**	**Constraints**	**Solution Method**
Dynamic	Multi-objective	Time window	Exact method
Disruption	Weights	Ride time	Metaheuristic
Recovery management	Quality of service	Capacitated	Online algorithm
Real-time	User inconvenience	User heterogeneity	Neighbourhood search
Re-optimisation	Deviation cost		Population based

Google Scholar's ranking was used to confine our search. It ranks documents based on where it was published, their author(s), and how often and recently it has been cited in other literature (Google, 2021). Articles written before 2010 were not considered, except for highly relevant articles cited in the surveys. In addition, the search was mainly limited to articles published in journals. After an initial selection, a more significant number of articles were browsed. Specifically, the articles' headlines, abstracts, and keywords were examined. The most relevant articles were selected for further review. These articles either resemble our problem or provide a new dimension to certain aspects of the DARP.

3.2 The Dial-a-Ride Problem

The DARP is a version of the pickup-and-delivery problem (PDP) and thus generalises the vehicle routing problem (VRP). The PDP concerns the design of routes and schedules to meet several requests for freight transport between specified origins and destinations (Savelsbergh and Sol, 1995). The VRP can be considered a PDP where all origins or destinations are located at the depot. If the requests in a PDP concern the transportation of people instead of goods, the problem is referred to as a DARP. Finding a feasible solution to both VRP and PDP is NP-hard as they are generalisations of the travelling salesman problem (TSP) (Cordeau, 2006). The human perspective of the DARP complicates the problem further by introducing an aspect of service quality. While a packet likely can stay on a vehicle for extended detours without being affected, a person will experience this as a decrease in the quality of service. Consequently, DARPs usually impose restrictions on ride durations as well as more narrow time windows for pick-up and delivery (Cordeau and Laporte, 2007).

3.2.1 Applications

DARPs are usually motivated by real-life applications. The features of each application differ and may result in specific constraints and objectives.

One major application area of DARPs is in health care, among others addressed by Beaudry et al. (2010), Detti et al. (2017), and Paquay et al. (2020). Time urgency is one of the major concerns for this application area, and imposing a prioritisation on the requests (urgent vs. normal) is often necessary (Beaudry et al., 2010). In addition, features such as staff/equipment compatibility and vehicle/patient compatibility are frequently considered. Furthermore, scheduling of staff and maintenance may add considerable complexity. Some papers also take patients' preferences into account. For instance, Detti et al. (2017) let patients choose their transportation provider from a given set of different non-profitable organisations.

Another application of DARPs is the one addressed in this paper, namely DAR services for the elderly and disabled. These are often non-profit services initiated to increase the life quality of their users. DARPs aimed at this application area may have multiple stakeholders with different and sometimes conflicting goals. Consequently, multi-criteria models, such as in Paquette et al. (2013) and Lehuédé et al. (2014), are necessary. Passenger heterogeneity is another feature that might be considered, for instance, in Karabuk (2009) and Qu and Bard (2015). Some passengers may be ambulant, while others are dependent on a wheelchair or rolling walkers. This may affect the layout of the different vehicles and create the need for constraints ensuring vehicle/user compatibility (Ho et al., 2018).

Public transportation is another emerging application area. DAR services can, for instance, be a suitable alternative to traditional public transportation in low-demand periods, e.g., during nighttime or in low-demand areas. The latter is addressed in Garaix et al. (2011), where an on-demand transportation system in a rural zone in France is considered. The objective is to maximise passenger occupancy rate, encouraging people to meet up during transportation for social cohesion. Another area of research is the integrated DARP, where DAR services are integrated with ordinary public transportation services. Posada et al. (2017) consider an integrated system for the elderly and disabled in Sweden. Traditional public transportation is used for the majority of the journey, while DAR services transport the passengers to and from the public transportation stations. Other articles addressing public transportation are Häll et al. (2009) and Parragh et al. (2015).

3.2.2 Taxonomy

Ho et al. (2018) give two axes that are commonly used to classify DARPs: (1) static and dynamic, and (2) stochastic and deterministic. Static DARPs only utilise the given information once and make decisions a priori. In contrast, dynamic DARPs modify existing solutions in response to newly introduced information. The information is known with certainty in deterministic models, while stochastic models base their decisions on uncertain information. This generates four basic DARP variants: static-deterministic, static-stochastic, dynamic-deterministic, and dynamic-stochastic.

Table 3.2: Taxonomy of the DARP.

		Information known with certainty (at time of decision)?	
		Yes	**No**
Decisions can be modified in response to new information received?	**No**	Static and deterministic	Static and stochastic
	Yes	Dynamic and deterministic	Dynamic and stochastic

The static-deterministic DARP is by far the most researched category. The majority of reviewed papers considering this variant of the DARP focus on algorithmic advancement. The remaining literature mainly regards modelling issues concerning new problem features motivated by real-life applications of the DARP, such as heterogeneity of users and vehicles, passenger transfers, and workforce requirements (Ho et al., 2018).

The majority of research on dynamic-deterministic DARPs considers the accommodation of new requests. Ho et al. (2018) categorise the research done on dynamic deterministic DARPs as either theoretical or experimental. A substantial amount of the reviewed papers are theoretical research focusing on the development of online algorithms. Experimental research on dynamic-deterministic DARPs mostly focuses on developing simulations or dynamic models in which new requests are considered as events triggering the replanning procedure, thereby updating the current states.

The research on static-stochastic DARPs is relatively scarce compared with the other three types of models. The three papers found in the survey of Ho et al. (2018) consider only user arrivals. Hyytiä et al. (2010) model user arrivals as a Poisson process, Ho and Haugland (2011) model user requests with a given probability, while Heilporn et al. (2011) utilise stochastic arrival times at the pick-up points.

Dynamic-stochastic DARPs appear to be the most challenging among the four categories of DARPs. In this type of DARP, the decision-maker is repeatedly confronted with uncertainty regarding the appearance of future requests and operations regarding users who have already appeared. The stochastic element may for instance concern future user requests (Schilde et al., 2011), stochastic travel times (Schilde et al., 2014), or user no-shows (Xiang et al., 2008). In such problems, additional solution procedures, such as Monte-Carlo sampling, might be required to obtain solutions (Ho et al., 2018).

3.2.3 Objective Function

A wide variety of objective functions have been used in DARPs. Typical objectives are to minimise the service providers' operational costs and/or maximise the quality of service. A small amount of work also considers other objectives such as optimising passenger occupancy rate or minimising vehicle emissions (Ho et al., 2018).

Both operational costs and quality of service can be challenging to measure as they depend on multiple different metrics. Operational costs often include metrics such as total distance travelled by the vehicles, route duration, and the number of vehicles in use. Quality of service may include user waiting time, total ride time, and deviation from the fastest route (Nasri et al., 2021). The quality of service can either be enforced through hard constraints, which set conditions on variables that must be met, or soft constraints, which penalise variable values in the objective function if the conditions are not met (Colorni and Righini, 2001). Most of the reviewed literature considers time windows and maximum ride time as hard constraints. In contrast, excess ride time over direct time is often modelled as soft constraints and thus included in the objective function.

Although a significant number of DARPs only consider a single objective, the importance of both operational costs and quality of service has led to a substantial number of multi-objective DARPs. A multi-criterion objective captures more aspects of the problem. However, operational costs and quality of service are often conflicting measures that are difficult to balance. Ho et al. (2018) discuss three approaches to treat multi-objective DARPs. The first one is the weighted sum method, among others used in the early work of Psaraftis (1980). Psaraftis' objective function formulation seeks to minimise the time used to service all customers and the degree of customer "dissatisfaction". Customer "dissatisfaction" is defined as the sum of each customer's total waiting time and ride time. Weights between 0 and 1 are assigned to each measure according to their importance for the problem. The weighted sum method is widely used, but it is susceptible to the choice of weights. Consequently, it is mainly applicable to problems where the relative importance of the different objectives is known.

The second method uses a lexicographic objective function where the different objectives are optimised in order of importance. Garaix et al. (2010) use operational costs as their primary objective and quality of service as their second objective. Hence, when solving the problem, they first minimise the operational costs and then minimise the time lost by users. Compared with the weighted sum method, a lexicographic objective function has the advantage that it can be used when the objectives are represented by different units. However, it is only suitable when one objective is significantly more important than the other, i.e., operational costs are considerably more crucial than the quality of service or vice versa (Ho et al., 2018).

The last method is to obtain the Pareto frontier of the problem. A Pareto frontier is a set of non-dominated solutions. A solution is said to be Pareto optimal if no other set of solutions dominates it. That is, the solution cannot be better at one objective without being worse in any other objective (Paquette et al., 2013). By obtaining the Pareto frontier, the decision-maker gets information about several optimal solutions. Paquette et al. (2013) combine a tabu search heuristic with a multi-criteria algorithm to solve a multi-objective DARP. The algorithm generates a Pareto frontier of the problem to help the decision-maker better understand the trade-offs between the different objectives. Although the Pareto method surpasses the other methods in terms of information richness, the method has a higher computational cost. Consequently, it might not be suitable for dynamic applications where the problem needs to be frequently revised (Ho et al., 2018).

3.2.4 Restrictions

Restrictions differ less in the reviewed literature compared with the objective functions. Constraints regarding the vehicle fleet, time windows, and ride time are represented in most of the reviewed literature on the DARP.

Vehicle Fleet

The vehicle fleet is either capacitated or uncapacitated. Enforcing capacity constraints is highly motivated by real-life applications as no actual vehicle has an infinite number of seats. Consequently, the majority of the reviewed literature considers capacitated vehicles.

In the case of an uncapacitated fleet, there is no limit on the number of passengers in a vehicle. Hyytiä et al. (2012) deploy such a fleet. They consider a dynamic variant of the dial-a-ride problem with immediate trip requests and relaxed pick-up and drop-off constraints. Although the fleet is uncapacitated, the number of passengers in a vehicle rarely exceeds a certain limit. In fact, when enforcing a maximum capacity of ten passengers per vehicle, they obtained almost the same results as in the uncapacitated case.

Another essential characteristic of the vehicle fleet is whether it is homogeneous or heterogeneous. The existence of heterogeneous users has led to a growing number of papers considering the latter. In a heterogeneous fleet, vehicles may be differentiated based on their capacities or equipment. Garaix et al. (2010), Marković et al. (2015), and Vallee et al. (2020) all consider a fleet where each vehicle has its own specified capacity. Literature considering vehicles with different equipment is usually motivated by real-life problems of transporting people with limited mobility (Ho et al., 2018). For instance, some passengers may require vehicles with wheelchair ramps or additional space for rolling walkers. These problem features enforce restrictions on user-vehicle compatibility, thus increasing the problem complexity (Ho et al., 2018). Berbeglia et al. (2012) and Hyytiä et al. (2010) both consider homogeneous fleets with one universal capacity for all vehicles.

Time Windows

Time windows give a lower and upper bound for when a request must be served. In the DARP, these are usually constructed at a fixed width around the user's desired pick-up and drop-off time. Psaraftis (1980) and Hyytiä et al. (2012) are the only reviewed articles where time windows are not implemented. In Hyytiä et al. (2012) they consider a purely dynamic problem where they instead try to minimise the total service time from trip request to delivery. Psaraftis (1980) imposes "maximum position shift" constraints which limit the difference between a user's position in the calling list and the vehicle route.

The remaining literature enforces time windows through either hard or soft constraints. Paquay et al. (2020), Vallee et al. (2020), and Gaul et al. (2021) define hard time windows for either pick-up or drop-off, while Berbeglia et al. (2012) specify it for both. Paquette et al. (2013) use a combination of hard and soft time windows. They enforce hard limits on time windows that must hold and soft limits where the deviation is allowed but penalised in the objective function.

Ride Time

The ride time is the time a user spends in the vehicle, i.e., the difference between the pick-up and drop-off time. Berbeglia et al. (2012) and Paquay et al. (2020) define one universal maximum ride time for all requests and enforce it through hard constraints. Similarly, Paquette et al. (2013) define two different maximum ride times, a smaller one associated with trips within the territory and a larger one imposed on trips going outside the territory. It is also possible to define specific maximum ride times for each request, as done in Garaix et al. (2010). As users have individual preferences, such an approach may increase the quality of service (Nasri et al., 2021).

The maximum ride time can also be defined based on the direct travel time, such as in Marković et al. (2015). They define the maximum ride time as the direct travel time from the pick-up location to the drop-off location multiplied by a predefined ride time coefficient. Vallee et al. (2020) select the coefficient from a set of predefined values depending on the value of the direct travel time. Gaul et al. (2021) is the only reviewed article enforcing maximum ride times through both hard and soft constraints. They define a multi-objective function that minimises the total routing cost, excess ride time, and the number of unaccepted requests. In addition, they enforce an upper limit on the ride time through hard constraints.

3.2.5 Solution Method

Different solution methods have been developed for the DARP and its variants. Due to the NP-hardness of the problem, most research focuses on developing efficient and effective heuristics. However, exact methods have also been developed (Ho et al., 2018). This section discusses both exact and heuristic methods for the DARP. The heuristics can be further distinguished into several subcategories: insertion heuristics, local search-based metaheuristics, population-based metaheuristics, and hybrid methods. As the problem considered in this thesis consists of both a static and a dynamic part, solution methods for both variants of the DARP are presented.

Exact Methods

Most of the literature focusing on developing exact methods for DARPs considers the static and deterministic case. This is because the majority of stochastic variants are too computationally complex, while dynamic variants need solutions to be generated in a timely manner that exact methods often cannot deliver (Ho et al., 2018). Exact solution methods are usually based on the branch-and-bound framework (Ho et al., 2018), such as branch-and-cut (Cordeau, 2006; Heilporn et al., 2011), branch-and-price (Garaix et al., 2011; Parragh et al., 2015), and branch-and-price-and-cut. The latter is the basis for the method proposed by Luo et al. (2019). They propose a two-phase branch-and-price-and-cut algorithm for a static-deterministic DARP in patient transportation. The algorithm is tested on 42 real-life test instances and can solve instances with up to 36 requests to optimality in four hours.

Insertion Heuristics

To solve real-life instances, heuristics and metaheuristics are developed. Among these are several simple insertion heuristics. Much of the work done on insertion heuristics is inspired by the greedy insertion heuristic by Jaw et al. (1986), where users are sorted according to their earliest feasible pick-up time and gradually inserted into vehicle routes such that the increase in objective value is minimised. Although less effective than metaheuristics, greedy insertion heuristics can construct viable solutions to large problems within a reasonable time. Consequently, they are widely used for the dynamic DARP or to initialise more complex methods (Ho et al., 2018).

Among the papers implementing an insertion heuristic for the dynamic-deterministic DARP is Souza et al. (2022). They propose a two-stage heuristic algorithm to solve the problem. The first stage happens at the beginning of the workday and constructs an initial route plan considering the requests known at that time. This constitutes the static part of the problem and is solved by a General Variable Neighborhood Search (GVNS). The second stage is an online problem where they deal with dynamic requests. As this requires requests to be handled in a timely manner, they implement an insertion heuristic. The heuristic assigns the request to the closest available vehicle, and then re-optimises the rest of that vehicle's route.

Local Search-Based Metaheuristics

Among the papers considering local search-based methods is Vallee et al. (2020). They combine an online insertion heuristic with a reinsertion heuristic to solve a dynamic-deterministic DARP. The objective of the reinsertion heuristic is to find feasible insertions of requests when the online insertion heuristic fails to find a solution. A total of three reinsertion heuristics are proposed. One uses destroy and repair neighbourhood operators, while the other two are based on the ejection chain concept. The results show that all reinsertion procedures improve the solution when compared with only using the online algorithm, with an increase in the number of served requests and a reduced number of vehicles.

Another algorithm based on destroy and repair methods is the Adaptive Large Neighborhood Search (ALNS). Pfeiffer and Schulz (2022) propose a dynamic programming algorithm (DP) and an ALNS to solve a variant of the static-deterministic DARP with ride and waiting time minimisation. Results show that while the DP is only suited for small instances, the ALNS is also effective for larger problems. A variant of the ALNS used to solve a dynamic DARP can be found in Syed et al. (2019). They evaluate the performance of a rolling horizon ALNS on a real-time ride-hailing problem. To test the ALNS, they use a simulation framework based on NYC taxi data. Results show that the ALNS can provide high-quality solutions also when only given a limited amount of time for optimisation.

Population-based Metaheuristics

Several population-based metaheuristics have been successfully applied to the DARP (Ho et al., 2018). Muñoz-Carpintero et al. (2015) develop an evolutionary algorithm to solve the dynamic-deterministic DARP within a hybrid predictive control (HPC) framework. The HPC formulation of the DARP incorporates stochasticity into the routing decisions by considering the impact of future requests on already scheduled customers. The algorithm uses different configurations of Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). Results show that the PSO strategies are more efficient than those based on GA, both in terms of speed and accuracy. Other papers proposing a GA for the DARP are Núñez et al. (2014) and Atahran et al. (2014), both considering a multi-objective framework. More recent research on population-based methods primarily focuses on including them as part of hybrid methods, as discussed below.

Hybrid Solution Methods

A growing trend in recent years is hybrid algorithms combining metaheuristics with other metaheuristics, mathematical programming, or constraint programming. Several successful hybrid algorithms combine a population-based metaheuristic with a single-solution method, for instance, Masmoudi et al. (2017). They consider a static-deterministic DARP with both heterogeneous users and vehicles. To solve the problem, a hybrid method combining a GA with several local search techniques is implemented. While the GA performs well in global search, the local search techniques serve to intensify the search. Consequently, by incorporating the local search into the GA, the algorithm is able to improve convergence and reduce computational time.

Another example of an effective hybrid algorithm is Belhaiza (2019). They propose an ALNS and GA hybrid method to solve a static-deterministic DARP. The GA provides diversification, while the ALNS is used for intensification. The algorithm is tested on real-life instances considering the transportation of elderly and impaired people in Vancouver. Results show that the algorithm reduced route durations by 5% on the real-life instances and by 4.5% on benchmark instances.

3.3 Disruption Management

Disruption management can be defined as the real-time dynamic alteration of an operational plan in response to unforeseen events (Yu and Qi, 2004). It is central when an initial plan has to be published in advance, but may be subject to several severe, random disruptions. Applications of disruption management range from production planning and supply chain coordination, to flight scheduling and logistics delivery (Wang et al., 2012). This section firstly discusses general key factors of disruption management. Then research on the disrupted DARP is presented.

3.3.1 Characteristics of Disruption Management

Eglese and Zambirinis (2018) summarise several key factors of disruption management. Firstly, the time for replanning may be limited. While it is practically acceptable to use several minutes or even hours to generate the initial plan, a disruption requires an immediate response. For large-scale problems, this is not necessarily trivial, thus calling for the implementation of real-time optimisation techniques. Secondly, there is always an initial plan to consult in disruption management. Consequently, there is no need to determine a complete plan from scratch when disruptions occur. Instead, the initial undisrupted plan may serve as a foundation for the new plan.

Another key consideration is the degree of deviation from the original plan. Specifically, too many undesirable changes may be difficult or even infeasible to allow due to human, organisational, or other issues. This is solved by introducing so-called deviation costs that force the revised plan to stay close to the original one. The costs may be quantifiable costs resulting from extra personnel or utilising a standby vehicle or non-quantifiable costs such as customer dissatisfaction caused by waiting and delays.

Lastly, the disruptions may result in new constraints not included in the initial plan. This may include the unavailability of a broken vehicle or a blocked road due to an accident.

3.3.2 Disruption Management in DARP

As mentioned, disruption management concerns the real-time alteration of the plan in response to unforeseen events. Consequently, a DARP considering disruptions is a variant of the dynamic DARP. Research on the dynamic DARP, as well as other variants of the dynamic VRP, has grown considerably during the last two decades (Ojeda Rios et al., 2021). However, as mentioned in Section 3.2.2, most research on dynamic DARPs only considers the accommodation of new user requests. In reality, DAR services may experience other types of disruptions, such as delays in vehicle arrival times or vehicle breakdowns.

Among the research considering the disruption of new user requests are the papers of Souza et al. (2022), Vallee et al. (2020), and Syed et al. (2019) presented in Section 3.2.5. Souza et al. (2022) closely resembles the problem presented in this thesis, where a static DARP is solved at the start of the day before re-optimising the problem as on-demand requests arrive throughout the day. Vallee et al. (2020) and Syed et al. (2019) consider a purely dynamic problem, hence no requests are known at the start of the day, and there is no initial plan to consult.

Other types of disruptions are considered in Paquay et al. (2020). They propose a reactive algorithm to handle three types of disruptions. These are delays, reinsertion of requests, and cancellations of requests. Reinsertion of requests only occurs as a consequence of previous delays. Hence, the insertion of new requests is not considered. When a request has to be reinserted due to a previous delay, four distinct, simple reinsertion operators are applied to reinsert the request successfully. The operators include pure reinsertion, destroy-and-repair, relaxed time windows, and extended working hours. If the first operator fails to provide a feasible insertion, the second one is applied and so on. Computational experiments prove the algorithm's effectiveness and show that the use of simple reinsertion operations is most often sufficient to rebuild the solution.

Another paper considering disruption management for the DARP is Ramasamy Pandi et al. (2020). They formulate and solve the disruptive DARP with vehicle breakdowns. A vehicle breakdown may affect a considerable number of requests, thus resulting in high recovery costs. To address the issue, they propose a GPU-based adaptive neighbourhood search (G-ALNS) algorithm combined with a fleet size minimisation (FSM) strategy. Results show that the G-ALNS algorithm can successfully reinsert the affected request, while the FSM strategy leads to a 15% reduction in operational costs under disruption.

3.4 Comparison of Reviewed Literature

Table 3.4 presents a comparison of a selection of the reviewed literature. The articles are compared according to suitable features previously discussed in this chapter. Features in common with our problem are marked in blue.

Table 3.3: Comparison of reviewed literature.

Reference	Taxonomy	Heterogeneity	Rejection	Disruptions	Objectives	Solution method
Our problem	**Dynamic deterministic**	**Users**	**Yes**	**Request, delay, cancellation, no-show**	**Operational, quality of service**	**ALNS**
Berbeglia et al. (2012)	Dynamic deterministic	-	Yes	Request	Operational, quality of service	Hybrid algorithm
Hyytiä et al. (2012)	Dynamic stochastic	-	No	Request	Operational, quality of service	Insertion heuristic
Paquette et al. (2013)	Static deterministic	Vehicles	No	-	Operational, quality of service	Tabu search
Marković et al. (2015)	Dynamic deterministic	Vehicles	No	Request	Operational	Insertion heuristic
Posada et al. (2017)	Static deterministic	Vehicles	No	-	Operational, quality of service	Branch-and-bound
Masmoudi et al. (2017)	Static deterministic	Users and vehicles	No	-	Operational, quality of service	GA
Vallee et al. (2020)	Dynamic deterministic	Vehicles	Yes	Request	Operational, quality of service	Reinsertion heuristic
Paquay et al. (2020)	Dynamic deterministic	Users and vehicles	Yes	Delay, cancellation	Quality of service	Reinsertion heuristic
Ramasamy Pandi et al. (2020)	Dynamic deterministic	-	No	Vehicle breakdown	Operational	ALNS
Gaul et al. (2021)	Dynamic deterministic	Vehicles	Yes	Request	Operational, quality of service	Rolling-Horizon branch-and-bound
Pfeiffer and Schulz (2022)	Static deterministic	-	No	-	Quality of service	ALNS
Souza et al. (2022)	Dynamic deterministic	Vehicles	No	Request	Operational, quality of service	GVNS

3.5 Motivation of the Thesis

DAR services are becoming increasingly relevant due to the upcoming age wave and the current focus on sustainable transportation. As outlined in Section 3.2.2, static-deterministic DARPs are the most researched variant of the DARP. This may be because traditionally, DAR services have depended on centralised planning requiring booking of trips in advance. However, as communication technologies have evolved, DAR services have progressed from being based on advance reservations to real-time planning and dispatching (Ramasamy Pandi et al., 2020). This creates the need to develop dynamic DARPs, such as the one addressed in this thesis.

As presented in Section 3.2.5, a wide range of solution methods have been proposed for the DARP. For the dynamic DARP, greedy insertion heuristics are widely used. These can construct viable solutions to large problems within a reasonable time. Hence, they are well-suited for real-time processing of new requests or to initialise more complex methods. The drawback of greedy insertion heuristics is that they can only apply small modifications to the current solution. Therefore, many choose to combine them with more complex methods. By using destroy and repair operators, an ALNS heuristic can modify large parts of the solution space in one single iteration. According to Belhaiza (2019), this has been revealed to be effective for tightly constrained problems, such as the DARP. Consequently, this thesis combines a greedy insertion heuristic with an ALNS heuristic.

This thesis' contributions to the field of operational research is threefold. Firstly, although the research on the dynamic DARPs has grown considerably during the last decades, the application of disruption management is limited. In fact, Ho et al. (2018) highlight disruption management as one of the promising areas for future research on the DARP. Disruption management has been widely applied to other problems, such as supply chain coordination and the VRP. Hence, it is likely to help reduce the impact of unforeseen events when applied to the DARP. Therefore, developing a fast and effective solution method for the disruptive DARP is one of the main goals of this thesis.

The second contribution is the wide range of disruption types considered. As previously mentioned, most research on dynamic DARPs only considers the accommodation of new user requests. However, in most real-life transportation problems, other types of events are likely to affect the service in focus. To more accurately model the real-life problem, other disruption types should be considered. In addition to the accommodation of new requests, this thesis includes disruptions related to delays in vehicle arrival times, cancellations of requests, and user no-shows.

Finally, the last contribution of our thesis is the developed simulation framework used to generate disruptions. A well-known way to simulate discrete events is to generate them using a homogeneous Poisson process, for instance, as in Schilde et al. (2011). The drawback of using a homogeneous Poisson process is that it has a constant arrival rate, and hence it is not able to capture the time-dependent nature of the arrival of disruptions. Therefore, to more accurately imitate the real-life problem in subject, this thesis uses inhomogeneous Poisson processes to generate the disruption times. Overall, the research of this thesis contributes to existing literature and creates a foundation for further research on the area of disruption management in the dynamic DARP.

Chapter 4

Problem Description

This chapter presents the DARP to be optimised. Section 4.1 provides a general overview of the problem, including a description of the initial plan and the different types of disruptions. Further, Section 4.2 discusses the characteristics and restrictions of the requests, users, and vehicle fleet. Finally, Section 4.3 presents the objectives of the problem.

4.1 Overview

We consider a DAR service where we receive requests for transportation between different locations and have to decide if and how to serve them. The DAR service is subject to different types of disruptions. Hence, the route plan may have to be modified several times throughout the day. The overall goal of the problem is to serve as many requests as possible, while minimising the total driving time and ensuring a sufficient degree of quality of service.

At the beginning of each day, an initial route plan is created. The initial plan consists of requests ordered before 10:00 the same day. The plan consists of the assignment of requests to vehicles, the route for each vehicle, and the time of service for each request. A disruption is defined as an unforeseen event that may occur during the operational hours of the DAR service and disrupt the initial plan. The disruptions may be related to the customer itself or result from traffic congestion, blocked roads, and other factors affecting the travel times. In this thesis, we distinguish between four different types of disruptions:

- *New request.* Customers can order rides on-demand. If the request is accepted, the new request must be inserted into the route plan.
- *Delay.* Delays are disruptions that occur when a scheduled route takes longer time than planned. This might be due to increased travel times between requests or delayed customers, resulting in increased service times.
- *Cancellation.* Some requests may be cancelled by the customer in advance. The request can then be removed from the route plan.
- *No-show.* A no-show is defined as an event in which the customer has previously scheduled a ride, but fails to appear for it when the service vehicle arrives at the pick-up location. Then, the drop-off location of the request can be removed from

the route plan. We assume that a customer either is at the pick-up location when the vehicle arrives or does not show up at all.

4.2 Problem Features

This section describes the main problem features of the DARP to be optimised. This includes a description of restrictions regarding the requests, passengers, and vehicles.

In the DARP, multiple users make their requests for transportation between their desired origin and destination. Each request has a specific pick-up and drop-off location and a specified time for when the user must leave its pick-up location or arrive at its drop-off location. A request cannot specify both a pick-up and a drop-off time. Time windows are constructed at a fixed width around the desired pick-up or drop-off time. There are two different types of time windows: an absolute time window and a preferred one. The former is a hard restriction, hence it cannot be violated. The latter is a soft restriction, thus, it can be violated for a penalty cost in the objective function.

The passengers are divided into passengers who can utilise standard seating and wheelchair users. Consequently, the requests also specify the number of wheelchair passengers and the number of standard seating passengers. Passengers in the same request must be transported by the same vehicle. Further, a request may be moved to another vehicle as long as the request is not currently being served or is already served. The DAR service allows ride-sharing, hence multiple requests may be served by the same vehicle simultaneously, even though their pick-up/drop-off locations differ. To avoid too large detours, a maximum ride time is defined for all requests.

The vehicle fleet is homogeneous, hence all vehicles have the same capacities and features. The capacity is divided between standard seats and wheelchair seats. Wheelchair seats can only be used by wheelchair users and vice versa. Each vehicle also has a specified origin where it must start its day. The origin may vary between the different vehicles. The vehicles do not have a specified destination, hence they end their day at the drop-off location of their last request. If a disruption occurs that affects a vehicle's next location while the vehicle is currently on its way there, the information is revealed only when arriving at the location.

4.3 Objective

The objective function consists of operational costs and penalty costs associated with reductions in the quality of service. Operational costs include the total travel time of the service vehicles. Quality costs occur for deviation outside requests' time windows, rejection of requests, and deviations from previously specified pick-up and drop-off times. The latter is not included when creating the initial plan as there are no previously specified pick-up and drop-off times.

Chapter 5

Mathematical Model

This chapter presents the mathematical model for the dial-a-ride problem (DARP) presented in this thesis, formulated as a mixed-integer linear program (MILP). The mathematical model is explained in this chapter to give insights into the DARP, but is not utilised as the underlying model to the solution method presented in Chapter 6. Further, the model is for the initial problem, i.e. when constructing the initial vehicle routes at the start of the day. To model the re-optimisation problem after a disruption, some additional modelling and notation are required, however, most of the model remains unchanged. The mathematical model is based on the model for the initial set-up in the report on RAT by Brenna et al. (2021).

The mathematical model for the DARP is presented as follows. Section 5.1 describes the assumptions made in the mathematical formulations. Next, Section 5.2 introduces the notation used, including the sets and their related indices, the parameters, and the decision variables for the initial model. The objectives and constraints of the model are presented in Section 5.3.

5.1 Assumptions

The problem is defined as a graph. A request's pick-up and drop-off locations are represented through nodes. Each pick-up and drop-off node has soft and hard time windows for the time of service. The time of service is here defined as the time the vehicle leaves the node. The vehicles' origin and artificial destination nodes do not have associated time windows. Moreover, the cost of travelling directly from a node to a vehicle's artificial destination is zero. If two locations are located at the same place, they are represented by two different nodes, but with zero distance between them. Lastly, the arcs in the graph represent a vehicle driving from one node to another, and it is assumed that the same vehicle must pick up and drop off a customer.

5.2 Notation

Sets

$\mathcal{K}$	-	set of vehicles $k \in \{1, ..., K\}$
$\mathcal{P}$	-	set of pick-up nodes $i, j \in \{1, ..., n\}$
$\mathcal{D}$	-	set of drop-off nodes $i, j \in \{n+1, ..., 2n\}$
$\mathcal{N}$	-	set of pick-up and drop-off nodes $i, j \in \{1, ..., 2n\}$
$\mathcal{N}^D$	-	set of all nodes $i, j \in \{1, ..., 2n, o(1), ..., o(K), d(1), ..., d(K)\}$
$\mathcal{A}$	-	set of all feasible arcs $(i, j) \in \mathcal{A} \subset \mathcal{N}^D \times \mathcal{N}^D$

Parameters

C^D_{ijk}	-	cost of traveling the distance from node i to node j using vehicle k
C^T	-	penalty cost for violation of soft time window
C^R	-	penalty cost for rejecting a request
Q^S_k	-	standard seats capacity of vehicle k
Q^W_k	-	wheelchair capacity of vehicle k
$o(k)$	-	origin of vehicle k
$d(k)$	-	artificial destination of vehicle k
L^S_i	-	standard seats load of request i
L^W_i	-	wheelchair load of request i
T_{ij}	-	direct traveling time from node i to node j
$\underline{T}^S_i$	-	lower bound of soft time window for time of service of request i
$\overline{T}^S_i$	-	upper bound of soft time window for time of service of request i
$\underline{T}^H_i$	-	lower bound of hard time window for time of service of request i
$\overline{T}^H_i$	-	upper bound of hard time window for time of service of request i
S_i	-	duration of embarking and disembarking of passenger(s) of request i at a node
F	-	fraction of direct travel time the actual travel time can exceed
α	-	weight of driving time term in objective function
β	-	weight of soft time window violation term in objective function

Decision variables

s_i	-	$\begin{cases} 1, & \text{if request } i \text{ is not served} \\ 0, & \text{otherwise} \end{cases}$
x_{ijk}	-	$\begin{cases} 1, & \text{if vehicle } k \text{ drives directly from node } i \text{ to node } j \\ 0, & \text{otherwise} \end{cases}$
q_{ik}^S	-	standard seats load of vehicle k when leaving node i
q_{ik}^W	-	wheelchair load of vehicle k when leaving node i
t_i	-	time of service of node i
l_i	-	violation of lower bound of soft time window for time of service of node i
u_i	-	violation of upper bound of soft time window for time of service of node i

5.3 Modeling

Objective function

$$\min \alpha \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} C_{ijk}^D x_{ijk} \tag{5.1}$$

$$+\beta \sum_{i \in \mathcal{N}} C^T (l_i + u_i) \tag{5.2}$$

$$+\sum_{i \in \mathcal{P}} C^R s_i \tag{5.3}$$

The goal of the objective function is to minimise the sum of operational costs and penalties caused by reductions in quality of service, henceforth referred to as quality costs. Objective term (5.1) defines the operational costs, specifically the cost of driving between nodes. The quality costs consist of objective terms (5.2) and (5.3). Objective term (5.2) sums the soft time window violations across all requests, while (5.3) sums the penalties from rejecting a request. The driving costs and the soft time window violation costs are weighted relative to each other, where α is the weighting of the driving time and β is the weighting of the quality costs.

Flow constraints

$$\sum_{j \in \mathcal{N}^D} \sum_{k \in \mathcal{K}} x_{ijk} = 1 - s_i \qquad i \in \mathcal{P} \tag{5.4}$$

$$\sum_{j \in \mathcal{N}^D} x_{o(k)jk} = \sum_{i \in \mathcal{N}^D} x_{id(k)k} \quad = 1 \qquad k \in \mathcal{K} \tag{5.5}$$

$$\sum_{j \in \mathcal{N}^D} x_{jik} - \sum_{j \in \mathcal{N}^D} x_{ijk} = 0 \qquad i \in \mathcal{N},\ k \in \mathcal{K} \tag{5.6}$$

$$\sum_{j\in\mathcal{N}^D} x_{ijk} - \sum_{j\in\mathcal{N}^D} x_{n+i,j,k} = 0 \qquad i\in\mathcal{P},\ k\in\mathcal{K} \tag{5.7}$$

Constraints (5.4) ensure that an accepted request is served exactly once and that a rejected request is not visited. Furthermore, constraints (5.5) limit the flow in and out of origin and destination nodes, while constraints (5.6) are flow balance constraints. Finally, constraints (5.7) ensure that the pick-up and drop-off node of a request is visited by the same vehicle.

Capacity constraints

$$q^S_{o(k)k} = 0 \qquad k\in\mathcal{K} \tag{5.8}$$

$$q^S_{ik} + L^S_j - q^S_{jk} \leq (Q^S_k + L^S_j)(1 - x_{ijk}) \qquad j\in\mathcal{P},\ (i,j)\in\mathcal{A},\ k\in\mathcal{K} \tag{5.9}$$

$$q^S_{ik} - L^S_j - q^S_{n+j,k} \leq Q^S_k(1 - x_{i,n+j,k}) \qquad j\in\mathcal{P},\ (i,n+j)\in\mathcal{A},\ k\in\mathcal{K} \tag{5.10}$$

$$\sum_{j\in\mathcal{N}^D} L^S_i x_{ijk} \leq q^S_{ik} \leq \sum_{j\in\mathcal{N}^D} Q^S_k x_{ijk} \qquad i\in\mathcal{P},\ k\in\mathcal{K} \tag{5.11}$$

$$\sum_{j\in\mathcal{N}^D} (Q^S_k - L^S_i) x_{n+i,j,k} \geq q^S_{n+i,k} \qquad i\in\mathcal{P},\ k\in\mathcal{K} \tag{5.12}$$

$$q^S_{ik} \leq Q^S_k(1 - x_{id(k)k}) \qquad i\in\mathcal{D},\ k\in\mathcal{K} \tag{5.13}$$

$$q^W_{o(k)k} = 0 \qquad k\in\mathcal{K} \tag{5.14}$$

$$q^W_{ik} + L^W_j - q^W_{jk} \leq (Q^W_k + L^W_j)(1 - x_{ijk}) \qquad j\in\mathcal{P},\ (i,j)\in\mathcal{A},\ k\in\mathcal{K} \tag{5.15}$$

$$q^W_{ik} - L^W_j - q^W_{n+j,k} \leq Q^W_k(1 - x_{i,n+j,k}) \qquad j\in\mathcal{P},\ (i,n+j)\in\mathcal{A},\ k\in\mathcal{K} \tag{5.16}$$

$$\sum_{j\in\mathcal{N}^D} L^W_i x_{ijk} \leq q^W_{ik} \leq \sum_{j\in\mathcal{N}^D} Q^W_k x_{ijk} \qquad i\in\mathcal{P},\ k\in\mathcal{K} \tag{5.17}$$

$$\sum_{j\in\mathcal{N}^D} (Q^W_k - L^W_i) x_{n+i,j,k} \geq q^W_{n+i,k} \qquad i\in\mathcal{P},\ k\in\mathcal{K} \tag{5.18}$$

$$q^W_{ik} \leq Q^W_k(1 - x_{id(k)k}) \qquad i\in\mathcal{D},\ k\in\mathcal{K} \tag{5.19}$$

Constraints (5.8) and (5.14) set the load at origin to zero for all vehicles. Constraints (5.9) - (5.10) and (5.15) - (5.16) ensure that the load is updated correctly at the pick-up and drop-off nodes for both standard seating and wheelchair seating. Constraints (5.11) and (5.17) impose upper and lower limits on load after pick-up, while constraints (5.12) and (5.18) impose an upper limit on load after drop-off. Finally, constraints (5.13) and (5.19) ensure that the load when entering the destination node is zero.

Time window constraints

$$\underline{T}_i^S - l_i \leq t_i \leq \overline{T}_i^S + u_i \qquad i \in \mathcal{N} \tag{5.20}$$

$$\underline{T}_i^H \leq t_i \leq \overline{T}_i^H \qquad i \in \mathcal{N} \tag{5.21}$$

$$t_i + T_{ij} + S_j - t_j \leq M_{ij}(1 - x_{ijk}) \qquad i, j \in \mathcal{N},\ k \in \mathcal{K} \tag{5.22}$$

$$t_i + T_{i,n+i} + S_{n+i} - t_{n+i} \leq 0 \qquad i \in \mathcal{P} \tag{5.23}$$

Constraints (5.20) give the soft time window for the requests and ensure that violations of the soft time windows are penalized in the objective function. Furthermore, constraints (5.21) state that all requests must be served within the hard time windows. Constraints (5.22) and (5.23) define the time of service of each node. In constraints (5.22), M_{ij} is defined as $\overline{T}_i^H + T_{ij} + S_j - \underline{T}_j^H$.

Ride time constraints

$$t_{n+i} - S_{n+i} - t_i \leq (1 + F)T_{i,n+i} \qquad i \in \mathcal{P} \tag{5.24}$$

Constraints (5.24) ensure that the actual travel time does not exceed the allowed fraction of the direct travel time.

Non-negativity and binary constraints

$$s_i \in \{0, 1\} \qquad i \in \mathcal{P} \tag{5.25}$$

$$x_{ijk} \in \{0, 1\} \qquad (i, j) \in \mathcal{A},\ k \in \mathcal{K} \tag{5.26}$$

$$q_{ik}^S, q_{ik}^W \geq 0,\ \text{integer} \qquad i \in \mathcal{N}^D,\ k \in \mathcal{K} \tag{5.27}$$

$$t_i, l_i, u_i \geq 0 \qquad i \in \mathcal{N} \tag{5.28}$$

Constraints (5.25) enforce binary restrictions on variables s_i, while constraints (5.26) enforce binary restrictions on variables x_{ijk}. Furthermore, constraints (5.27) enforce integer requirements on the load variables q_{ik}^S and q_{ik}^W. Constraints (5.28) ensure non-negativity of the remaining decision variables.

Chapter 6

Solution Method

This chapter presents the solution method proposed to solve the DARP presented in Chapter 4. Section 6.1 introduces the solution representation, while an overview of the solution method is presented in Section 6.2. In Section 6.3, the construction of the initial solution is explained. Then, Section 6.4 presents the ALNS heuristic used to improve the initial solution and re-optimise the solution in case of disruptions. Finally, Section 6.5 explains how the solution method handles incoming disruptions.

6.1 Solution Representation

A DARP solution is represented using a matrix structure, similar to the one presented in Souza et al. (2020). The matrix consists of $|\mathcal{K}|$ rows, one for each vehicle in use. Each row represents the route assigned to the vehicle, such that the kth row refers to the route of the kth vehicle. The column indices represent the order of the nodes in a specific route, hence the leftmost node in a row is visited first in that route.

Figure 6.1 shows a solution representation for an instance with four vehicles and twelve requests. The nodes with integer numbers represent pick up locations, while those with fractional numbers represent drop-off locations. For instance, the node with id 1 is the pick-up location of request 1, while the node with id 1.5 is the corresponding drop-off location. All vehicles start at their depot, represented by node id 0.

1	0	1	3	1.5	3.5	5	5.5
2	0	2	2.5	6	6.5		
3	0	7	9	7.5	9.5	11	11.5
4	0	8	8.5	10	10.5		
5	0	12	12.5				

Figure 6.1: Solution representation of instance with five vehicles and 12 requests. The leftmost numbers represents the vehicles.

Each cell in the matrix consists of a tuple, (i, t, d, q^S, q^W), where i is the node id, t is the time of when the vehicle finishes service of the node, and d is the total deviation from the requested pick-up/drop-off time. Further, q^S is the number of occupied standard seats of the vehicle when leaving the node, while q^W is the number of occupied wheelchair seats. In Figure 6.1, only the node id is shown.

6.2 Overview of Solution Method

An overview of the proposed solution method is illustrated in Figure 6.2. It consists of four main parts: a greedy construction heuristic, an ALNS heuristic, a greedy insertion heuristic and a disruption updater. Additionally, a simulator is used to generate disruptions. The simulator will be further explained in Chapter 7.

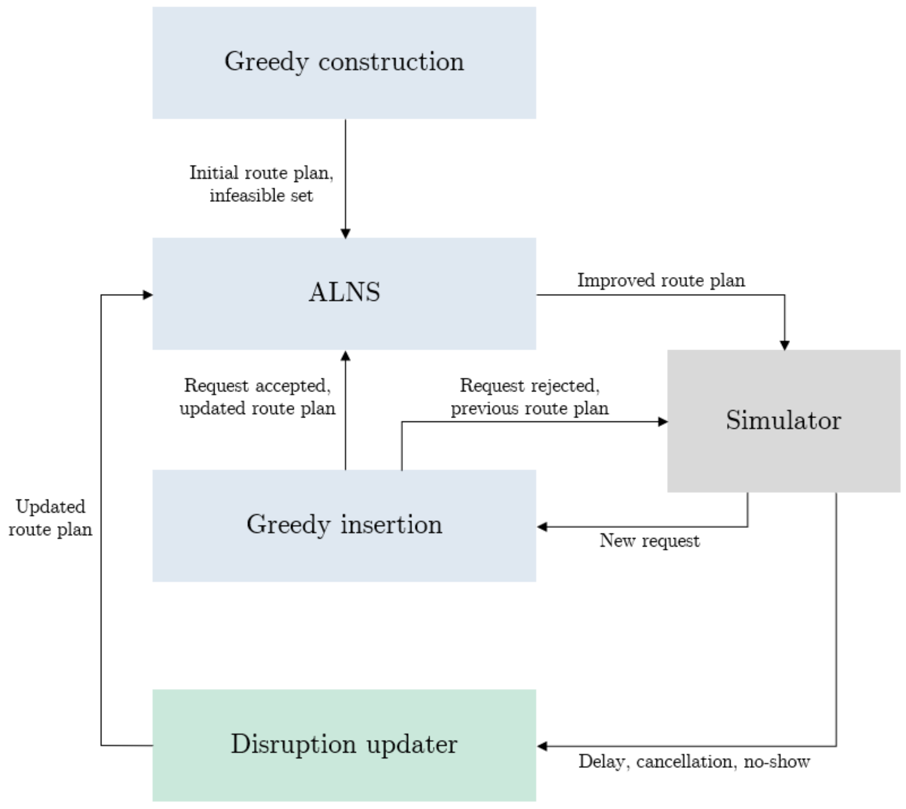

Greedy construction
Initial route plan, infeasible set
ALNS
Improved route plan
Request accepted, updated route plan
Request rejected, previous route plan
Simulator
Updated route plan
Greedy insertion
New request
Disruption updater
Delay, cancellation, no-show

Figure 6.2: Overview of solution method.

First, the greedy construction heuristic creates an initial solution at the start of each day for the requests received before 10:00. The initial solution consists of an initial route plan and an infeasible set, namely the set of requests not successfully inserted by the construction heuristic. The initial solution is then sent to the ALNS, which attempts to improve the solution. The ALNS also tries to insert the infeasible requests into the route plan. Requests not inserted by neither the construction heuristic nor the ALNS are rejected, and a rejection cost is added to the objective function.

Next, the improved route plan is sent to the simulator, which generates a disruption. In the case of a new request, a greedy insertion heuristic tries to insert the request into the current route plan quickly. To accommodate the new request, the heuristic tries to adjust the service time of the nodes already in the route plan, while remaining within their absolute time windows. If successful, the request is accepted, and the updated route plan is sent to the ALNS for improvement. If the greedy insertion heuristic cannot find a feasible solution, the request is rejected, and the route plan remains unchanged.

The other disruption types are first handled by a disruption updater. The disruption updater adapts the current route plan to include the disruption, for instance, by removing a cancelled request. The updated route plan is sent to the ALNS for improvement. The route plan from the ALNS is then sent back to the simulator, which generates a new disruption. This loop continues until the simulator reaches the end of the operational day.

6.3 Construction of Initial Solution

This section presents the heuristic used for the construction of the initial solution. It is a greedy insertion heuristic, similar to the one used in Marković et al. (2015). Greedy insertion heuristics can construct good feasible solutions in a reasonable time and have been successfully applied to DARPs before (Ho et al., 2018). The pseudocode of the greedy construction heuristic is shown in Algorithm 1.

Algorithm 1 Greedy Construction Heuristic

```
1: Input: set of requests $N$
2: Input: set of vehicles $K$
3: Initialize set of unassigned requests, $U := N$
4: Initialize infeasible set, $I := \emptyset$
5: Initialize set of introduced vehicles, $K^I := K.pop()$
6: Initialize solution, $x := \{\}$
7: while $U \neq \emptyset$ do
8:     Select $i \in U$ with earliest pick-up time
9:     Initialize the set of all feasible insertions of $i$ in $x$, $P_i := \{\}$
10:    for $k \in K^I$ do
11:        Generate all feasible insertions of $i$ in $k$ in $x$, $p_1, ..., p_m$
12:        Append $p_1, ..., p_m$ to $P_i$, $P_i := P_i + \{p_1, ..., p_m\}$
13:    end for
14:    if $P \neq \emptyset$ then
15:        Update solution $x$ to best insertion of $i$, $p^b \in P_i$, $x := p^b$
16:        $U := U \setminus \{i\}$
17:    else
18:        if $K \neq \emptyset$ then
19:            $k := K.pop()$
20:            $K^I := K^I + \{k\}$
21:            Generate insertion of $i$ in $k$ to $x$, $p_1$
22:            Update solution $x$ to insertion of $i$, $p_1$, $x := p_1$
23:        else
24:            $I := I + \{i\}$
25:        end if
26:    end if
27: end while
28: return $x$, $I$
```

The heuristic works by sorting the requests by requested pick-up time and then inserting them one at a time according to the earliest requested pick-up time. If a request has a requested drop-off time, an artificial pick-up time is used for that request. The calculation of artificial pick-up times is explained more in detail in Section 8.2. When inserting a request, the heuristic first generates all feasible insertions of the request into already introduced vehicles. Here, an introduced vehicle is defined as a vehicle which already has a route with requests. It then inserts the request into the position yielding the lowest costs. If no feasible insertions are found, a new vehicle is introduced, and the request is assigned to it. If all vehicles have been introduced and no feasible insertions are found, the request is added to an infeasible set.

Requests already inserted into the route plan cannot be rejected to accommodate the new request, but their service times may be adjusted. Specifically, the service time of a node can be updated to any time within the absolute time window of the node. Consequently, the set of all feasible insertions, P_i, in line 9 in Algorithm 1, consists of all insertions of request i that satisfy the absolute time windows of the nodes, the ride time constraints, the capacity constraints of the vehicles, and serves the already inserted requests.

6.4 Adaptive Large Neighbourhood Search

This section presents the ALNS heuristic used to improve the initial solution and re-optimise the solution in case of disruptions. Firstly, Section 6.4.1 provides an overview of the ALNS heuristic. Then, Section 6.4.2 introduces the implemented destroy operators, while Section 6.4.3 describes the repair operators. The adaptive weight adjustment is presented in Section 6.4.4, while the simulated annealing acceptance criterion is presented in Section 6.4.5.

6.4.1 Overview of ALNS

The ALNS heuristic is run for a given number of iterations. In each iteration, a destroy operator removes a certain number of requests, q, from the current route plan. Then, a repair operator tries to reinsert the removed requests back into the solution. The destroy and repair operators are chosen based on their previous performance in the algorithm. Hence, if a specific operator repeatedly results in new and better solutions, it is used more frequently. Whether to accept or reject the new solution is based on the simulated annealing acceptance criterion, explained in Section 6.4.5. The pseudocode of the ALNS heuristic is presented in Algorithm 2.

Algorithm 2 ALNS

```
1: Input: current solution, $x$
2: Input: set of infeasible requests, $I$
3: Initialize best known solution, $x^b := x$
4: Initialize destroy weights, $p^- := (1, ..., 1)$
5: Initialize repair weights, $p^+ := (1, ..., 1)$
6: while stopping criterion not met do
7:     Select destroy and repair operators $d \in \Omega^-$ and $r \in \Omega^+$ using $p^-$ and $p^+$
8:     Define candidate solution, $x^c := r(d(x, I), I)$
9:     if accept($x^c, x$) then
10:        $x := x^c$
11:    end if
12:    if $c(x^c) \leq c(x^b)$ then
13:        $x^b := x^c$
14:    end if
15:    Update $p^-$ and $p^+$
16: end while
17: output: solution $x^b$
```

The algorithm keeps track of three different solutions: x^b is the best solution observed during the search, x is the current solution, and x^c is a candidate solution obtained by applying the operators in a specific iteration. x^c may either be discarded or chosen as the new current solution, depending on the results of the accept function in line 9. Ω^- is the set of destroy operators, and Ω^+ is the set of repair operators. The function $d(*)$ corresponds to a chosen destroy operator, while $r(*)$ is a chosen repair operator. The weights of the destroy and repair operators are stored in the variables $p^- \in \mathbb{R}^{\Omega^-}$ and $p^+ \in \mathbb{R}^{\Omega^+}$, respectively. Initially, all operators have the same weight. $c(x)$ denotes the objective value of a solution x. The stopping criterion of the ALNS is simply that the algorithm stops after a specified number of iterations. The number of iterations is further inspected in Section 9.

6.4.2 Destroy Operators

The destroy operators remove a specified number of q requests from the current solution. The value of q is based on the destruction degree, D, specifying the percentage of the requests in the current solution to remove in each iteration.

Random Removal

The random removal operator selects q requests randomly and removes them from the route plan. The random removal might seem like an ineffective operator as it may remove favourable parts of the solution, however it contributes by providing diversification. Consequently, although it might not be a productive operator on its own, it works well in combination with the other destroy operators.

Worst Deviation Removal

The worst deviation removal operator selects the q requests with the highest deviation and removes them from the route plan. The idea behind the operator is to remove the parts of the solution that contribute to high costs. Hopefully, the repair operator is then able to insert the requests into a position that reduces the high deviation costs.

Shaw Removal

The Shaw removal operator selects a request, i, from the route plan and removes its $q-1$ most similar requests. If there are requests in the infeasible set, i is chosen randomly among these, and its $q-1$ most similar requests in the route plan are removed. The similarity of two requests, i and j, is defined using a relatedness measure $R(i,j)$. The lower the value of $R(i,j)$, the more related are the two requests. The relatedness measure consists of a travel time term and a time of service term. Thus, the relatedness measure is given by Equation (6.1).

$$R(i,j) = (T_{ij} + T_{i+n,j+n} + T_{i+n,j} + T_{i,j+n}) + (|t_i - t_j| + |t_{i+n} - t_{j+n}|) \tag{6.1}$$

where T_{ij} is the direct travel time between node i and j, and t_i is the time of service for node i. Here, node i represents the pick-up node for request i, while node $i+n$ represents the drop-off node. Consequently, $R(i,j)$ sums the travel time between the pick-up and drop-off nodes of the requests, as well as the difference in service time of the requests.

The general idea behind this operator is that if we remove requests that are somewhat similar, the requests can easily be exchanged when reinserting them into the solution, thereby creating a new and perhaps better solution. On the contrary, requests that are very different from each other might be hard to shuffle around, hence they might only be reinserted into their original position.

Travel Time Related Removal

The travel time related removal operator is similar to the Shaw removal, however, it does not consider the service time of the requests. Hence, it selects a request, i, from the route plan and removes its $q-1$ most similar requests in terms of travel time from both the pick-up and drop-off node. Again, if there are requests in the infeasible set, i is chosen randomly among these, and its q most similar requests in the route plan are removed. The relatedness measure used in this operator corresponds to the first term in Equation 6.1.

Service Time Related Removal

The service time related removal operator selects a request, i, from the route plan and removes its $q-1$ most similar requests in terms of service time of both pick-up and drop-off nodes. As with the Shaw removal, i is chosen from the infeasible set if there are requests there and its q most similar requests in the route plan are removed. The idea behind this operator is the same as for Shaw removal, however, the relatedness measure used is defined only by the second term in Equation 6.1.

6.4.3 Repair Operators

After the destroy operator has removed the requests from the route plan, the heuristic tries to update the service times of the remaining nodes to minimise the soft time window violation. The resulting route plan is defined as the destroyed route plan. The destroyed solution also includes a set of unassigned requests. This set consists of the requests removed from the solution by the destroy operator and the requests in the infeasible set.

The repair operators receive the destroyed solution and try to repair it by reinserting the removed requests into the route plan. As with the construction heuristic described in Section 6.3, the repair operators cannot remove already inserted nodes from the route plan. However, it can adjust their service times to accommodate the unassigned requests. Adjustments can be made as long as their service times are kept within their hard time windows.

Greedy Insertion

The greedy insertion operator is the same as the construction heuristic presented in Section 6.3. The operator sorts the set of unassigned requests according to the earliest pick-up time and inserts the requests into the route plan, one at a time. The operator first tries to insert the requests into already utilised vehicles. If no feasible insertion into already utilised vehicles can be found, it adds a new vehicle and inserts the request into its route. If all vehicles have been introduced and no feasible insertions are found, the request is added to the infeasible set.

Regret-k Insertion

The regret-k insertion operators try to improve the pure greedy insertion operator by incorporating some degree of look ahead when selecting which request to insert first. Let x_{ik} represent the insertion of request i with the kth lowest cost among all feasible insertions of request i. Hence, if c is the cost of inserting request i, we have $c_{i,x_{ik}} \leq c_{i,x_{ik'}}$ for $k \leq k'$. Instead of sorting the requests according to the earliest pick-up time, the regret-k heuristic sorts them by descending regret value. The regret value for a request, i, is calculated as in Equation 6.2.

$$\max \left\{ \sum_{j=1}^{k} (c_{i,x_{ij}} - c_{i,x_{ik}}) \right\} \tag{6.2}$$

In general, regret heuristics with larger k values will result in more look ahead, however, the computational time increases (Ropke and Pisinger, 2006). Consequently, the proposed ALNS heuristic implements two different regret-k operators: regret-2 and regret-3.

6.4.4 Adaptive Weight Adjustment

The weights, p^- and p^+, select the destroy and repair operators. Specifically, the algorithm calculates the probability ϕ_i^- of choosing the ith destroy operator as

$$\phi_i^- = \frac{p_i^-}{\sum_{k \in \Omega^-} p_k^-} \tag{6.3}$$

and similarly for the repair operators.

One of the key features of the ALNS is the dynamic updating of the weights based on the past performance of the operators. At the end of each iteration of the ALNS heuristic, a score ψ is given to the destroy and repair method used in the iteration. The score is calculated based on the following formula:

$$\psi = \begin{cases} \omega_1, & \text{if the new solution is a new global best} \\ \omega_2, & \text{if the new solution is better than the current one} \\ \omega_3, & \text{if the new solution is accepted} \\ 0, & \text{if the new solution is rejected} \end{cases} \tag{6.4}$$

After n_{iter} iterations, the weights of the operators are updated as follows:

$$p_i^- = \begin{cases} (1-\lambda)p_i^- + \lambda\frac{\Psi_i}{\theta_i}, & \text{if } \theta_i > 0 \\ p_i^-, & \text{if } \theta_i = 0 \end{cases} \tag{6.5}$$

$$p_i^+ = \begin{cases} (1-\lambda)p_i^+ + \lambda\frac{\Psi_i}{\theta_i}, & \text{if } \theta_i > 0 \\ p_i^+, & \text{if } \theta_i = 0 \end{cases} \tag{6.6}$$

where Ψ_i is the accumulated score of operator i, θ_i is the number of times operator i has been used, and λ is the reaction factor. The reaction factor controls how sensitive the weights are to changes in the performance of the operators. After updating the weights, Ψ_i and θ_i are reset to zero.

6.4.5 Acceptance Criterion

A simple acceptance criterion is only to accept better solutions than the current solution. However, using such a criterion might lead to getting trapped in a local optimum (Ropke and Pisinger, 2006). Consequently, the heuristic uses the acceptance criterion from simulated annealing, where occasionally, solutions that are worse than the current solution are accepted. Specifically, a worse solution is accepted with probability $exp(-\Delta/T)$, where $T > 0$ is the temperature and Δ is the difference in objective value. This corresponds to line 8-10 in the pseudocode in Algorithm 3.

Algorithm 3 Simulated Annealing Acceptance Criterion

```
1: Input: candidate objective, $c(x^c)$
2: Input: current objective, $c(x)$
3: Initialize $T = T_{start}$
4: Initialize accept := False
5: if $c(x^c) < c(x)$ then
6:     accept := True
7: else
8:     $\Delta = c(x^c) - c(x)$
9:     $r$ := random number, uniformly drawn from $[0, 1]$
10:    if $r < \exp(-\Delta/T)$ then
11:        accept := True;
12:    end if
13: end if
14: $T := T \cdot C$
15: output: accept
```

T starts out at a start temperature, T_{start}, and is decreased at every iteration according to the cooling rate, $0 < C < 1$. This corresponds to line 14 in Algorithm 3. To choose a proper value for the parameter T_{start}, the method used by Ropke and Pisinger (2006) is adopted. They calculate T_{start} by inspecting the initial solution. Specifically, the temperature is set such that a solution that is z percent worse than the current solution is accepted with probability 0.5. Then, the parameter to be set is z instead of T_{start}. The parameter z is called the start temperature control parameter.

6.5 Re-Optimisation Procedure

This section describes how the solution method handles incoming disruptions. As mentioned in Chapter 4, we consider four types of disruptions: new request, delay, cancellation, and no-show. All disruption types except for new requests trigger a disruption updater. The updater updates the current route plan to include the disruption, for instance, by removing a cancelled request.

Before generating the disruptions, any deviations from the requested pick-up times in the initial route plan are reset to zero. Consequently, when solving the re-optimisation problem, the heuristic minimises the deviation from the current route plan instead of the requested pick-up times. This ensures that the new route plan does not deviate too much from the current route plan, thus providing predictability for the users.

6.5.1 New Request

In case of a new request, the algorithm first tries to insert the request using the greedy insertion heuristic presented in Section 6.4.3. If the insertion heuristic fails to find a feasible insertion, the request is rejected. If accepted, it is inserted at its requested pick-up and drop-off times. Then, the ALNS is utilised to improve the new route plan.

Figure 6.3 shows the insertion of a new request into the route plan. As mentioned in Section 6.3, the service times of already inserted nodes can be adjusted to accommodate new requests, as long as it is within the hard time window of the request. In the figure, a new request with pick-up node P and drop-off node D is inserted into the route. The service times of P and D are 11:00 and 11:20, respectively. Node A is already inserted at 11:40. As the travel time between A and D is 25 minutes, node A's service time must be adjusted by 5 minutes to accommodate the request. The service time of 11:45 is within node A's hard time window, hence node D can be inserted at 11:20.

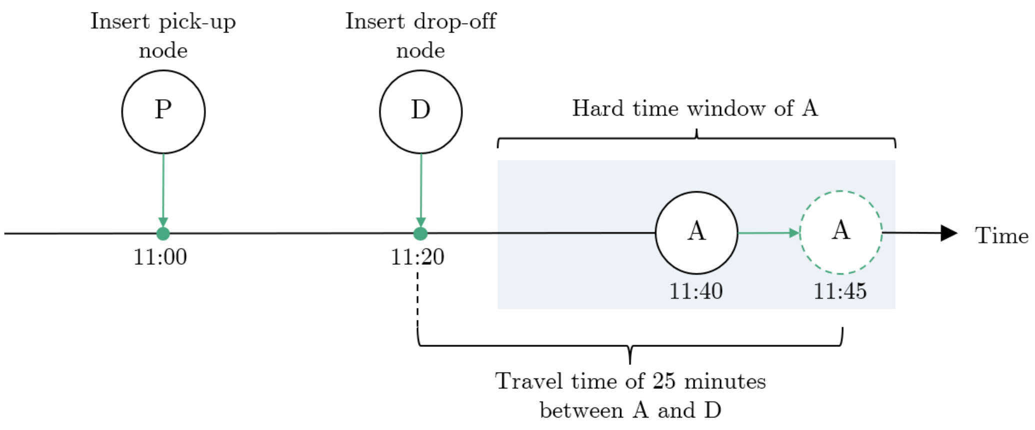


Figure 6.3: Adjustment of service times of already inserted nodes to accommodate new requests. A is already inserted into the route, while P and D is the pick-up and drop-off nodes of a new request.

6.5.2 Delay

A delay may lead to a shift in the time of service of all subsequent requests on the affected vehicle's route. Figure 6.4 illustrates how the route plan is updated in case of a delay. In the figure, node A is delayed by five minutes, from 11:00 to 11:05. The subsequent node, B, has a service time of 11:15 and the travel time from A to B is 15 minutes. Consequently, node B is also delayed by five minutes. The next node in the route plan, C, is not affected by the delay. This is because there is still enough time to travel between B and C after updating the service time of node B.

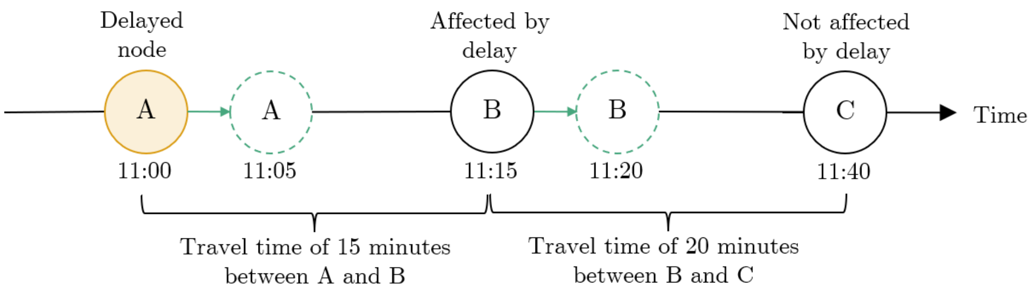


Figure 6.4: Adjustment of service times of nodes on a vehicle route impacted by a delay.

After the disruption updater has updated the affected nodes' service times, the ALNS is run to minimise the impact of the delay by reducing the deviation from the original time of service. If there are nodes with a deviation from the delay after the ALNS is run, the deviation for those nodes is reset to zero before the next disruption. As previously mentioned, this is to ensure that the heuristic tries to minimise deviation from the current plan in case of a new disruption.

6.5.3 Cancellation and No-show

In the case of cancellations and no-shows, the disruption updater removes the affected nodes from the route plan. If the removed nodes previously caused deviation costs for other nodes in the route, their service times may now be adjusted to minimise deviation. Figure 6.5 illustrates the cancellation of a request with the pick-up node, P, and drop-off node, D. Nodes P and D, which previously caused an adjustment of node A's service time, are thus removed. Hence, node A's service time can be updated to its original service time of 11:40.

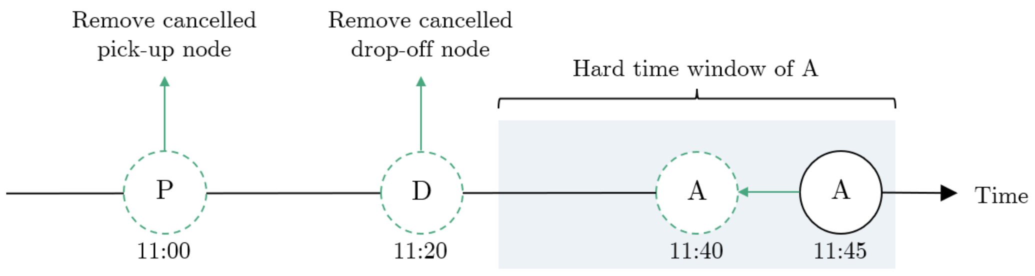


Figure 6.5: Re-adjustment of service times of the remaining nodes in case of a cancellation. Node A's service time is updated as a consequence of the removal of nodes P and D.

Chapter 7

Simulation Framework

This chapter presents the simulation framework used to generate disruptions during an operational day. Section 7.1 gives an overview of the simulator and explains how the disruption stack is generated. Then, Section 7.2 describes the generation of each disruption type in detail.

7.1 Overview of Simulator

In the simulator, an operational day is simulated with its corresponding disruptions. The length of an operational day is from 10:00 to 18:00. The simulator is event-based, hence it triggers the rest of the solution framework when an event, referred to as a disruption, occurs.

As mentioned in Chapter 4, there are four disruption types: new request, delay, cancellation, and no-show. Each disruption includes information about its disruption type and disruption time. The disruption time is defined as the moment the service is notified about the disruption. The corresponding disruption times are modelled as an inhomogeneous Poisson process for each disruption type. An inhomogeneous Poisson process is a Poisson process with time-dependent arrival rates (Gabbiani and Cox, 2010). This is used to capture the time-dependent nature of the arrival of disruptions. For instance, in the historical RAT request data, more requests tend to arrive at the beginning of the day compared with later in the day. As mentioned in Section 2.2, this is typical for transport of elderly. The arrival rates are based on the historical RAT request data and calculated for each operational hour.

Each Poisson process results in a list of disruption times for each disruption type throughout the operational day. The disruption times for each disruption type are combined into a disruption stack, which is sorted chronologically. Figure 7.1 shows an overview of the simulator. The simulator generates a disruption by removing the earliest disruption in the disruption stack. Then, it checks if the triggered disruption is valid. What makes a disruption invalid is explained in Section 7.2. If the disruption becomes invalid, the simulator continues to the next disruption in the disruption stack. If the disruption is valid, the associated actions for that disruption type are performed by the other parts of the solution framework, as presented in Section 6.2.

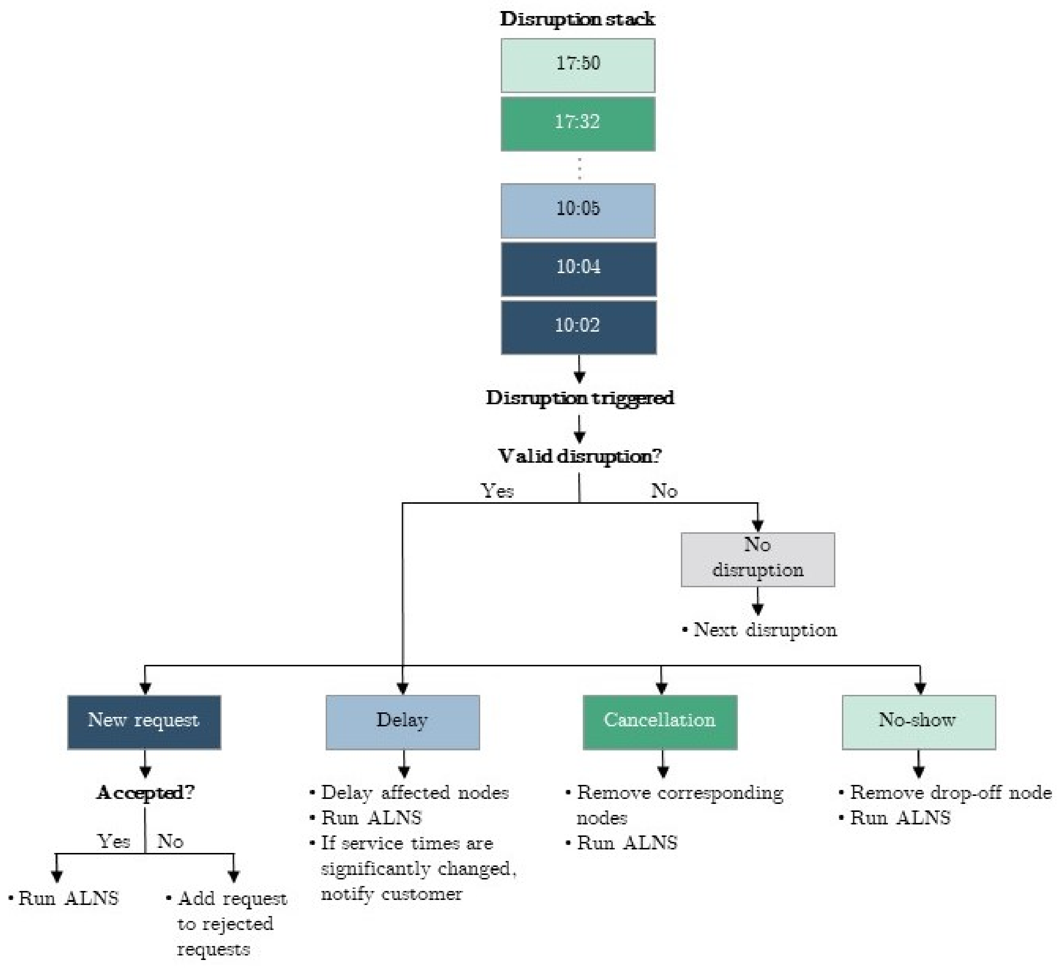


Figure 7.1: Overview of the simulator.

As mentioned in Section 6.2, the current route plan from the previous stage in the solution framework is used as input for the simulator. This is used to decide which requests are affected by the disruption. Then, the disruption is sent to either the greedy insertion heuristic or the disruption updater, depending on the disruption type. The greedy insertion heuristic/disruption updater then decides which actions to perform to respond to the disruption, and the updated route plan is sent to the ALNS. The simulator is run iteratively along with the rest of the solution framework until the disruption stack is emptied.

7.2 Disruption Types

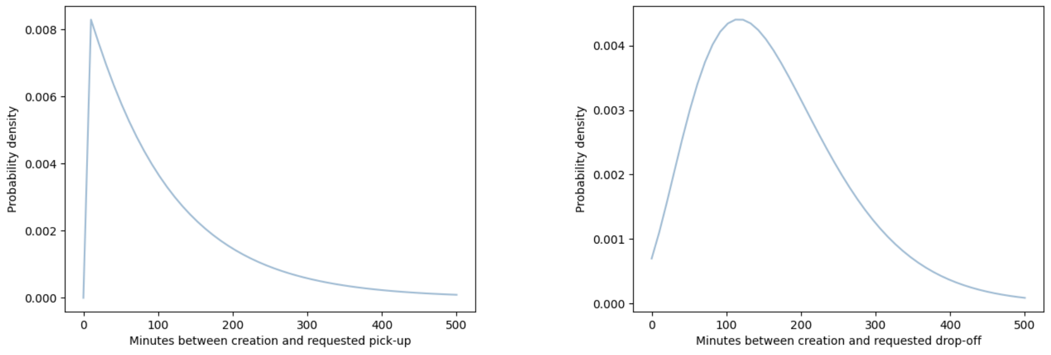
This section explains more in detail how each disruption type is generated. Additionally, what can cause a disruption type to become invalid is explained.

7.2.1 New Request

A new request occurs when a customer orders a trip. The disruption time defines the creation time of the request, namely the time the customer contacts RAT to order the trip. As previously mentioned, a customer can either request a pick-up or a drop-off time.

The probabilities of generating a request with the requested pick-up and drop-off time are 98.25% and 1.75%, respectively. A random number between zero and one is generated to decide whether the customer requests a pick-up or drop-off time. If the random number is below the drop-off percentage, the customer requests a drop-off time. Otherwise, a requested pick-up time is specified.

The time between the creation and requested times is drawn from a distribution to generate the requested pick-up or drop-off times. Two gamma distributions fitted to historical request data are used, one for requested pick-up time and one for requested drop-off time. These are shown in Figure 7.3. The gamma distributions for requested pick-up and drop-off times have means of 109 and 165 minutes, respectively, and they were chosen as they fitted well to the historical data. If the disruption time of a new request is after 17.45, it is considered invalid. Additionally, if the request has a requested pick-up time after 17.45 or a requested drop-off time after 18.00, it becomes invalid. This is because it is assumed that the service will not be able to serve these requests within operational hours.



(a) Requested pick-up time

(b) Requested drop-off time

Figure 7.3: Gamma distribution of minutes between creation and requested time.

A request also specifies the latitude and longitude of the origin and the destination, the number of wheelchair passengers, and the number of passengers who can utilise standard seating. This information is generated by drawing a random request from a historical request data set containing all requests ordered after 10:00 on the same day as their requested pick-up/drop-off time. The information of the historical request is then used for the new request.

7.2.2 Delay

For a delay, the disruption time is when the service becomes aware of the delay. We assume that a delay can be known both at a location and when the vehicle is on its way to a location. In addition, we only look at significant delays, here defined as delays over five minutes. To decide which location is the first to be affected by the delay, the location with the earliest planned pick-up/drop-off time after the intermediate disruption time is chosen. Figure 7.4 illustrates how the delayed location is chosen when a delay occurs. The location of the node with id 2.5, which is a drop-off location, has the earliest planned pick-up/drop-off time after the disruption time and is therefore chosen as the first delayed location.

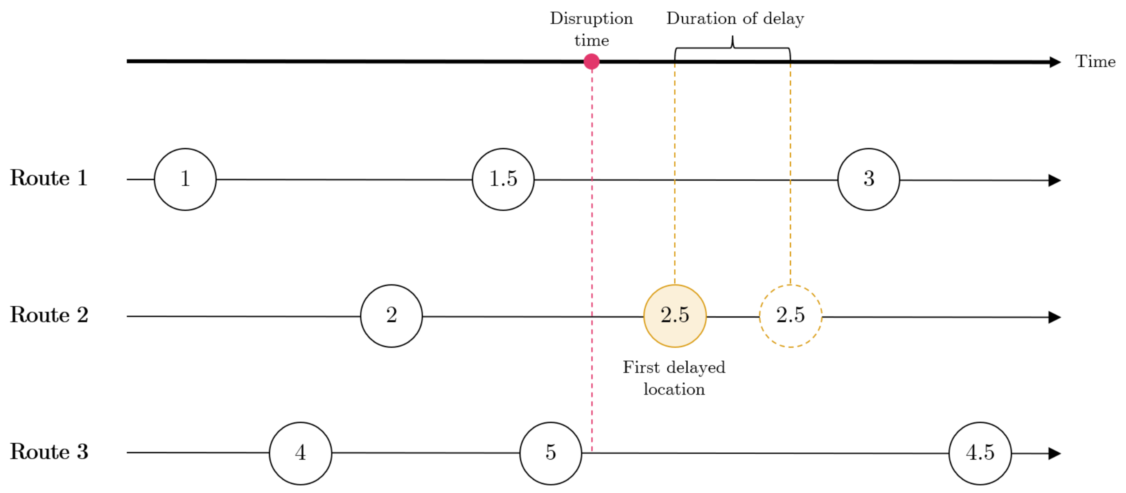


Figure 7.4: Illustration of a delay disruption.

If no pick-up or drop-off locations lie after the disruption time, the disruption becomes invalid. To decide the duration of the delay, a random number is drawn from a beta distribution fitted to historical data of delays, shown in Figure 7.5. The mean of the beta distribution is 12 minutes. Additionally, a node can be delayed several times by different delay disruptions.

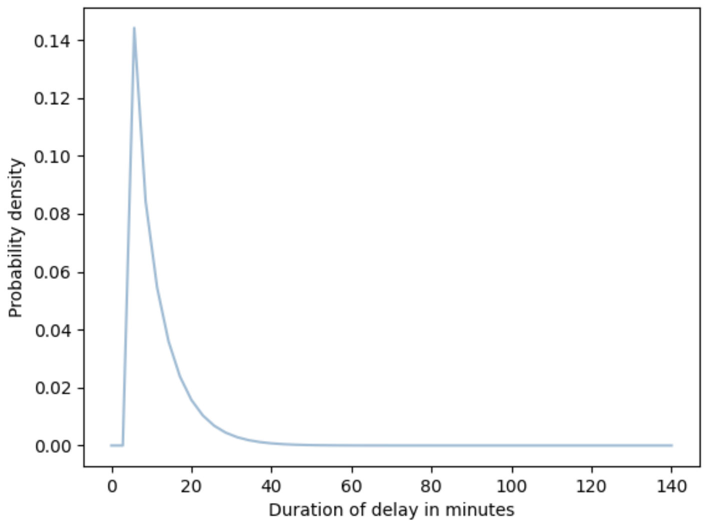


Figure 7.5: Beta distribution of duration of delay.

7.2.3 Cancellation

When a cancellation occurs, the disruption time equals when the customer contacts the service to cancel the ride. A customer can cancel a ride right up to the planned pick-up time. To decide how much in advance the cancellation occurs, a random number is drawn from the beta distribution shown in Figure 7.6. The beta distribution is fitted to historical request data and has a mean of 132 minutes. The earliest possible pick-up time of a cancelled request then becomes the disruption time plus the drawn random number, here defined as the simulated pick-up time of the cancelled request. If there is no request with a planned pick-up time precisely at the simulated pick-up time, the request with the earliest planned pick-up time after the simulated pick-up time is chosen as the cancelled request.

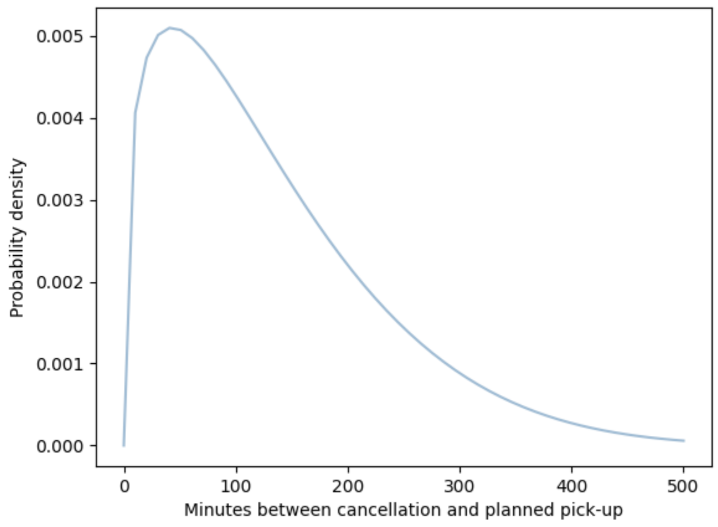


Figure 7.6: Beta distribution of minutes between cancellation and planned pick-up time.

In Figure 7.7, the pick-up location of the node with id 3, marked in yellow, is the earliest pick-up location after the simulated pick-up time. Hence, request 3 becomes the cancelled request. Further, as only requests that have not yet been served can be cancelled, only pick-up locations are considered when choosing the request to cancel. For example, the node with id 2.5 in the figure has the earliest planned service time after the simulated pick-up time, but it is not considered as it is a drop-off location. The disruption becomes invalid if no pick-up nodes lie after the simulated pick-up time.

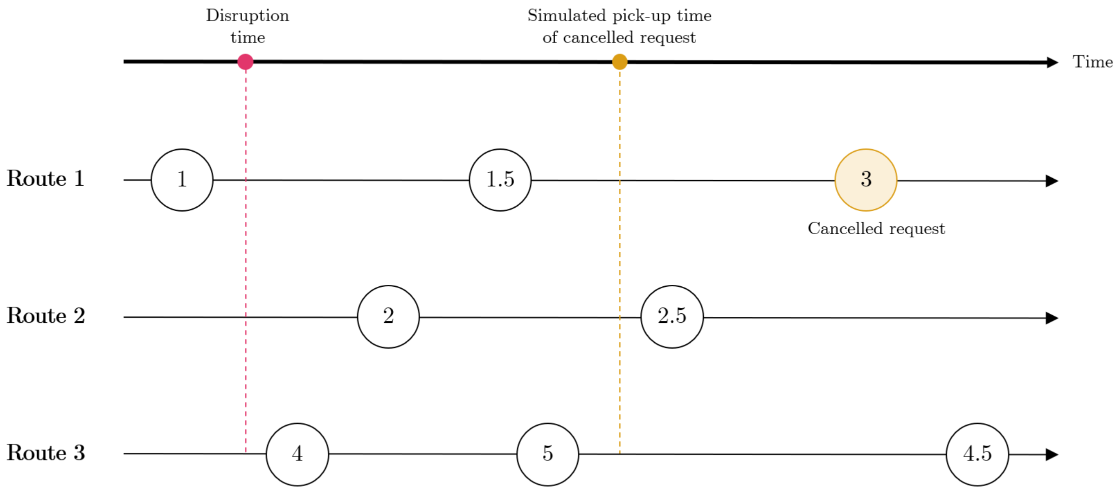


Figure 7.7: Illustration of a cancellation disruption.

7.2.4 No-show

A no-show occurs when a customer does not show up at its pick-up location. The disruption time generated by the Poisson process is only an initial disruption time, as the actual disruption time is when the vehicle reaches the pick-up location and the customer is not there. The pick-up location with the earliest planned pick-up time after the initial disruption time is chosen as the no-show request. For example, in Figure 7.8, the no-show request is the location of the node with id 3. The actual disruption time is then set to the location of node id 3's planned pick-up time.

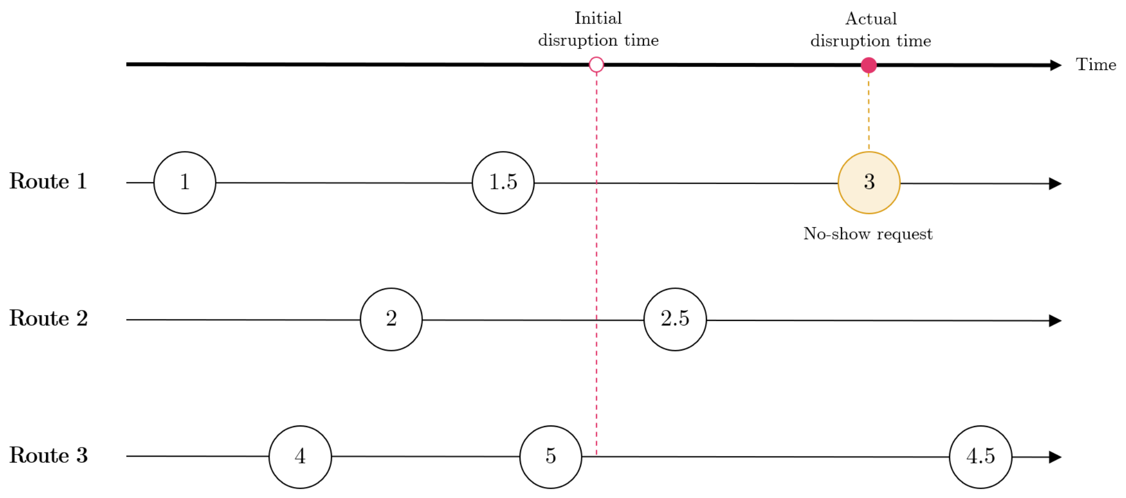


Figure 7.8: Illustration of a no-show disruption.

A no-show disruption can become invalid due to two different circumstances. The first one is if no pick-up locations lie after the initial disruption time. The second circumstance is if the actual disruption time lies after the disruption time of the next disruption in the disruption stack. This is because the next disruption may change the current route plan, which is the basis for selecting a specific request as the no-show request. For instance, if the next disruption is a cancellation, the same request may be chosen as a cancelled request.

Chapter 8

Implementation and Generation of Test Instances

This chapter presents the implementation and generation of test instances used in the computational study in Chapter 9. Section 8.1 describes the data set provided by RAT. This is used both to determine parameters and generate tuning and test instances. Further, Section 8.2 explains the calculation of some of the parameters. Lastly, Section 8.3 describes the generation of tuning and test instances.

8.1 Data Set

A real-life data set containing historical data about requests, delays, cancellations, and no-shows has been provided by RAT. This is used to calculate the historical arrival rates used by the simulator and generate test and tuning instances. The data consists of ride requests over the period May 2021 to January 2022 in the operational areas of RAT. Table 8.1 shows the most relevant columns in the data set. As a customer specifies either a requested pick-up or drop-off time, either *Requested Pickup Time* or *Requested Drop-off Time* has a NaN value, representing that the corresponding time has not been specified. For example, request 1 has a NaN value in the *Requested Pickup Time* column, meaning that the customer specified a requested drop-off time. Even though it is possible to specify a requested drop-off time, the data set only includes columns on pick-up time when it comes to original planned and actual pick-up times.

Table 8.1: Relevant columns from the data set along with three example requests.

Column name	**Request 1**	**Request 2**	**Request 3**
Request Creation Time	13/10/2021 11:33:24	11/10/2021 09:27:25	29/09/2021 13:51:00
Requested Pickup Time	NaN	14/10/2021 15:00:00	14/10/2021 13:15:00
Requested Dropoff Time	14/10/2021 11:15:00	NaN	NaN
Number of Passengers	1	1	2
Wheelchair	0	0	0
Origin Lat	59.9659	59.9218	59.9243
Origin Lng	10.7762	10.6875	10.6746
Destination Lat	59.9299	59.9312785	59.9450753
Destination Lng	10.6599	10.7045154	10.8021844
Original Planned Pickup Time	14/10/2021 10:55:00	14/10/2021 15:00:00	14/10/2021 13:16:05
Request Status	Completed	Cancel	No Show
Actual Pickup Time	14/10/2021 11:02:24	NaN	NaN
Cancellation Time	NaN	14/10/2021 14:22:02	NaN
No Show Time	NaN	NaN	14/10/2021 13:18:56

In Table 8.1, requests 2 and 3 both have requested pick-up times. However, request 2 has been cancelled, as seen from the *Request Status* and *Cancellation Time* columns. Request 3, on the other hand, resulted in a customer no-show. This can be deduced from the *Request Status* and *No Show Time* columns. A delay can be indicated by the difference in *Original Planned Pickup Time* and *Actual Pickup Time*. Request 1, for example, was delayed by approximately seven minutes.

8.2 Generation of Parameters

Additional preprocessing is needed to determine the value of some of the parameters used by the solution method. This section discusses how these parameters are calculated.

8.2.1 Pick-up Time

As mentioned in Chapter 6, the greedy construction heuristic and the greedy repair operator sort the requests to be inserted based on pick-up times. Hence, an artificial pick-up time must be calculated if a request has a requested drop-off time. This is done by using the formula in Equation 8.1.

$$\textit{Artificial pick-up time} = \textit{Requested drop-off time} - T_{ij} \tag{8.1}$$

where T_{ij} is the travel time between pick-up location i and drop-off location j. If the current day is a weekday and the requested drop-off time is between 15:00 and 17:00, the travel time is increased by a rush-hour factor of 1.5. Further, the duration of embarking and disembarking, S_i, is not included when calculating the artificial pick-up time, but included later in the solution method when the requests are inserted by the greedy insertion heuristic.

8.2.2 Travel Time Matrix

For simplification purposes, the direct distance between two locations is calculated instead of the road-travelling distance. The haversine formula, shown in Equation (8.2), is used to calculate the direct distance, d. In the formula, r is the Earth's radius, ϕ_1 and ϕ_2 are the latitudes of location 1 and 2 respectively, and λ_1 and λ_2 are the corresponding longitudes.

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{8.2}$$

The direct travel times between all locations are calculated to generate the travel time matrix. This is done by dividing the direct distance, d, by the average speed for buses within the centre of Oslo. The average speed is estimated at 25 km/h, but is here lowered to 20 km/h to compensate for the fact that the direct distance is used instead of the road-travelling distance (Helgheim and Rydland, 2012). If two requests both have pick-up times between 15:00 and 17:00, and if the current day is a weekday, rush-hour modelling is applied. This is done by increasing the direct travel times between the associated locations of those requests by a rush-hour factor of 1.5.

8.2.3 Rejection Cost

A penalty cost is added to the objective when a request is rejected. The penalty cost is calculated based on the initial problem and is kept the same for the remaining of the operational day. As it is preferable to keep the rejection rate low, the rejection cost is equal to an estimate of the highest possible cost of inserting a request. Using the introduced notation from Section 5.2, the rejection cost is calculated as:

$$C^R = 4 \cdot \alpha \cdot \max_{i,j \in N} T_{ij} + 2 \cdot \beta \cdot D^{max} \cdot \frac{|\mathcal{N}|}{|\mathcal{K}|} \tag{8.3}$$

Here, α and β are the objective weights for the travel time and soft time window violation, respectively. T_{ij} is the direct travel time between location i and j, while D^{max} is the maximum soft time window violation a node can have without violating the hard time window. Furthermore, $\mathcal{N}$ and $\mathcal{K}$ are the sets of requests and service vehicles, respectively. The first term is the cost of the maximum travel time between two nodes multiplied by four. The reason for multiplying it by four is that a maximum of four new arcs are generated when inserting a request: the arc to the pick-up node, from the pick-up node, to the drop-off node and from the drop-off node. The second term is the maximum soft time window violation costs that may occur as a result of inserting a request. As the insertion of a request worst case results in soft time window violations for all other requests in the same vehicle route, the cost is multiplied by the average number of requests per vehicle. As all requests consist of both a pick-up and a drop-off node, the term is multiplied by two.

8.3 Tuning and Test Instances

As previously mentioned, the RAT service starts the operational day with a pool of initial requests. These have been requested before 10:00 for that specific operational day. Hence, instances of initial requests are necessary to test the solution method. These instances are generated by filtering the data set from RAT on date and request creation time. Figure 8.1 shows an example of an instance of initial requests. Requests arriving after 10:00 are generated by the simulator, as presented in Chapter 7.

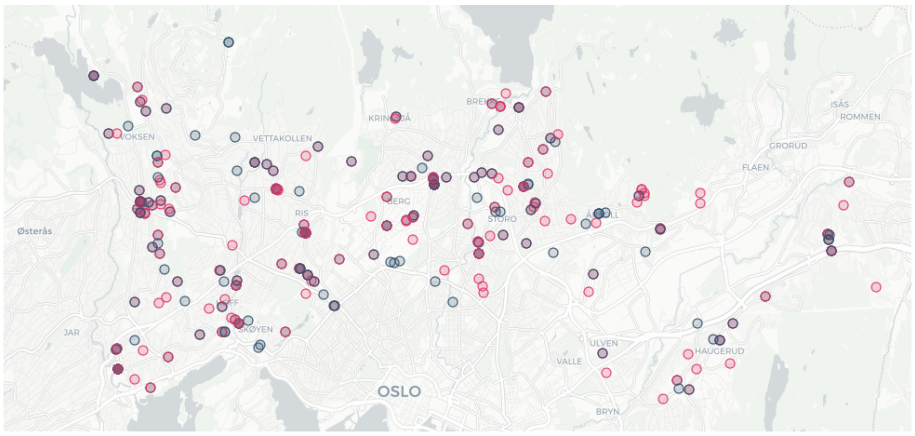


Figure 8.1: All initial requests for the random day of 14.10.2021. Pick-up locations are shown in pink, and drop-off locations are shown in blue.

The instances are divided into three categories based on the number of requests: small, medium, and large. The size intervals of each category are decided based on historical request data. Small instances have less than 110 initial requests, medium instances have between 110 and 170 initial requests, and large instances have more than 170 initial requests. The small instances represent low demand days, medium instances are normal demand days, and large instances are high demand days.

Table 8.2 shows the value of the input parameters used for the test and tuning instances. The fleet size, vehicle capacities, and service times are set according to the current values used by RAT. The value of F, which is the fraction over direct ride time that the ride time can exceed, is selected based on a survey executed by The Institute of Transport Economics (TØI). The survey revealed that the duration of 45% of all travels done by ordinary public transportation was more than twice as long as the direct travel time (Handagard, 2019). As RAT is a substitute for ordinary public transport, the value of F is set to 1, meaning that the travel time can maximum be up to twice the direct travel time. Lastly, the rush-hour factor, R_F, is set to 1.5. This is the expected increase in travel times during rush-hours in cities (Frøyland, 2009).

Table 8.2: Parameter values used for the test and tuning instances

Parameter	**Symbol**	**Value**
Fleet size	$\lvert\mathcal{K}\rvert$	16
Vehicle standard seat capacity	Q^S	15
Vehicle wheelchair seat capacity	Q^W	1
Service time duration standard seat	S^S	2
Service time duration wheelchair	S^W	5
Fraction of direct ride time	F	1
Rush-hour factor	R_F	1.5

Chapter 9

Computational Study

This chapter discusses the parameter tuning and performance of the proposed solution method and presents managerial insights. Table 9.1 shows the specifications of the computer used to obtain the results presented in this chapter.

Table 9.1: Specifications of computer, solver, and programming language.

Processor	2 x Intel Xeon Gold 5115 - 20 core
Operating System	CentOS Linux 7 (Core)
RAM	96 GB
Python version	3.9.6

The chapter is structured as follows: Section 9.1 introduces the parameter tuning of the solution method. Next, Section 9.2 examines the technical aspects of the proposed solution method. Finally, Section 9.3 presents the managerial insights obtained through analyses of different policies.

9.1 Parameter Tuning

This section presents the configuration of the objective weights and the heuristic parameters. As the re-optimisation problem inherits most of the features of the initial problem, the optimal parameters are presumed to be about the same for both problems. Consequently, the configuration of the objective weights and the heuristic parameters is only performed on the initial problem, i.e. when constructing the initial routes at the start of the day. The same configurations are then used in the re-optimisation problem. Each tuning instance was run ten times for each parameter configuration for each parameter.

9.1.1 Tuning Instances

Table 9.2 shows the characteristics of the nine instances used in the parameter tuning. All instances are retrieved from different days in the real-life data set provided by RAT and are generated as explained in Chapter 8.

Table 9.2: Characteristics of tuning instances

	Small			Medium			Large		
Instance	**S1**	**S2**	**S3**	**M1**	**M2**	**M3**	**L1**	**L2**	**L3**
Requests	64	71	85	130	139	158	192	194	213

9.1.2 Configuration of Objective Weights

In this section, the configuration of the objective weights is discussed. As presented in Chapter 4, the objective function consists of three parts: the total driving time of the service vehicles, the total soft time window violation, and a rejection cost. The driving time has objective weight α, while the soft time window violation is weighted by β. As discussed in Chapter 9.1.1, the rejection cost is calculated based on the two other objectives, hence it does not have a weight.

Table 9.3 presents the weight sets that have been tested. Weight set 1 is a balanced weight set. Here, the values of α and β are set such that the total soft time window violation and the total driving time are balanced. Weight set 2 is weighted towards the soft time window violation, while weight set 3 is weighted towards the driving time.

Table 9.3: Objective weight sets

Weight set	**1**	**2**	**3**
Driving time, α	1	2	1
Soft time window violation, β	100	100	200

The weight sets were tested by running the heuristic solution method with the instances presented in Section 9.1.1. The differences in the objective values across the different weight sets are minor for the small and medium tuning instances. Consequently, the configuration of objective weights is based on the results for the three large tuning instances presented in Table 9.4.

Table 9.4: Average objective term values in hours for each objective weight set

Instance	Weight Set	Weighted Objective	Normalised Driving Time	Normalised Soft Time Window Violation	Number of Rejections
L1	**1**	42.35	42.34	0.00	0.00
	2	83.97	41.95	0.00	0.00
	3	42.70	42.70	0.00	0.00
L2	**1**	116.06	43.89	0.05	0.11
	2	153.23	44.01	0.04	0.10
	3	1,317.77	44.82	0.05	0.89
L3	**1**	1,931.86	53.71	0.38	2.75
	2	1,825.42	53.92	0.36	2.50
	3	6,542.73	53.82	0.56	4.78

As seen in Table 9.4, the total driving time and the soft time window violation are relatively stable across all three instances. Consequently, the weight set is chosen mainly based on the rejection rate. Instance L1 has zero rejected requests and zero deviation across all weight sets, while the total driving time is slightly lower with weight set 2. For instance L3, weight set 2 yields the best values for all three objective terms. Weight set 1 yields the best objective term values for instance L2. However, the results were only slightly better than with weight set 2. Additionally, weight set 2 yields the overall lowest number of rejections. Consequently, weight set 2 is chosen as the final configuration and is thus the weight set used for the remaining computations.

9.1.3 Configuration of Heuristic Parameters

This section presents the results from the configuration of the different heuristic parameters. There are, in total, nine parameters to configure. Table 9.5 presents the different parameters, their description, and their initial value. The initial values were determined based on a number of test runs and were also used in the objective weights configuration. The parameters were configured one by one in the order they are listed. After tuning a parameter, its value was fixed to its best configuration before tuning the next parameter.

Table 9.5: Description of heuristic parameters

Parameter	**Description**	**Initial value**
I	Number of iterations in ALNS	100
Z	Start temperature control parameter	0.60
C	Cooling rate	0.96
D	Destruction degree	0.40
R	Reaction factor	0.70
V_1	Weight score for new global best solution	15
V_2	Weight score for new improving solution	10
V_3	Weight score for new non-improving, accepted solution	5
N_U	Batch size for updating weights	0.20

The average objective value and the average computational time of the heuristic were used to evaluate the parameters. The former is the most important when creating the initial solution, while the latter is central to the re-optimisation procedure. Consequently, both measures are equally important when considering the different configurations. For the small and medium instances, the differences in objective value and computational time were, in general, not very significant. Hence, the configuration is mainly based on the results from the large instances.

Table 9.6 summarises the tested and final values of all parameters. To determine the interval for the values to test, a number of test runs were executed. For most of the parameters, the initial interval was sufficient to find the final configuration. However, for some parameters, an extremity value yielded the best results. In that case, the interval for the values to test was extended. More detailed results can be found in Appendix B.

Table 9.6: Tested and final values of heuristic parameters

Parameter	**Tested values**	**Final value**
I	50, 100, 150	100
Z	0.20, 0.30, 0.40, 0.60, 0.80	0.60
C	0.92, 0.94, 0.96, 0.98	0.96
D	0.20, 0.30, 0.40, 0.50, 0.60	0.40
R	0.40, 0.50, 0.60, 0.70, 0.80	0.70
(V_1, V_2, V_3)	(15, 10, 5), (15, 5, 10), (10, 5, 1), (10, 1, 5)	(10, 5, 1)
N_U	0.10, 0.20, 0.30, 0.40, 0.50	0.30

9.2 Technical Analysis

This section examines the technical aspects of the proposed solution method. This includes an analysis of the impact of the different operators in the ALNS, the value of having adaptive weights, and the computational time for both the initial model and the different disruptions. All analyses are performed with the test instances presented in Section 9.2.1, with five simulations per instance. The heuristic parameters are fixed to the final values presented in Table 9.6.

9.2.1 Test Instances

Table 9.7 presents the characteristics of the three test instances used for the remaining of the computational study. These are introduced to increase validity of the results and not test the model on the same instances as the model was tuned. Instance 1 can be classified as a medium-sized instance, while instances 2 and 3 are large instances. As explained in Chapter 7, some of the simulated disruptions might turn out to be invalid. Consequently, the realised count for each disruption might be slightly lower.

Table 9.7: Characteristics of test instances

Instance	**1**	**2**	**3**
Initial requests	158	185	197
New requests	63	73	64
Delays	44	62	40
Cancellations	15	8	12
No-shows	8	6	4

9.2.2 Impact of Operators

This section analyses the impact of the destroy and repair operators used in the ALNS. Firstly, the ALNS was run with all operators, and the total number of new best solutions found by each operator was calculated. Based on these results, the worst and best-performing operators were removed from the heuristic to determine their impact on the final solution.

Table 9.8 presents the number of new best solutions found, the total count, and the percentage of best solutions by count for each destroy operator. For each instance, the best and worst-performing operators are marked green and red, respectively.

Table 9.8: Number of best solutions found by each destroy operator in relation to their count.

Instance	Operator	#Best Found	Total Count	Best/Count
1	Random	17.00	1,981.00	0.86%
	Worst	21.00	2,001.60	1.05%
	Shaw	26.60	2,115.00	1.26%
	Travel Time	13.80	1,968.60	0.70%
	Service Time	24.60	2,093.80	1.17%
2	Random	6.20	2,131.60	0.29%
	Worst	12.80	2,163.80	0.59%
	Shaw	10.20	2,187.40	0.47%
	Travel Time	9.40	2,161.60	0.43%
	Service Time	12.60	2,195.60	0.57%
3	Random	2.80	1,753.40	0.16%
	Worst	9.20	1,801.80	0.51%
	Shaw	8.80	1,874.00	0.47%
	Travel Time	4.20	1,780.80	0.24%
	Service Time	8.20	1,830.00	0.45%

As seen from Table 9.8, random removal and travel time removal have the lowest best solutions found to count ratios. While the random removal operator removes requests regardless of their location and service time, travel time related removal removes requests that are close in space. However, although close in space, the removed requests may be far away from each other in time. Therefore, both operators possibly end up removing requests that are very different from each other. As these can be hard to shuffle around, the requests may end up being reinserted into their original position, thus not generating new best solutions. On the contrary, Shaw removal and service time related removal both remove requests that are close in time. Consequently, the removed requests can easily be exchanged when reinserting them, thus creating new and possibly better solutions. As seen from the table, these are among the best-performing operators regarding the number of new best solutions found. Additionally, worst deviation removal is able to find a relatively high number of new best solutions.

Table 9.9 presents the same results for the repair operators. The regret-3 repair operator has the highest best to count ratio for all instances. Regret-2 finds a relatively high number of new best solutions for instance 1 and 2, however it has the lowest number of new best solutions found for instance 3. The greedy operator has the lowest average number of new best solutions found, but is still able to achieve competitive best-to-count ratios for instances 1 and 3. Additionally, as the regret-2 and regret-3 operators are more computationally complex, excluding the greedy operator would increase the computational time significantly. Consequently, we conclude that that all repair operators should be included.

Table 9.9: Number of best solutions found by each repair operator in relation to their count.

Instance	Operator	#Best Found	Total Count	Best/Count
1	Greedy	30.60	3,343.60	0.92%
	Regret-2	35.60	3,403.60	1.05%
	Regret-3	39.00	3,412.80	1.14%
2	Greedy	12.60	3,582.80	0.35%
	Regret-2	16.80	3,590.40	0.47%
	Regret-3	21.80	3,666.80	0.59%
3	Greedy	10.20	3,011.20	0.34%
	Regret-2	9.60	2,994.80	0.32%
	Regret-3	14.40	3,034.00	0.47%

To further assess the impact of the different operators, two variants of the ALNS excluding different operators are implemented. The operators used in each variant are shown in Table 9.10. A is the default variant of the ALNS, which includes all operators. Variant B excludes the worst-performing destroy operators, while variant C excludes the best-performing destroy operators. As all three repair operators yielded acceptable best to count ratios, none of these are excluded from any variants.

Table 9.10: ALNS variants. Included destroy/repair operators are indicated with x.

Operator	Variant A (Default)	Variant B (Removed Worst)	Variant C (Removed Best)
Random	x		x
Worst	x	x	
Shaw	x	x	
Travel Time	x		x
Service Time	x	x	
Greedy	x	x	x
Regret-2	x	x	x
Regret-3	x	x	x

The objective value and computational time for the different ALNS variants are shown in Table 9.11. Removing the worst-performing destroy operators, namely variant B, yields a slightly better objective value for instances 1 and 2 compared with variant A. For instance 3, the change is insignificant. The differences in computational time are also minor, with a small improvement for instances 1 and 3 and a small deterioration for instance 2. Removing the best-performing operators, namely variant C, yields a large decrease in computational time across all instances. Additionally, we see that the decrease in computational time is at the expense of a large increase in objective value. Therefore, the increase is most likely a result of more rejections. This results in fewer runs of the ALNS, hence the computational time is reduced. As variant B yields the overall best results, this variant is used for the remainder of the chapter.

Table 9.11: Objective and computational time in hours for different ALNS variants. Variant A represents the default model, while variant B has removed the worst operators and variant C has removed the best operators.

Instance	**ALNS variant**	**Objective**	**Change in objective**	**Computational time**	**Change in computational time**
1	A	12,394.63	-	3.16	-
	B	12,176.79	−1.76%	3.09	−2.31%
	C	14,206.20	14.62%	2.83	−10.50%
2	A	21,452.06	-	4.15	-
	B	21,232.38	−1.02%	4.23	1.87%
	C	25,204.03	17.49%	3.60	−13.13%
3	A	18,006.48	-	4.25	-
	B	18,026.87	0.11%	4.17	−1.91%
	C	22,384.17	24.31%	3.48	−18.10%

9.2.3 Value of Adaptivity

As described in Section 6.4, the ALNS is implemented with adaptive operator weights. Hence, all operators initially have the same probabilities of being chosen, which are then repeatedly updated based on the operators' past performance in the algorithm. However, the value of adaptivity was shown by Turkeš et al. (2021) to be limited for the majority of the reviewed implementations. Consequently, this section evaluates the importance of the adaptive layer in the implemented ALNS.

To assess the value of adaptivity, the three instances were solved with the ALNS both with and without adaptive weights. Results showed that the ALNS with adaptive weights, on average, yielded better objective values across all instances. Figure 9.1 shows the results from running five simulations of instance 3 with the original ALNS and the ALNS without adaptive weights. The data points are sorted by the final objective value of the simulation.

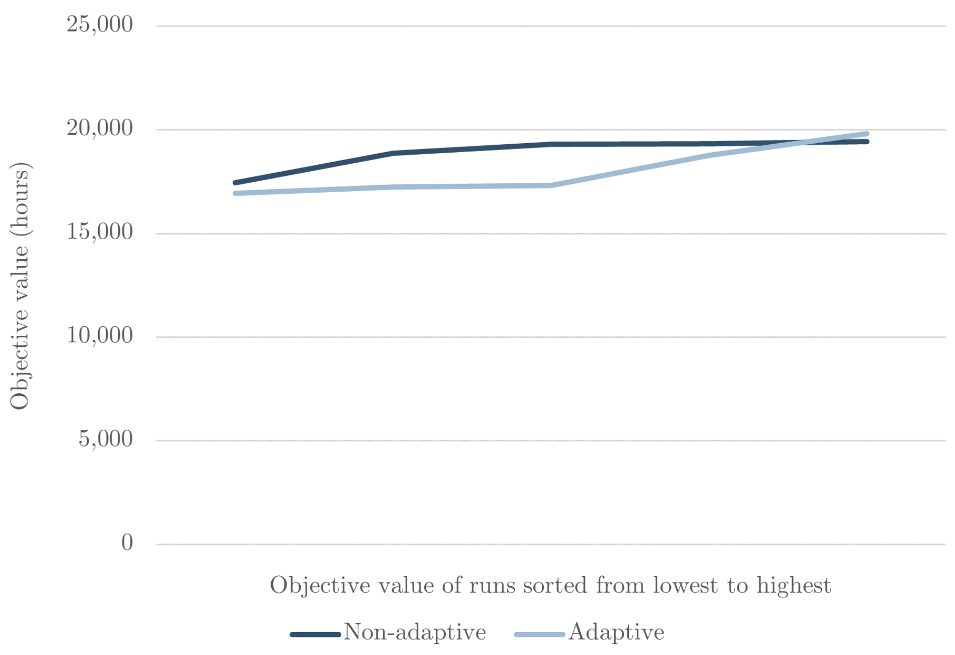


Figure 9.1: Results from running five simulations with the original ALNS and the ALNS without adaptive weights on instance 3.

As seen from the figure, the adaptive algorithm yields lower costs in four out of five simulations. Specifically, it achieves an average reduction in the objective value of 4.95% compared with the non-adaptive version. The adaptive version also yields the best results for instances 1 and 2, with an average cost reduction of 1.82% and 12.57%, respectively. Results for instances 1 and 2 can be found in Appendix C.1. It is evident that the value of adaptivity differs considerably between the three test instances, but in all cases, adaptivity does contribute with some value. Consequently, the implementation with adaptive weights is used for the remainder of this chapter.

9.2.4 Computational Time

This section presents the computational time of the solution method for both the initial problem and the different disruption types. While the initial requests might take some time to process, the disruptions must be handled in a timely manner. Table 9.12 presents the average computational time for the initial problem, the average response time to the customer, and the average computational time per disruption type.

Table 9.12: Computational time in seconds for initial problem and per disruption.

	Average	**Standard deviation**
Initial	332.81	50.14
Response time	61.32	12.89
New request	135.36	87.00
Delay	77.76	79.30
Cancellation	130.52	81.42
No-show	59.40	62.85

The most critical metric is the response time. This corresponds to the time the greedy insertion heuristic uses to process a new request. Consequently, it is the time it takes for the customer to receive an answer on whether its request is accepted or rejected. As seen from the table, the average response time is approximately one minute. This is slightly longer than preferred. However, with a faster computer and programming language, the response time will likely improve enough to be considered real-time.

The computational time for a delay and a no-show is relatively fast, while the computational time for cancellation and new request is slightly longer. A new request is the computationally most complex disruption to handle, as it involves both the greedy insertion heuristic and the ALNS. This is also reflected in the computational time, which, as shown in Table 9.12, is approximately 2.5 minutes.

For all disruptions, the corresponding standard deviation is relatively large. This is because the computational time is aggregated across all three test instances, which vary in the number of requests. More requests lead to a larger route plan to process, hence a higher computational time for both the greedy insertion heuristic and the ALNS. Additionally, the computational time for each disruption within one instance varies. This is because the route plan to process both shrinks and increases in size throughout the day. For instance, only the part of the current route plan, which lies after the disruption time, is considered when processing a delay. Consequently, a delay at the start of the day might take longer to process than a delay later in the day. However, as new requests cause the route plan to increase in size, the opposite may also be the case.

9.3 Managerial Insights

This section evaluates the proposed solution method and presents managerial insights relevant to RAT's decision-making. These insights are obtained by analysing the value of extending the pick-up interval, implementing an order deadline on new requests, changing the vehicle fleet size, and implementing a morning and afternoon shift. The analyses to perform have been determined in cooperation with RAT. All analyses are performed with the test instances presented in Section 9.2.1, with five simulations per test instance.

9.3.1 Key Performance Indicators

Three key performance indicators (KPIs) are introduced to evaluate the solutions of the different policies discussed. The primary goal of RAT is to serve as many requests as possible while maintaining a high level of service quality and having effective routes. Hence, the following KPIs are considered: rejection rate, cost per trip, and degree of ride-sharing. These are all KPIs currently used by RAT to evaluate their service. Additionally, the solutions are evaluated based on their objective value. Note that except for the rejection rate, the KPIs are not included in the objective function when solving the problem. Consequently, an improved solution may not necessarily yield an improvement of the KPIs.

Rejection Rate

A rejection rate KPI is introduced to compare the number of rejections across policies. It is calculated by dividing the number of rejections by the total number of requests, as seen in Equation 9.1. As each request results in two nodes, namely a pick-up and drop-off node, the requests are here represented by their respective pick-up nodes. Consequently, $\mathcal{P}$ is the set of pick-up locations for all received requests, while s_i describes whether request i is rejected.

$$\textit{Rejection rate} = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} s_i \tag{9.1}$$

Cost per Trip

The cost per trip KPI is introduced to measure the cost-effectiveness of the policies. As mentioned in Section 2.4, RAT defines the cost per trip as the hourly hiring price per bus divided by the total number of passengers transported per bus per hour. Equation 9.2 shows the calculation of the KPI. Here, C^B , denotes the hourly bus hiring price, while H denotes the total operating hours.

$$\textit{Cost per trip} = \frac{C^B}{\left(\frac{\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}} \left(q_{ik}^S + q_{ik}^W \right)}{|\mathcal{K}|} \right)} \cdot \frac{1}{H} \tag{9.2}$$

The total operating hours, H, is 8 hours as RAT operates from 10:00 to 18:00. Further, an estimated hourly price per bus, C^B, of 600 NOK is used. This is based on the cost per trip for comparable services in other parts of Norway, presented in the report by COWI (2018).

Level of Ride-sharing

A higher level of ride-sharing may be beneficial to achieving more cost-effective routes. The level of ride-sharing is calculated by dividing the number of passengers by the number of arcs in the route plan, as shown in Equation 9.3.

$$\textit{Level of ride-sharing} = \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}} \frac{(q_{ik}^{S} + q_{ik}^{W})}{\sum_{j \in \mathcal{N}} x_{ijk}} \tag{9.3}$$

9.3.2 Evaluation of the Solution Method

In this section, the proposed solution method introduced in Chapter 6 is compared with two different solution approaches. The first is a naive solver consisting only of the greedy insertion heuristic introduced in Section 6.3. Consequently, there is no improvement phase when constructing the initial route plan and no re-optimisation after delays, cancellations, and no-shows. In the case of new requests, the naive solver can slightly shift the service time of already inserted nodes to accommodate the new request, as long as the hard time windows are not violated. The aim of comparing the heuristic to the naive solver is to investigate the value of the ALNS.

The second solution approach combines the naive solver and the proposed heuristic, henceforth referred to as the semi-naive solver. Specifically, this approach utilises the ALNS to improve the initial route plan, but disruptions are solely handled by the naive solver. The semi-naive solver is meant to imitate RAT's current approach to creating initial routes and re-optimising the routes in case of disruptions. As mentioned in Chapter 2, RAT's replanning in case of disruptions is limited. Consequently, the comparison between the heuristic and the semi-naive solver indicates the value of re-optimising routes in case of disruptions.

Figure 9.2 shows the accumulated number of rejections for instance 1 for each of the three solvers. While the naive solver rejects a considerably higher number of requests, the difference between the semi-naive solver and the heuristic is minor. The figure shows that the naive solver starts with a much higher number of rejections, but manages to accommodate approximately the same number of on-demand requests. Hence, it is concluded that the difference in the number of rejected requests is caused mainly by the lack of the improvement phase when constructing the initial route plan. As most requests stem from the initial route plan, this is expected. However, as the heuristic is able to serve two more request than the semi-naive solver, the re-optimisation does contribute with some value when it comes to the number of served requests. The results for instances 2 and 3 are shown in Appendix D.1.

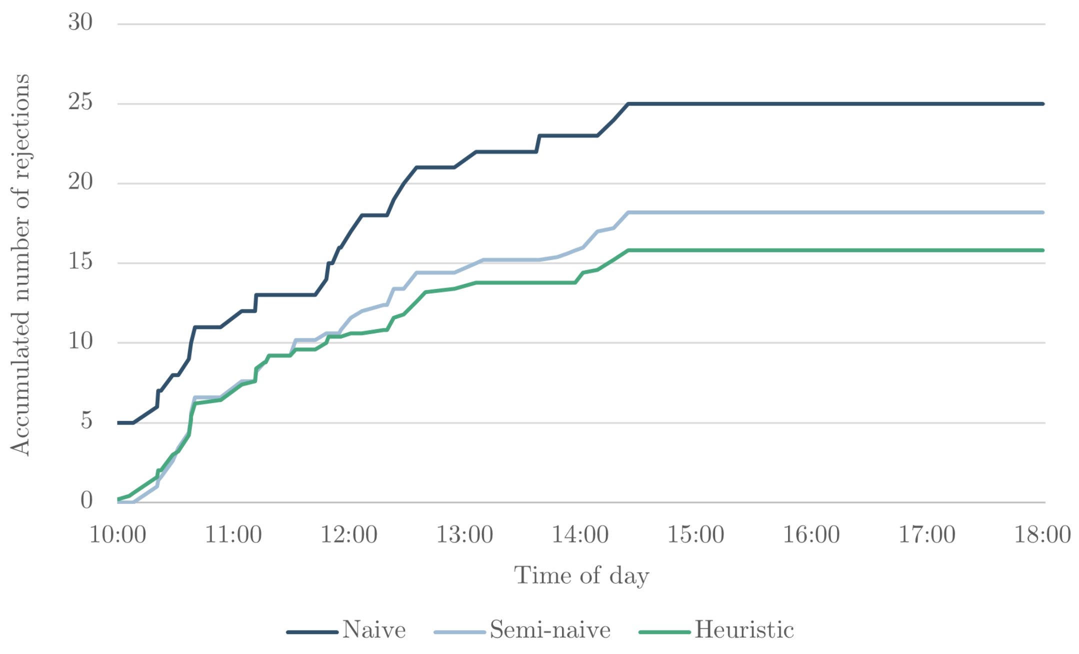


Figure 9.2: Average accumulated number of rejections from running five simulations with the naive, semi-naive, and heuristic solver approaches on instance 1.

Table 9.13 presents the different KPIs for the different solvers. As outlined in Section 9.3.1, both the cost per trip and the level of ride-sharing are calculated based on the number of transported passengers. Consequently, it is not surprising that the naive solver has the poorest performance across all KPIs. The heuristic solver has the lowest rejection rate across all instances, however, it does not always yield the best values for the cost per trip and level of ride-sharing. Specifically, the semi-naive solver yields a lower cost per trip and a higher level of ride-sharing for instances 1 and 2. As the requests have a varying number of passengers, the semi-naive solver has a higher number of passengers for these instances despite serving fewer requests. Consequently, the cost per trip and level of ride-sharing also improve.

Table 9.13: KPIs for the naive, semi-naive, and heuristic solver approaches.

Instance	**Solver**	**Rejection rate**	**Cost per trip**	**Ride-sharing**
1	Naive	11.68%	393.85	0.61
	Semi-naive	8.50%	308.93	0.76
	Heuristic	7.38%	311.69	0.74
2	Naive	12.85%	308.43	0.66
	Semi-naive	10.12%	260.69	0.76
	Heuristic	9.56%	263.37	0.74
3	Naive	11.42%	313.47	0.67
	Semi-naive	9.21%	266.48	0.77
	Heuristic	8.82%	254.98	0.80

To further evaluate the value of the re-optimisation, the impact of the different disruptions on the objective value is investigated. As we want to evaluate the re-optimisation procedure in isolation, the initial route plans must be comparable. Consequently, the pure naive solver is not included in the analysis. Table 9.14 shows the average change in the sum of the soft time window violation and travel time for each disruption type.

Table 9.14: Average change in the sum of the soft time window violation and travel time by disruption type for semi-naive and heuristic solver.

Instance	**Solver**	**Accepted request**	**Delay**	**Cancellation**	**No-show**
1	Semi-naive	0.96%	10.63%	−0.47%	−0.69%
	Heuristic	1.27%	9.51%	−0.27%	−0.04%
2	Semi-naive	1.50%	8.63%	−0.58%	−0.64%
	Heuristic	1.87%	7.64%	−0.09%	−0.10%
3	Semi-naive	2.75%	7.53%	−0.39%	−0.56%
	Heuristic	2.21%	8.96%	−0.17%	−0.02%

Both new requests and delays are likely to cause an increase in the value of both objective terms. While the semi-naive solver does nothing to reduce this impact, the heuristic solver utilises the ALNS to find better scheduling of the requests. The results presented in Table 9.14 indicate that for delays, the re-optimisation does provide some value. With a lower rejection rate, there are potentially more requests affected by the delay. Despite this, the heuristic solver is, on average, able to achieve a slightly lower increase in objective value than the semi-naive solver. On average, the semi-naive solver yields the lowest increase for new requests. Again, this might be because the heuristic solver is able to serve more requests. Consequently, the route plan is more crowded, resulting in higher soft time window violations when inserting the new request.

On the contrary, cancellations and no-shows are likely to reduce the value of both objective terms. However, as seen from Table 9.14, the reduction is negligible for both solvers. Consequently, the value of re-optimising in the case of such events is limited.

In general, the difference in performance between the semi-naive and the heuristic solver is relatively small. This may indicate that the problem is tightly constrained, leaving less room for improvement when re-optimising the solution. To further confirm this hypothesis, additional testing was performed. This included obtaining a lower bound on the rejection rate by implementing a static-deterministic version of the problem, and evaluating different objective weight sets on the re-optimisation problem to see how this affected the impact of the disruptions. The ALNS yielded a much lower rejection rate on the static-deterministic problem. This is likely because the solution space is larger compared with the dynamic problem, which is tightly constrained due to less time for planning. Further, the changes in objective value by varying the objective weights was insignificant. Hence, the ALNS seems to find similar solutions regardless of the tuning of the weights. Both results further strengthens the original hypothesis that the room for improvement when re-optimising is limited. Still, the ALNS is able to achieve an overall lower rejection rate and slightly lower costs in case of delays compared to the semi-naive solver. As the rejection rate is the most important objective for RAT, it can be concluded that the overall performance of the heuristic is slightly better than the semi-naive solver.

9.3.3 Value of Adjusting Requested Pick-Up Time

RAT's main objective is to serve as many requests as possible. Consequently, a request is only rejected if there is no feasible insertion of it in the current route plan. However, by slightly adjusting the pick-up time of a rejected request, RAT might still be able to serve it. As of today, RAT may suggest an alternative pick-up time for the customer if the service cannot serve it at the requested time. This is a manual process done in dialogue with the customer. A more efficient process may be achieved by including such suggestions as a part of the solution method. Consequently, this section examines the impact of allowing such adjustments of the pick-up time in the heuristic. We implement two different adjustments of the pick-up time, $+/-15$ minutes and $+/-30$ minutes. In the latter case, the algorithm always tries to first insert the rejected request at the +/- 15 minutes pick-up times.

Figure 9.3 shows the accumulated number of rejections throughout an operational day for different adjustments of pick-up times for instance 3. The presented results show that by adjusting the pick-up time of a rejected request, the service is able to serve more requests. Further, while the difference in the accumulated number of rejections between the default and the $+/-15$ minutes adjustment is significant, the difference between the $+/-15$ and $+/-30$ minutes is less substantial. This indicates that implementing a $+/-15$ minutes adjustment of the pick-up time would yield the highest marginal benefit for the service. The results for instances 1 and 2 show similar trends, and can be found in Appendix D.2.

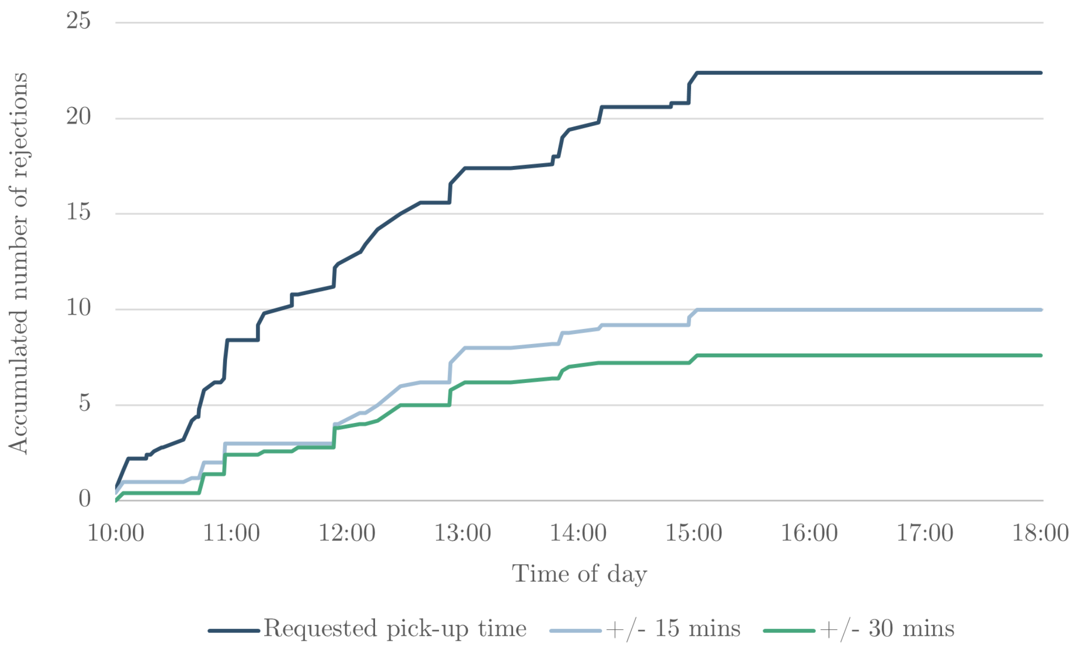


Figure 9.3: Average accumulated number of rejections from running five simulations with different pick-up time adjustments on instance 3.

Table 9.16 contains the rejection rate, cost per trip, and level of ride-sharing KPIs for each adjustment of the pick-up time. As discussed above, the rejection rate is reduced significantly by adjusting the pick-up time of a rejected request. For instance 3, the rejection rate is reduced from 8.82% to 3.94% by implementing the $+/-15$ minutes adjustment and further reduced to 2.99% by introducing the $+/-30$ minutes adjustment.

As a result of serving more requests, the cost per trip is reduced, and the level of ride-sharing increased. Even though the $+/-30$ minutes adjustments results in the best KPI values, it may not be desirable for the customer to adjust their pick-up time significantly. As the $+/-15$ adjustments yields the highest marginal benefit, it suggests that this is the most advantageous for RAT to implement.

Table 9.15: KPIs for different pick-up time adjustments.

Instance	**Adjustment of pick-up time**	**Rejection rate**	**Cost per trip**	**Ride-sharing**
1	No adjustment (default)	7.38%	311.69	0.74
	+/- 15 mins	3.46%	299.07	0.74
	+/- 30 mins	2.34%	287.64	0.76
2	No adjustment (default)	9.56%	263.37	0.74
	+/- 15 mins	5.62%	240.75	0.79
	+/- 30 mins	4.34%	233.29	0.80
3	No adjustment (default)	8.82%	254.98	0.80
	+/- 15 mins	3.94%	241.36	0.80
	+/- 30 mins	2.99%	236.31	0.83

9.3.4 Value of Implementing Order Deadline for New Requests

This section examines the value of implementing a one-hour deadline for new requests. Currently, RAT does not have a deadline for how much in advance a customer must make a request. However, as seen from Figure 9.4, the majority of rejected new requests are ordered less than an hour before the requested pick-up time. This holds for all instances, as seen in Appendix D.3. By implementing a request deadline, the service is likely to have more flexibility and possibilities for creating effective route plans. Hence, a deadline may enable RAT to serve more of the incoming requests. To implement this, the creation time of a new request ordered less than one hour before the requested pick-up time is changed to be the requested pick-up time minus one hour. If the new creation time is before 10:00, the request is added to the initial requests known at the beginning of the operational day.

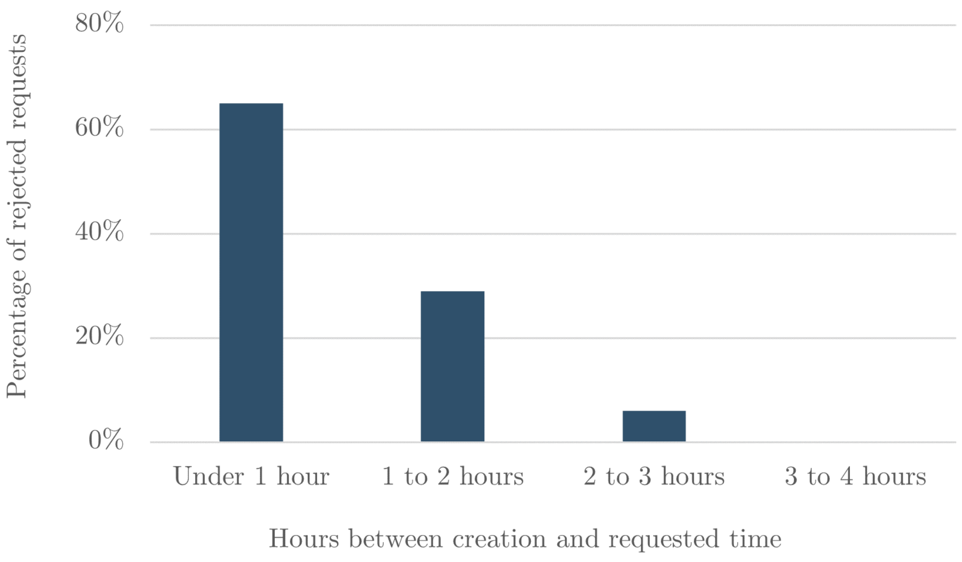


Figure 9.4: Percentage of rejected requests per hour between creation and requested pick-up time from five simulations on instance 3.

Figure 9.5 shows the accumulated number of rejections with and without a one-hour deadline for instance 3. The one-hour deadline policy significantly reduces the number of accumulated rejections, indicating that it would be beneficial to implement a one-hour deadline regarding the rejection rate. Similar trends are observed for instances 1 and 2, which are shown in Appendix D.4.

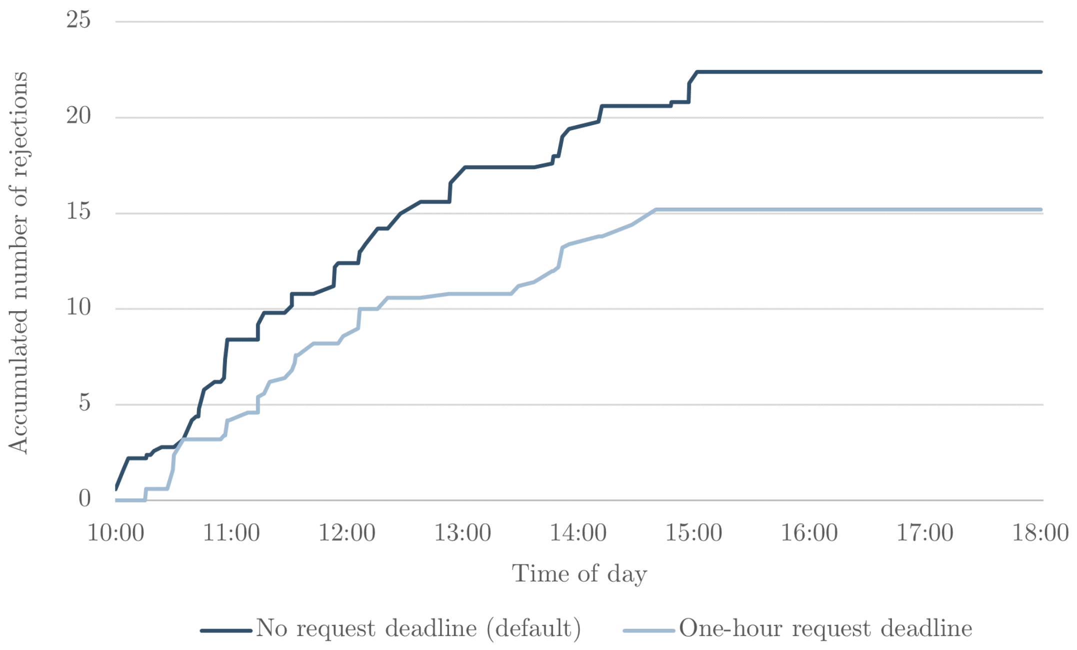


Figure 9.5: Average accumulated number of rejections from running five simulations with and without new request deadline on instance 3.

Table 9.16 shows the KPIs for all instances with and without the request deadline. A one-hour deadline results in a reduced rejection rate across all instances. The most significant reduction occurs for instances 1 and 3, while for instance 2, the reduction is smaller. This is likely because the set of rejected new requests in instance 2 have few feasible insertions, regardless of how much in advance they become known.

Table 9.16: KPIs for different deadline policies.

Instance	Request policy	Rejection rate	Cost per trip	Ride-sharing
1	No deadline (default)	7.38%	311.69	0.74
	One-hour deadline	4.21%	281.94	0.79
2	No deadline (default)	9.56%	263.37	0.74
	One-hour deadline	8.59%	260.87	0.74
3	No deadline (default)	8.82%	254.98	0.80
	One-hour deadline	5.98%	248.38	0.78

Further, Table 9.16 shows that the deadline reduces the cost per trip across all instances. This is most likely a result of the increase in the number of served requests. On the other hand, the level of ride-sharing is not significantly affected by the one-hour deadline. This may be because the number of passengers and the number of traversed arcs vary across the policies. Even though the policy might create more effective route plans to serve more requests, the number of ride sharing passengers does not necessarily increase.

As both the rejection rate and cost per trip are reduced by implementing a one-hour deadline, the analysis indicates that a deadline could benefit RAT. However, RAT would lose its on-demand offer, thus making the service less flexible. From a customer point-of-view, this is not desirable. Hence, it may be more favourable only to encourage the customers to order at least one hour in advance, but still allow for on-demand requests.

9.3.5 Service Vehicle Management

In this section, we investigate how different policies related to the vehicle fleet impact the KPIs presented in Section 9.3.1. As mentioned, RAT pays an hourly hiring price for each bus, hence their costs are greatly affected by the number of utilised vehicles. Consequently, the effects of varying the vehicle fleet size are evaluated. Further, as the number of requests decreases throughout the day, the value of having a different number of vehicles in the morning than in the afternoon is examined.

Analysis of Vehicle Fleet Size

RAT currently deploys a fleet of 16 vehicles. This section investigates the impact of reducing or increasing the fleet size. Figure 9.6 shows the accumulated number of rejections for instance 2 with a varying number of vehicles. As expected, an increase in fleet size leads to fewer rejections, while a reduction in fleet size leads to more rejections. Additionally, we observe that the marginal benefit is the highest when going from 17 to 18 vehicles. In comparison, there is not much to gain by increasing the fleet size by one extra vehicle. However, as seen in Appendix D.5, this varies between the instances. An explanation for this is that as the model is purely myopic. Hence, the marginal increase in total number of requests served might not be highest when increasing the fleet size by one as it is dependant on the specific requests received and the order they are received throughout the day. However, in general, more vehicles always leads to fewer rejections, hence the fleet size must be evaluated with the other KPIs in mind.

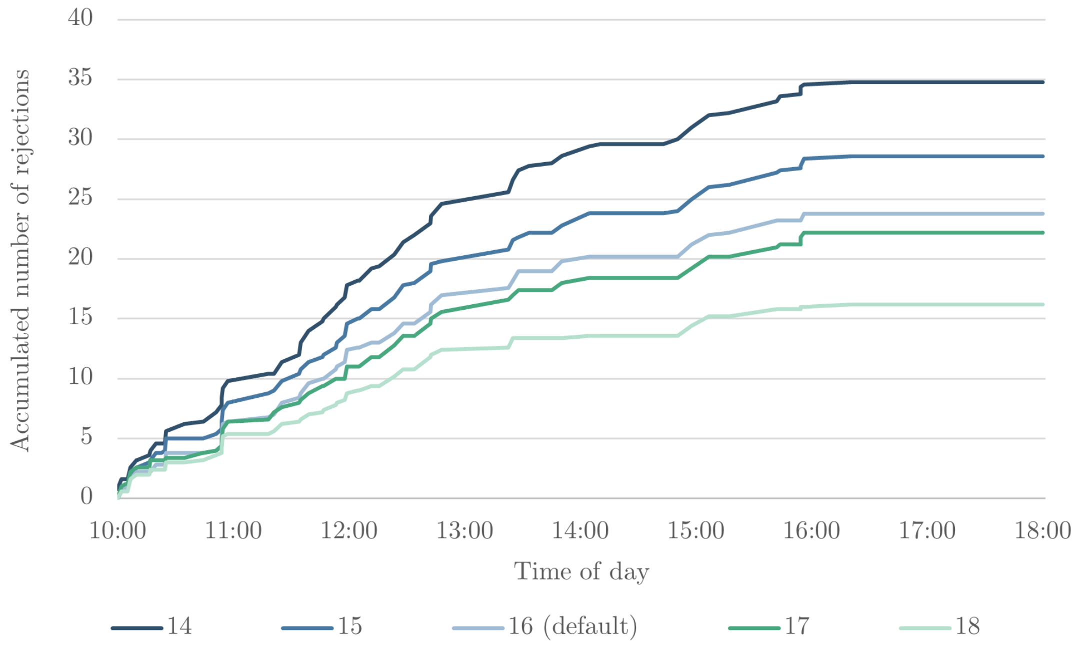


Figure 9.6: Average accumulated number of rejections from running five simulations with different fleet sizes on instance 2.

Table 9.17 presents the resulting KPIs for different fleet sizes for all three test instances. While the rejection rate generally decreases with the number of vehicles, the cost per trip increases. This indicates that the increase in the number of served requests does not compensate for the extra costs of renting more vehicles. Consequently, there exists a trade-off between the rejection rate and the cost per trip. Furthermore, it can be seen that for instance 3 the rejection rate is not strictly decreasing as the fleet size increases. This is somewhat unexpected, but may be a result of the timing of the utilisation of the vehicles, as well as the random characteristics of the ALNS.

Table 9.17: KPIs for different fleet sizes.

Instance	Fleet size	Rejection rate	Cost per trip	Ride-sharing
1	14	11.87%	292.68	0.73
	15	8.88%	315.24	0.71
	16 (default)	7.38%	311.69	0.74
	17	5.33%	310.74	0.77
	18	4.58%	344.50	0.72
2	14	13.98%	226.26	0.81
	15	11.49%	247.76	0.76
	16 (default)	9.56%	263.37	0.74
	17	8.92%	268.60	0.77
	18	6.51%	274.63	0.78
3	14	11.89%	232.04	0.81
	15	9.21%	240.00	0.82
	16 (default)	8.82%	254.98	0.80
	17	9.13%	275.49	0.79
	18	6.93%	298.34	0.75

The level of ride-sharing does not exhibit the same correlation with the number of vehicles. If a reduction in fleet size leads to a high increase in the number of rejections, the level of ride-sharing may decline. Equivalently, an increase in fleet size may lead to fewer rejections and thus an increase in ride-sharing. Hence, there is no apparent trend in which fleet size yields the highest level of ride-sharing.

Value of Having Morning and Afternoon Shifts

This section examines the impact of having morning and afternoon shifts with different vehicle fleet sizes for each shift. Analyses of the historical request data show that over 60% of the requests have requested pick-up/drop-off time before 14:00. Hence, the need for service vehicles may be reduced after 14:00. Consequently, it can be more cost-effective to introduce morning and afternoon shifts where the vehicle fleet size is reduced after 14:00. Additionally, the effect of increasing the vehicle fleet size before 14:00 is discussed.

Table 9.18 shows the rejection rate, cost per trip and level of ride-sharing for different shift configurations. The results show that the lowest rejection rate across all instances is obtained by having 18 vehicles in the morning shift and 14 in the afternoon. This shift configuration yields the same total rental costs as having 16 vehicles for the whole day, but as the rejection rate is lowered, the cost per trip on average decreases compared with the default fleet size. A further decrease in the cost per trip is obtained by having 18 vehicles in the morning and 12 in the afternoon. This results in a slight increase in rejection rate for instance 1 and 3, but the increase is minor. For instance 2, the rejection rate is reduced compared with the default fleet size. Reducing the number of vehicles in the afternoon shift further increases the rejection rate across all instances.

Table 9.18: KPIs for different fleet size shifts. The default fleet shift is given in blue, while proposed solutions to fleet size shifts are given in green.

Instance	**Morning fleet size**	**Afternoon fleet size**	**Rejection rate**	**Cost per trip**	**Ride-sharing**
1	16 all day (default)		7.38%	311.69	0.74
	16	14	9.53%	299.50	0.74
	18	14	6.45%	321.61	0.70
	18	12	8.22%	315.24	0.69
	18	10	10.28%	279.77	0.76
2	16 all day (default)		9.56%	263.37	0.74
	16	14	12.69%	243.24	0.79
	18	14	8.19%	252.47	0.76
	18	12	8.51%	238.41	0.77
	18	10	13.41%	233.50	0.77
3	16 all day (default)		8.82%	254.98	0.80
	16	14	10.16%	249.83	0.78
	18	14	7.72%	251.97	0.80
	18	12	9.29%	250.52	0.78
	18	10	10.55%	230.77	0.80

The results indicate that it might be beneficial for RAT to implement different fleet sizes in the morning and the afternoon. By having more vehicles in the morning and fewer in the afternoon, they are able to serve more requests without increasing their costs. Specifically, an increase of two vehicles in the morning and a decrease of two or four vehicles in the afternoon yields satisfactory results across all instances. To determine the exact shift configurations, the rejection rate has to be evaluated against the cost per trip. Also, an important prerequisite for such an implementation is that RAT can negotiate such a solution.

Chapter 10

Concluding Remarks

The purpose of this thesis is to understand and solve the dial-a-ride problem (DARP) faced by Ruter Aldersvennlig Transport (RAT), thereby providing them with managerial insights relevant to their decision-making. Consequently, an understanding of the problem, as well as possible areas of improvement, is needed. To obtain this, a thorough background study and literature review were conducted. The review revealed that existing literature on the disruptive DARP is limited, focusing mainly on the arrival of new requests. The range of disruption types considered should be expanded to more accurately model the real-world problem. This thesis considers disruptions in the form of new requests, delays, cancellations, and no-shows. Hence, it contributes to existing literature and creates a foundation for further research on disruption management in the DARP.

In order to perform the relevant analyses, an effective solution method is implemented. The proposed solution method combines a greedy insertion heuristic with an adaptive large neighbourhood search (ALNS). Additionally, a realistic simulation framework has been developed to simulate the occurrence of disruptions. Historical request data provided by RAT has been used to generate test instances and develop the proposed simulation framework to best mirror the real-life problem. The solution method and simulation framework are used together to obtain relevant insights on different route planning policies.

Through a technical analysis, it has become apparent that the proposed solution method solves all instances within a reasonable time and successfully handles disruptions in a timely manner. The latter is especially important in the case of new requests, as the customer expects a quick reply on whether its request is accepted. Results show that the response time for a new request is about one minute, which is considered reasonable considering the technical specifications of the test environment.

Further, the performance of the proposed solution method is compared to two other solution approaches, namely, a naive and semi-naive solver. As the main objective of RAT is to serve as many requests as possible, the performance is mainly evaluated based on the rejection rate. The heuristic achieves an average rejection rate reduction of 3.4% compared with the naive solver and an average reduction of 1.3% compared with the semi-naive. Additionally, results indicate that the proposed solution method better mitigates the effect of delays than the semi-naive solver. Hence, it can be concluded that the proposed solution method is applicable for solving problems related to disruptive DARPs.

As mentioned, the main purpose of this thesis is to provide RAT with managerial insights relevant to their decision-making. These have been obtained through analyses of different policies. First, the value of slightly adjusting the pick-up time of rejected requests was investigated. Results indicate that it is advantageous to implement an extended pick-up interval of $+/-15$ minutes, yielding an average reduction in the rejection rate of 4.2%. Further, the implementation of an order deadline for new requests was examined. Results show that the reduction in rejection rate is 2.3% compared with having no deadline. Both policies result in more cost-effective routes, with reductions in the cost per trip and increases in ride-sharing.

Policies related to the vehicle fleet have also been investigated, specifically the size of the fleet. The cost per trip decreases significantly by reducing the number of vehicles, but this is at the expense of the number of requests served. Further, the effect of varying the vehicle fleet size throughout the operational day was examined. Results show that by increasing the vehicle fleet size in the morning and decreasing it in the afternoon, the cost per trip can be significantly reduced without compromising the number of requests served. The policies related to the fleet size require more long term planning than the other proposed policies. However, the analysis reveals that this may be worth considering.

With the upcoming age wave and the current focus on sustainable transportation, dial-a-ride (DAR) services are becoming an increasingly important addition to regular transportation modes. This thesis has proposed a solution method that successfully solves real-life instances of the disruptive DARP faced by RAT. Compared with simpler solvers, the heuristic is able to serve more requests and better reduce the impact of delays. The work may thus serve as a basis for further research on the disruptive DARP. Additionally, it was shown that several of the proposed policies resulted in a lower rejection rate and more cost-effective routes, thus providing RAT with valuable insights relevant to their decision-making. In turn, this can help RAT become more profitable and deliver a better service to the elderly.

Chapter 11

Future Research

Although the proposed solution method successfully solves DARP instances of realistic size, there are still several ways to make the heuristic and simulation framework more adaptable to the real-life problem. Additionally, there exist several extensions of the problem that may be interesting to investigate further. Therefore, this chapter aims to highlight future research opportunities related to the DARP faced by RAT.

The problem considered in this thesis is a dynamic and deterministic DARP. Consequently, the aspect of uncertainty is not included. According to Ritzinger et al. (2022), most dynamic and stochastic solution approaches show that the solution quality benefits from incorporating stochastic information. The proposed solution method handles unforeseen events by solving the DARP dynamically throughout the day. However, as seen in Section 9.3.2, the room for improvement by re-optimisation is limited due to a tightly constrained problem. By utilising stochastic information in the initial planning, it may be possible to schedule the requests in a way that leaves more room for new requests. Hence, to further improve the solution, stochastic models are worth considering.

One stochastic element to include is the uncertainty of travel times. Although the proposed solution method includes simple rush-hour modelling, it assumes deterministic travel times. In reality, travel times are likely to vary due to various travel patterns and unforeseen events causing delays. As Ruter operates most public transport in Oslo, data on travel times are assumed to be easily retrieved. By utilising this data, the initial routes can be created with robustness in mind, which might help further reduce the impact of delays.

Stochastic information regarding the appearance of future user requests may also be relevant. As presented in Section 9.3.4, implementing an one-hour deadline on new requests resulted in fewer rejections. This indicates that a lower rejection rate might be obtained by utilising information regarding unknown future demand when solving the problem. By examining historical data, general patterns and trends can be discovered and used to predict future demand. As the service expands, RAT accumulates more and more historical demand data. Consequently, using stochastic information in the solution method becomes increasingly relevant.

Another possible area of research concerns the investigation of alternative objectives. As mentioned in Chapter 2, RAT struggles with a low degree of ride-sharing and a high cost per trip. Still, the proposed solution method does not include these aspects in the objective function. By optimising these terms, a decrease in the cost per trip and an increase in ride-sharing might be obtained. However, a trade-off exists between the quality of service and the mentioned objectives. For instance, relaxing the hard time windows and the maximum ride time restrictions is likely to yield more ride-sharing and a reduced cost per trip, but may also reduce customer satisfaction. Consequently, future research should also consider the importance of each objective and the interaction between them.

The integration of RAT with ordinary public transport may also be of interest. The integrated DARP includes the possibility for the users to transfer between different modes of transportation and travel some part of the route by ordinary public transport (Häll et al., 2009). As mentioned in Chapter 2, ordinary public transport is the transportation mode with the highest degree of ride-sharing. Consequently, from an environmental and economic point of view, the integrated DARP is of high interest. Ruter already manages most of the public transport in Oslo. Hence it could be interesting to evaluate different ways of integrating the current RAT service with the existing fixed-route network. This may also allow for extended operational areas, which, as mentioned in Chapter 2, is desired by the customers.

Bibliography

Aguiar, B. and Macário, R. (2017). The need for an elderly centred mobility policy. *Transportation Research Procedia*, 25:4355–4369.

Atahran, A., Lenté, C., and T'kindt, V. (2014). A multicriteria dial-a-ride problem with an ecological measure and heterogeneous vehicles. *Journal of Multi-Criteria Decision Analysis*, 21(5-6):279–298.

Beaudry, A., Laporte, G., Melo, T., and Nickel, S. (2010). Dynamic transportation of patients in hospitals. *OR Spectrum*, 32(1):77–107.

Belhaiza, S. (2019). A hybrid adaptive large neighborhood heuristic for a real-life dial-a-ride problem. *Algorithms*, 12(2).

Berbeglia, G., Cordeau, J.-F., and Laporte, G. (2012). A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. *INFORMS Journal on Computing*, 24:343–355.

Brenna, M. L., Ellestad, A. O., and Lunde, A. S. (2021). A rolling horizon approach to a dynamic dial-a-ride formulation of ruter aldersvennlig transport. Project report. Norwegian University of Science and Technology.

Colorni, A. and Righini, G. (2001). Modeling and optimizing dynamic dial-a-ride problems. *International Transactions in Operational Research*, 8(2):155–166.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54:573–586.

Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem (darp): Models and algorithms. *Annals of Operations Research*, 153:29–46.

COWI (2018). Bestillingstransport - erfaringer og muligheter. `https://www.skyss.no/globalassets/om-skyss/strategiar-og-fagstoff/fagrapportar-og-utgreiingar/2018/bestillingstransport---erfaringer-og-muligheter.pdf`. Accessed on 08.06.2022.

Detti, P., Papalini, F., and de Lara, G. Z. M. (2017). A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega*, 70:1–14.

Eglese, R. and Zambirinis, S. (2018). Disruption management in vehicle routing and scheduling for road freight transport: a review. *Top*, 26(1):1–17.

Eidstuen, A. (2011). Groruddalssatsingen. `https://mgromdotcom.wordpress.com/2011/08/13/groruddalssatsingen`. Accessed on 10.12.2021.

Epinion (2021). Brukerundersøkelse - aldersvennlig transport. Epinion Aarhus.

Frøyland, P. (2009). Trafikk i kollektivfelt. `https://vegvesen.brage.unit.no/vegvesen-xmlui/bitstream/handle/11250/2686512/T%C3%98I-rapport%20176.pdf?sequence=1&isAllowed=y`. Accessed on 23.05.2022.

Gabbiani, F. and Cox, S. J. (2010). Chapter 16 - stochastic processes. In Gabbiani, F. and Cox, S. J., editors, *Mathematics for Neuroscientists*, pages 251–266. Academic Press, London.

Garaix, T., Artigues, C., Feillet, D., and Josselin, D. (2010). Vehicle routing problems with alternative paths: An application to on-demand transportation. *European Journal of Operational Research*, 204(1):62–75.

Garaix, T., Artigues, C., Feillet, D., and Josselin, D. (2011). Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. *Computers & Operations Research*, 38(10):1435–1442.

Gaul, D., Klamroth, K., and Stiglmayr, M. (2021). Solving the Dynamic Dial-a-Ride Problem Using a Rolling-Horizon Event-Based Graph. In Müller-Hannemann, M. and Perea, F., editors, *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021)*, volume 96 of *Open Access Series in Informatics (OASIcs)*, pages 1–16, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Gilhooly, M., Hamilton, K., O'Neill, M., Gow, J., Webster, N., Pike, F., and Bainbridge, D. (2002). Transport and ageing: Extending quality of life for older people via public and private transport. Glasgow Caledonian University.

Google (2021). About Google Scholar. `https://scholar.google.com/intl/en/scholar/about.html`. Accessed on 25.11.2021.

Häll, C. H., Andersson, H., Lundgren, J. T., and Värbrand, P. (2009). The integrated dial-a-ride problem. *Public Transport*, 1(1):39–54.

Handagard, I. (2019). Knusende dom over kollektivtilbudet i de største byene. `https://www.naf.no/om-naf/naf-mener/knusende-dom-over-kollektivtilbudet-i-de-storste-byene/`. Accessed on 23.11.2021.

Heilporn, G., Cordeau, J.-F., and Laporte, G. (2011). An integer l-shaped algorithm for the dial-a-ride problem with stochastic customer delays. *Discrete Applied Mathematics*, 159(9):883–895.

Helgheim, S. V. and Rydland, S. (2012). Knusende dom over kollektivtilbudet i de største byene. `https://www.nrk.no/vestland/her-kommer-bussene-raskest-frem-1.8352939`. Accessed on 09.05.2022.

Hjorthol, R., Nordbakke, S., Vågane, L., Levin, L., Siren, A., and Ulleberg, P. (2011). Eldres mobilitet og velferd – utvikling, reisebehov og tiltak. Transportøkonomisk institutt (TØI), 1179/2011.

Ho, S. and Haugland, D. (2011). Local search heuristics for the probabilistic dial-a-ride problem. *OR Spectrum*, 33:961–988.

Ho, S. C., Szeto, W., Kuo, Y.-H., Leung, J. M., Petering, M., and Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111:395–421.

Hyytiä, E., Aalto, S., Penttinen, A., and Sulonen, R. (2010). A stochastic model for a vehicle in a dial-a-ride system. *Operations Research Letters*, 38(5):432–435.

Hyytiä, E., Penttinen, A., and Sulonen, R. (2012). Non-myopic vehicle and route selection in dynamic darp with travel time and workload objectives. *Computers & Operations Research*, 39(12):3021–3030.

Jaw, J.-J., Odoni, A. R., Psaraftis, H. N., and Wilson, N. H. (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 20(3):243–257.

Karabuk, S. (2009). A nested decomposition approach for solving the paratransit vehicle scheduling problem. *Transportation Research Part B: Methodological*, 43(4):448–465.

Lehuédé, F., Masson, R., Parragh, S. N., Péton, O., and Tricoire, F. (2014). A multi-criteria large neighbourhood search for the transportation of disabled people. *Journal of the Operational Research Society*, 65(7):983–1000.

Luo, Z., Liu, M., and Lim, A. (2019). A two-phase branch-and-price-and-cut for a dial-a-ride problem in patient transportation. *Transportation Science*, 53(1):113–130.

Marković, N., Nair, R., Schonfeld, P., Miller-Hooks, E., and Mohebbi, M. (2015). Optimizing dial-a-ride services in Maryland: Benefits of computerized routing and scheduling. *Transportation Research Part C: Emerging Technologies*, 55:156–165.

Masmoudi, M. A., Braekers, K., Masmoudi, M., and Dammak, A. (2017). A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Computers & Operations Research*, 81:1–13.

Molenbruch, Y., Braekers, K., and Caris, A. (2017). Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, 259(1):295–325.

Muñoz-Carpintero, D., Saez, D., Cortés, C., and Núñez, A. (2015). A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach. *Transportation Science*, 49:239–253.

Nasri, S., Bouziri, H., and Aggoune-Mtalaa, W. (2021). Customer-oriented dial-a-ride problems: A survey on relevant variants, solution approaches and applications. In Ben Ahmed, M., Mellouli, S., Braganca, L., Anouar Abdelhakim, B., and Bernadetta, K. A., editors, *Emerging Trends in ICT for Sustainable Development*, pages 111–119, Cham. Springer International Publishing.

Nordbakke, S., Philips, R., Skollerud, K., and Milch, V. (2020). Helseeffekter av ruter aldersvennlig transport. Transportøkonomisk institutt (TØI), 1810/2020.

Núñez, A., Cortés, C. E., Sáez, D., De Schutter, B., and Gendreau, M. (2014). Multi-objective model predictive control for dynamic pickup and delivery problems. *Control Engineering Practice*, 32:73–86.

O'Fallon, C. and Sullivan, C. (2003). Older people's travel patterns transport sustainability in New Zealand cities. *26th Australasian Transport Research Forum Wellington New Zealand.*

Ojeda Rios, B. H., Xavier, E. C., Miyazawa, F. K., Amorim, P., Curcio, E., and Santos, M. J. (2021). Recent dynamic vehicle routing problems: A survey. *Computers & Industrial Engineering*, 160. 107604.

Paquay, C., Crama, Y., and Pironet, T. (2020). Recovery management for a dial-a-ride system with real-time disruptions. *European Journal of Operational Research*, 280(3):953–969.

Paquette, J., Cordeau, J.-F., Laporte, G., and Pascoal, M. M. (2013). Combining multi-criteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological*, 52:1–16.

Parragh, S. N., Pinho de Sousa, J., and Almada-Lobo, B. (2015). The dial-a-ride problem with split requests and profits. *Transportation Science*, 49(2):311–334.

Pfeiffer, C. and Schulz, A. (2022). An alns algorithm for the static dial-a-ride problem with ride and waiting time minimization. *OR Spectrum*, 44(1):87–119.

Posada, M., Andersson, H., and Häll, C. H. (2017). The integrated dial-a-ride problem with timetabled fixed route service. *Public Transport*, 9(1):217–241.

Psaraftis, H. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14:130–154.

Qu, Y. and Bard, J. F. (2015). A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. *Transportation Science*, 49(2):254–270.

Ramasamy Pandi, R., Ho, S. G., Nagavarapu, S. C., Tripathy, T., and Dauwels, J. (2020). Disruption management for dial-a-ride systems. *IEEE Intelligent Transportation Systems Magazine*, 12(4):219–234.

Ritzinger, U., Puchinger, J., Rudloff, C., and Hartl, R. F. (2022). Comparison of anticipatory algorithms for a dial-a-ride problem. *European Journal of Operational Research*, 301(2):591–608.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472.

Savelsbergh, M. and Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, 29:17–29.

Schilde, M., Doerner, K., and Hartl, R. (2011). Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & Operations Research*, 38(12):1719–1730.

Schilde, M., Doerner, K., and Hartl, R. (2014). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, 238(1):18–30.

Souza, A., Chagas, J., Penna, P., and Souza, M. (2020). A hybrid heuristic algorithm for the dial-a-ride problem. In *Variable Neighborhood Search*, pages 53–66. Springer International Publishing.

Souza, A. L., Bernardo, M., Penna, P. H., Pannek, J., and Souza, M. J. (2022). Bi-objective optimization model for the heterogeneous dynamic dial-a-ride problem with no rejects. *Optimization Letters*, 16(1):355–374.

Statistisk sentralbyrå (2020). Nasjonale befolkningsframskrivinger 12881: Framskrevet folkemengde 1. januar, etter kjønn, alder, innvandringskategori og landbakgrunn, i 15 alternativer 2020 - 2100. `https://www.ssb.no/statbank/table/12881/`. Accessed on 22.05.2022.

Statistisk sentralbyrå (2022). Befolkning 10211: Alders- og kjønnsfordeling i hele befolkningen 1846 - 2022. `https://www.ssb.no/statbank/table/10211/`. Accessed on 22.05.2022.

Syed, A. A., Kaltenhaeuser, B., Gaponova, I., and Bogenberger, K. (2019). Asynchronous adaptive large neighborhood search algorithm for dynamic matching problem in ride hailing services. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3006–3012.

Turkeš, R., Sörensen, K., and Hvattum, L. M. (2021). Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. *European Journal of Operational Research*, 292(2):423–442.

United Nations, D. o. E. and Social Affairs, P. D. . (2019). World population ageing 2019: Highlights. *ST/ESA/SER.A/430.*

Utviklingssenter for sykehjem og hjemmetjenester (2020). Ruter aldersvennlig transport (Rosa busser). `https://levehelelivet.utviklingssenter.no/idebanken-leve-hele-livet/ruter-aldersvennlig-transport-rosa-busser`. Accessed on 20.11.2021.

Vallee, S., Oulamara, A., and Cherif-Khettaf, W. R. (2020). New online reinsertion approaches for a dynamic dial-a-ride problem. *Journal of Computational Science*, 47:101–199.

Wang, X., Ruan, J., and Shi, Y. (2012). A recovery model for combinational disruptions in logistics delivery: Considering the real-world participators. *International Journal of Production Economics*, 140(1):508–520.

Xiang, Z., Chu, C., and Chen, H. (2008). The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. *European Journal of Operational Research*, 185(2):534–551.

Yu, G. and Qi, X. (2004). *Disruption management: framework, models and applications.* World Scientific.

Appendix

A Formulation of the Mathematical Model

A.1 Notation

Sets

$\mathcal{K}$	-	set of vehicles $k \in \{1, ..., K\}$
$\mathcal{P}$	-	set of pick-up nodes $i, j \in \{1, ..., n\}$
$\mathcal{D}$	-	set of drop-off nodes $i, j \in \{n+1, ..., 2n\}$
$\mathcal{N}$	-	set of pick-up and drop-off nodes $i, j \in \{1, ..., 2n\}$
$\mathcal{N}^D$	-	set of all nodes $i, j \in \{1, ..., 2n, o(1), ..., o(K), d(1), ..., d(K)\}$
$\mathcal{A}$	-	set of all feasible arcs $(i, j) \in \mathcal{A} \subset \mathcal{N}^D \times \mathcal{N}^D$

Parameters

C^D_{ijk}	-	cost of traveling the distance from node i to node j using vehicle k
C^T	-	penalty cost for violation of soft time window
C^R	-	penalty cost for rejecting a request
Q^S_k	-	standard seats capacity of vehicle k
Q^W_k	-	wheelchair capacity of vehicle k
$o(k)$	-	origin of vehicle k
$d(k)$	-	artificial destination of vehicle k
L^S_i	-	standard seats load of request i
L^W_i	-	wheelchair load of request i
T_{ij}	-	direct traveling time from node i to node j
$\underline{T}^S_i$	-	lower bound of soft time window for time of service of request i
$\overline{T}^S_i$	-	upper bound of soft time window for time of service of request i
$\underline{T}^H_i$	-	lower bound of hard time window for time of service of request i
$\overline{T}^H_i$	-	upper bound of hard time window for time of service of request i
S_i	-	duration of embarking and disembarking of passenger(s) of request i at a node
F	-	fraction of direct travel time the actual travel time can exceed
α	-	weight of driving time term in objective function
β	-	weight of soft time window violation term in objective function

Decision variables

s_i	-	$\begin{cases} 1, & \text{if request } i \text{ is not served} \\ 0, & \text{otherwise} \end{cases}$
x_{ijk}	-	$\begin{cases} 1, & \text{if vehicle } k \text{ drives directly from node } i \text{ to node } j \\ 0, & \text{otherwise} \end{cases}$
q^S_{ik}	-	standard seats load of vehicle k when leaving node i
q^W_{ik}	-	wheelchair load of vehicle k when leaving node i
t_i	-	time of departure from node i
l_i	-	violation of lower bound of soft time window for time of departure from node i
u_i	-	violation of upper bound of soft time window for time of departure from node i

A.2 Mathematical Model

$$\min \alpha \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} C^D_{ijk} x_{ijk} + \beta \sum_{i\in\mathcal{N}} C^T(l_i + u_i) + \sum_{i\in\mathcal{P}} C^R s_i \tag{A.1}$$

$$\text{s.t.} \qquad \sum_{j\in\mathcal{N}^D} \sum_{k\in\mathcal{K}} x_{ijk} = 1 - s_i \qquad i \in \mathcal{P} \tag{A.2}$$

$$\sum_{j\in\mathcal{N}^D} x_{o(k)jk} = \sum_{i\in\mathcal{N}^D} x_{id(k)k} \quad = 1 \qquad k \in \mathcal{K} \tag{A.3}$$

$$\sum_{j\in\mathcal{N}^D} x_{jik} - \sum_{j\in\mathcal{N}^D} x_{ijk} = 0 \qquad i \in \mathcal{N},\ k \in \mathcal{K} \tag{A.4}$$

$$\sum_{j\in\mathcal{N}^D} x_{ijk} - \sum_{j\in\mathcal{N}^D} x_{n+i,j,k} = 0 \qquad i \in \mathcal{P},\ k \in \mathcal{K} \tag{A.5}$$

$$q^S_{o(k)k} = 0 \qquad k \in \mathcal{K} \tag{A.6}$$

$$q^S_{ik} + L^S_j - q^S_{jk} \le (Q^S_k + L^S_j)(1 - x_{ijk}) \qquad j \in \mathcal{P},\ (i,j) \in \mathcal{A},\ k \in \mathcal{K} \tag{A.7}$$

$$q^S_{ik} - L^S_j - q^S_{n+j,k} \le Q^S_k(1 - x_{i,n+j,k}) \qquad j \in \mathcal{P},\ (i,n+j) \in \mathcal{A},\ k \in \mathcal{K} \tag{A.8}$$

$$\sum_{j\in\mathcal{N}^D} L^S_i x_{ijk} \le q^S_{ik} \le \sum_{j\in\mathcal{N}^D} Q^S_k x_{ijk} \qquad i \in \mathcal{P},\ k \in \mathcal{K} \tag{A.9}$$

$$\sum_{j\in\mathcal{N}^D} (Q^S_k - L^S_i) x_{n+i,j,k} \ge q^S_{n+i,k} \qquad i \in \mathcal{P},\ k \in \mathcal{K} \tag{A.10}$$

$$q^S_{ik} \le Q^S_k(1 - x_{id(k)k}) \qquad i \in \mathcal{D},\ k \in \mathcal{K} \tag{A.11}$$

$$q^W_{o(k)k} = 0 \qquad k \in \mathcal{K} \tag{A.12}$$

$$q^W_{ik} + L^W_j - q^W_{jk} \le (Q^W_k + L^W_j)(1 - x_{ijk}) \qquad j \in \mathcal{P},\ (i,j) \in \mathcal{A},\ k \in \mathcal{K} \tag{A.13}$$

$$q^W_{ik} - L^W_j - q^W_{n+j,k} \le Q^W_k(1 - x_{i,n+j,k}) \qquad j \in \mathcal{P},\ (i,n+j) \in \mathcal{A},\ k \in \mathcal{K} \tag{A.14}$$

$$\sum_{j\in\mathcal{N}^D} L^W_i x_{ijk} \le q^W_{ik} \le \sum_{j\in\mathcal{N}^D} Q^W_k x_{ijk} \qquad i \in \mathcal{P},\ k \in \mathcal{K} \tag{A.15}$$

$$\sum_{j \in \mathcal{N}^D} (Q_k^W - L_i^W) x_{n+i,j,k} \geq q_{n+i,k}^W \qquad i \in \mathcal{P},\ k \in \mathcal{K} \tag{A.16}$$

$$q_{ik}^W \leq Q_k^W (1 - x_{id(k)k}) \qquad i \in \mathcal{D},\ k \in \mathcal{K} \tag{A.17}$$

$$\underline{T}_i^S - l_i \leq t_i \leq \overline{T}_i^S + u_i \qquad i \in \mathcal{N} \tag{A.18}$$

$$\underline{T}_i^H \leq t_i \leq \overline{T}_i^H \qquad i \in \mathcal{N} \tag{A.19}$$

$$t_i + T_{ij} + S_j - t_j \leq M_{ij}(1 - x_{ijk}) \qquad i, j \in \mathcal{N},\ k \in \mathcal{K} \tag{A.20}$$

$$t_i + T_{i,n+i} + S_{n+i} - t_{n+i} \leq 0 \qquad i \in \mathcal{P} \tag{A.21}$$

$$t_{n+i} - S_{n+i} - t_i \leq (1 + F) T_{i,n+i} \qquad i \in \mathcal{P} \tag{A.22}$$

$$s_i \in \{0, 1\} \qquad i \in \mathcal{P} \tag{A.23}$$

$$x_{ijk} \in \{0, 1\} \qquad (i, j) \in \mathcal{A},\ k \in \mathcal{K} \tag{A.24}$$

$$q_{ik}^S, q_{ik}^W \geq 0,\ \text{integer} \qquad i \in \mathcal{N}^D,\ k \in \mathcal{K} \tag{A.25}$$

$$t_i, l_i, u_i \geq 0 \qquad i \in \mathcal{N} \tag{A.26}$$

B Results from Parameter Tuning

The values marked in green represent the final configurations utilised in the model.

B.1 Parameter Tuning of Number of Iterations in ALNS

Table B.1: Results parameter tuning of number of iterations in ALNS, I.

		Objective (hours)		**Computational time (seconds)**	
I	**Instance**	**Average**	**Standard Deviation**	**Average**	**Standard Deviation**
50	L1	429.63	323.55	201.93	11.43
	L2	86.20	1.78	215.85	11.85
	L3	2,113.33	918.38	244.27	8.22
100	L1	153.23	189.43	394.41	25.57
	L2	84.27	1.74	435.48	22.10
	L3	1,772.91	654.62	485.10	23.99
150	L1	93.24	7.11	521.03	17.69
	L2	84.16	2.01	546.17	18.81
	L3	1,849.52	2,129.91	688.01	41.79

B.2 Parameter Tuning of Start Temperature Control Parameter

Table B.2: Results parameter tuning of start temperature control parameter, Z.

		Objective (hours)		**Computational time (seconds)**	
Z	**Instance**	**Average**	**Standard Deviation**	**Average**	**Standard Deviation**
0.20	L1	636.00	1,326.50	393.00	40.70
	L2	85.40	2.90	447.70	23.70
	L3	1,831.20	1,657.30	491.30	14.80
0.30	L1	652.20	1,147.10	388.50	26.60
	L2	85.40	1.90	444.30	18.20
	L3	2,202.80	1,658.70	479.00	37.50
0.40	L1	225.80	252.60	357.80	33.60
	L2	85.10	2.00	408.40	29.10
	L3	3,845.90	3,707.50	440.20	38.50
0.60	L1	153.20	189.40	394.40	25.60
	L2	84.30	1.70	435.50	22.10
	L3	1,772.90	654.60	485.10	24.00
0.80	L1	229.55	261.20	395.80	0.01
	L2	217.66	420.10	435.00	0.00
	L3	2,130.48	1,397.30	491.50	0.00

B.3 Parameter Tuning of Cooling Rate

Table B.3: Results parameter tuning of cooling rate, C.

		Objective (hours)		Computational time (seconds)	
C	Instance	Average	Standard Deviation	Average	Standard Deviation
0.92	L1	781.76	1,934.42	400.68	29.79
	L2	84.60	2.34	428.98	27.95
	L3	3,186.09	3,393.25	491.99	31.52
0.94	L1	226.80	247.69	404.50	24.30
	L2	84.27	1.48	429.30	25.89
	L3	2,472.95	2,109.01	466.46	35.48
0.96	L1	153.23	189.43	394.41	25.57
	L2	84.27	1.74	435.48	22.10
	L3	1,772.91	654.62	485.1	23.99
0.98	L1	282.04	290.57	404.31	25.91
	L2	85.29	2.22	425.81	21.16
	L3	1,841.14	561.62	464.87	26.02

B.4 Parameter Tuning of Destruction Degree

Table B.4: Results parameter tuning of destruction degree, D.

		Objective (hours)		Computational time (seconds)	
D	Instance	Average	Standard Deviation	Average	Standard Deviation
0.20	L1	1,659.15	1,213.74	241.13	5.46
	L2	343.95	431.10	259.7	21.46
	L3	6,424.97	2,242.77	328.69	31.31
0.30	L1	441.28	429.95	336.69	19.72
	L2	234.26	417.05	384.77	19.12
	L3	2,372.13	1,983.23	425.92	18.47
0.40	L1	153.23	189.43	394.41	25.57
	L2	84.27	1.74	435.48	22.1
	L3	1,772.91	654.62	485.1	23.99
0.50	L1	299.33	326.75	446.62	27.3
	L2	86.15	2.35	494.15	49.27
	L3	1,772.02	640.67	527.63	46.14
0.60	L1	533.22	405.45	430.37	15.36
	L2	86.10	1.87	504.3	46.53
	L3	2,218.31	821.27	598.73	19.76

B.5 Parameter Tuning of Reaction Factor

Table B.5: Results parameter tuning of reaction factor, R.

		Objective (hours)		Computational time (seconds)	
R	Instance	Average	Standard Deviation	Average	Standard Deviation
0.40	L1	278.17	408.21	402.49	28.66
	L2	85.07	1.91	438.38	29.77
	L3	2,263.39	2,131.66	482.62	15.79
0.50	L1	104.40	21.16	391.64	22.34
	L2	85.13	3.26	430.87	13.47
	L3	3,022.09	2,439.03	490.52	32.88
0.60	L1	342.69	432.23	404.32	17.47
	L2	84.26	1.44	448.02	22.22
	L3	3,229.18	3,024.26	465.32	40.65
0.70	L1	153.23	189.43	394.41	25.57
	L2	84.27	1.74	435.48	22.1
	L3	1,772.91	654.62	485.1	23.99
0.80	L1	570.69	749.07	364.3	20.36
	L2	86.83	6.00	402.77	18.15
	L3	1,741.65	1,801.85	449.91	39.49

B.6 Parameter Tuning of Weight Scores

Table B.6: Results parameter tuning of weight scores, (V_1, V_2, V_3).

		Objective (hours)		Computational time (seconds)	
(V_1, V_2, V_3)	Instance	Average	Standard Deviation	Average	Standard Deviation
(15, 10, 5)	L1	153.23	189.43	394.4	25.6
	L2	84.27	1.74	435.5	22.1
	L3	1,772.91	654.62	485.1	24
(15, 5, 10)	L1	347.74	428.42	389.04	36.71
	L2	85.56	2.76	446.99	17.61
	L3	3,228.09	2,451.37	489.14	26.28
(10, 1, 5)	L1	283.59	298.34	399.69	28.58
	L2	147.53	197.09	433.25	20.41
	L3	2,456.80	2,810.48	499.32	14.89
(10, 5, 1)	L1	337.15	313.86	387.03	31.18
	L2	84.95	2.06	432.37	35.5
	L3	1,437.88	871.00	499.95	20.65

B.7 Parameter Tuning of Batch Size for Updating Weights

Table B.7: Results parameter tuning of batch size for updating weights, N_U.

		Objective (hours)		**Computational time (seconds)**	
N_U	**Instance**	**Average**	**Standard Deviation**	**Average**	**Standard Deviation**
0.10	L1	226.92	252.45	399.07	28.55
	L2	84.85	1.80	438.35	11.16
	L3	2,284.07	1,254.77	492.74	28.81
0.20	L1	337.15	313.86	387.03	31.18
	L2	84.95	2.06	432.37	35.5
	L3	1,437.88	871.00	499.95	20.65
0.30	L1	99.89	11.87	386.41	22.29
	L2	84.37	1.72	438.84	18.68
	L3	3,332.33	3,726.39	470.7	34.08
0.40	L1	891.82	1,906.66	361.8	18.18
	L2	84.86	1.25	396.6	19.91
	L3	2,855.66	2,711.51	431.31	14.52
0.50	L1	286.98	291.57	403.38	13.54
	L2	85.15	1.96	424.94	29.55
	L3	1,841.80	646.38	479.13	39.53

C Results from Technical Analysis

C.1 Value of Adaptivity

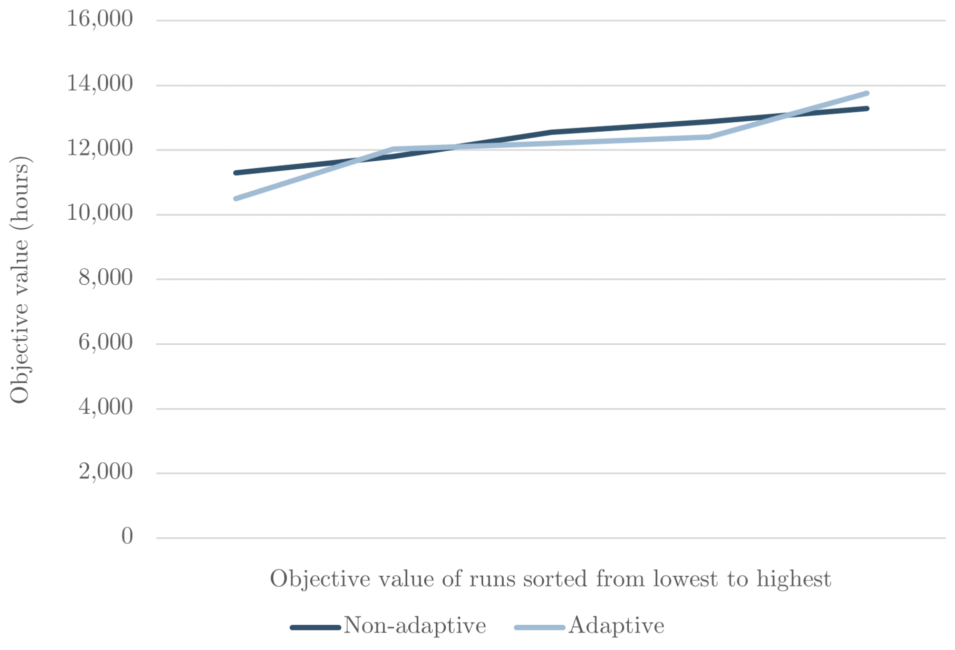


Figure C.1: Results from running 5 simulations with the original ALNS and the ALNS without adaptive weights on instance 1.

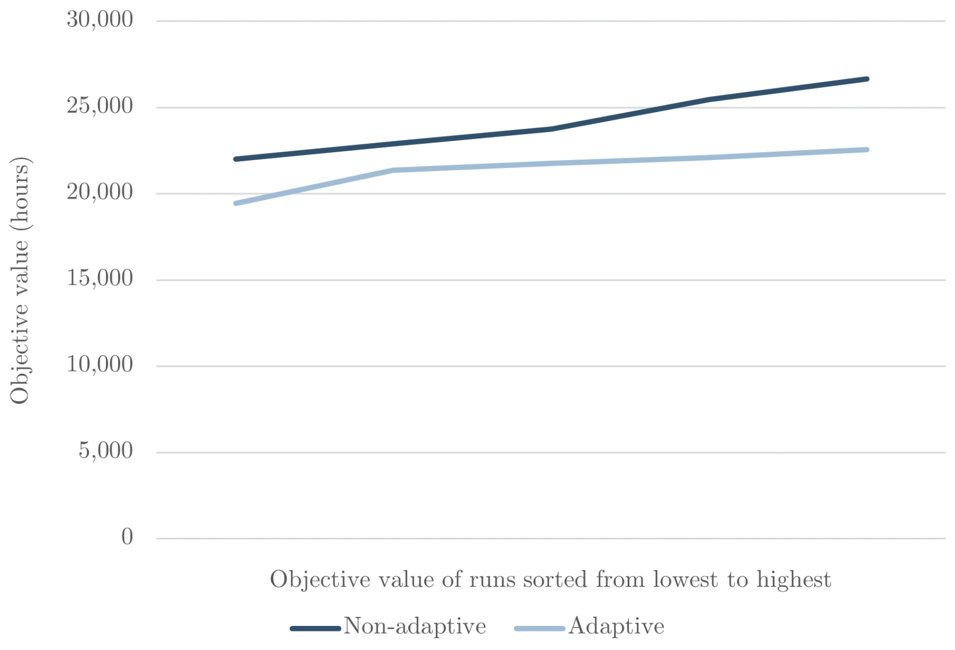


Figure C.2: Results from running 5 simulations with the original ALNS and the ALNS without adaptive weights on instance 2.

D Results from Managerial Insights

D.1 Evaluation of the Solution Method

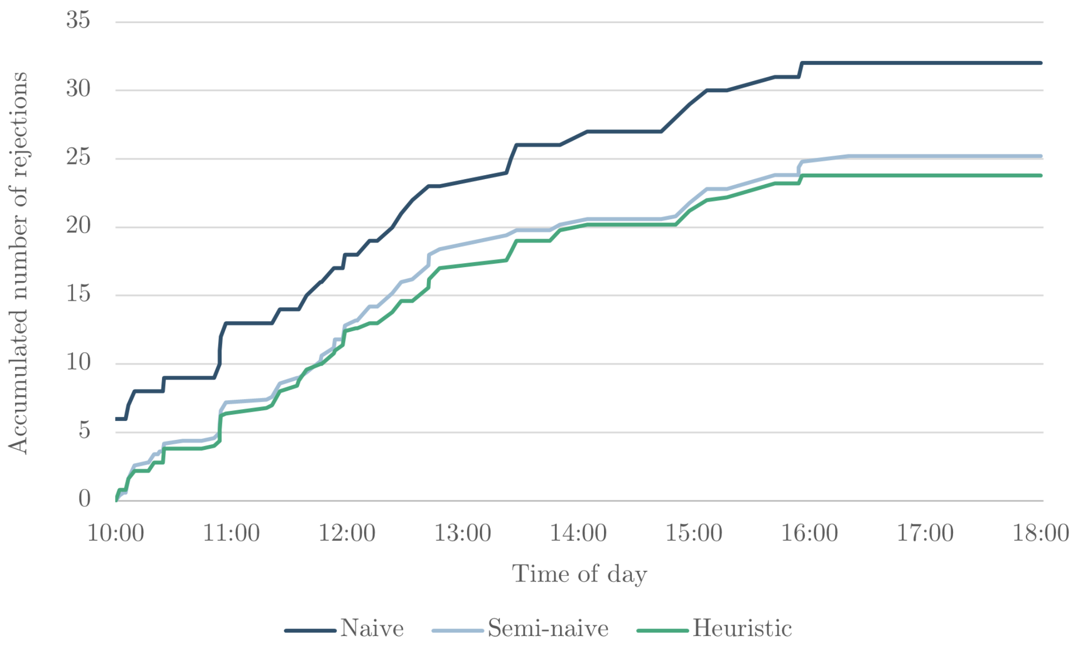


Figure D.1: Average accumulated number of rejections from running five simulations with the naive, semi-naive, and heuristic solver approaches on instance 2.

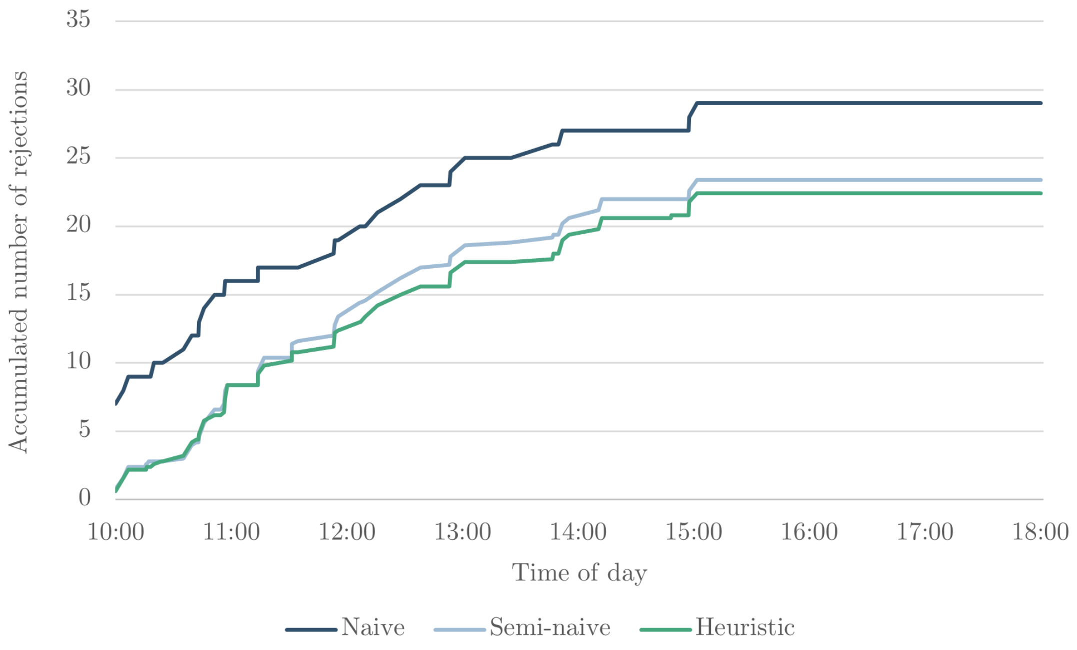


Figure D.2: Average accumulated number of rejections from running five simulations with the naive, semi-naive, and heuristic solver approaches on instance 3.

D.2 Value of Extending Pick-Up Interval

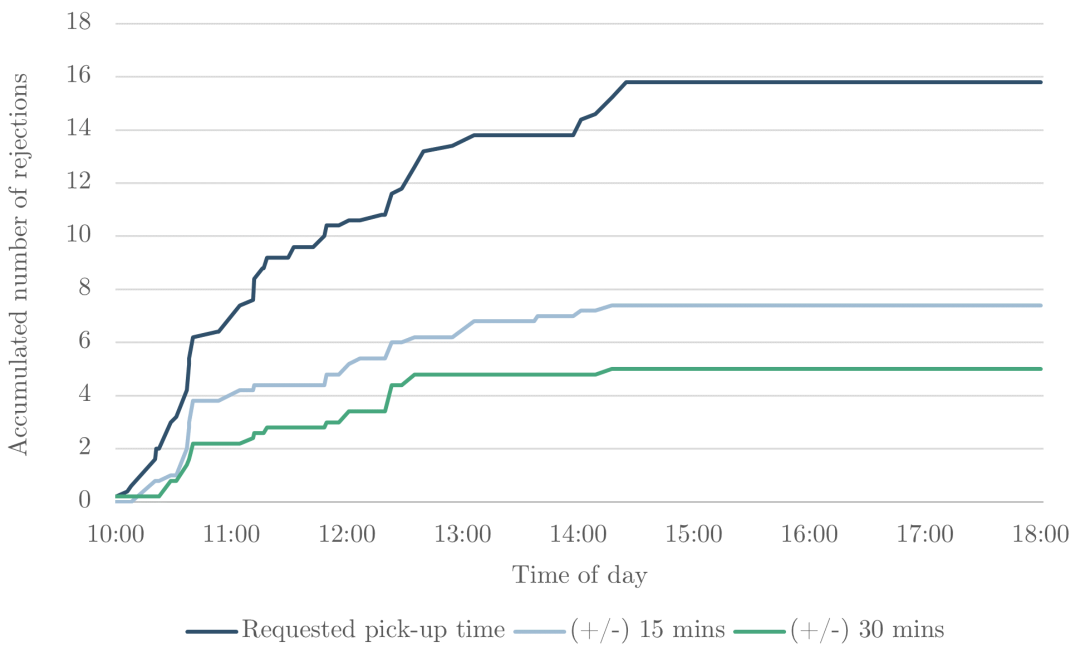


Figure D.3: Average accumulated number of rejections from running five simulations with different extended pick-up intervals on instance 1.

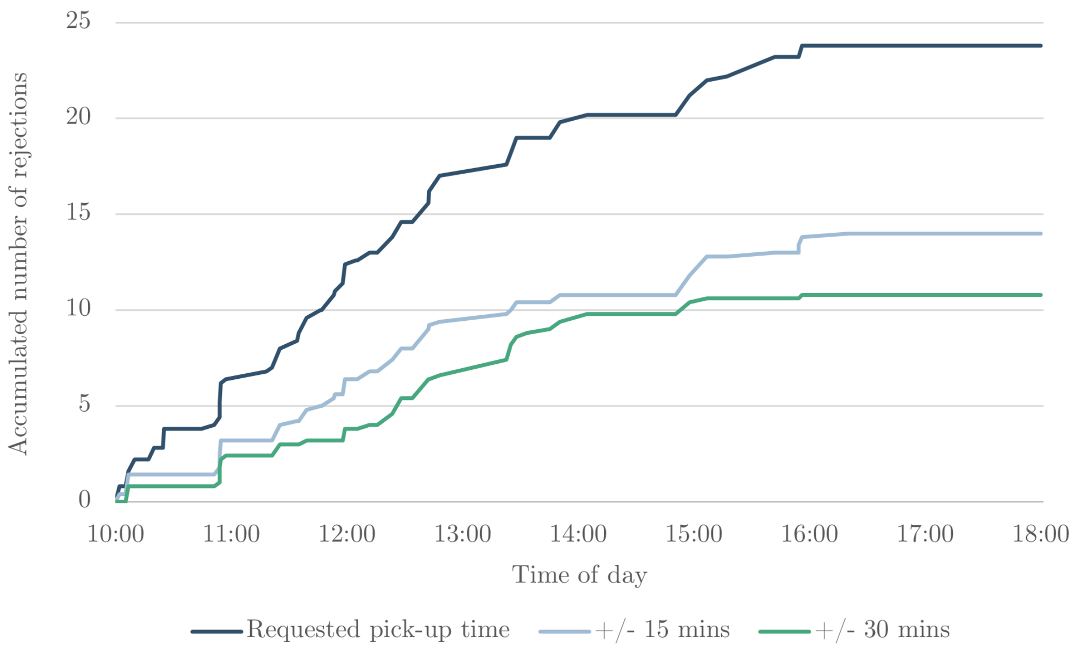


Figure D.4: Average accumulated number of rejections from running five simulations with different extended pick-up intervals on instance 2.

D.3 Rejected Requests per Hour Between Creation and Requested Time

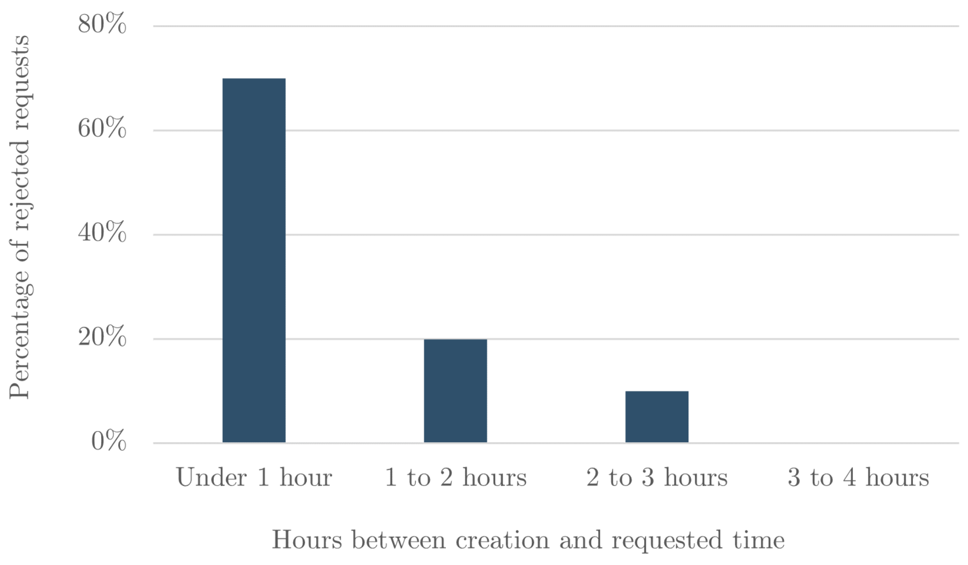


Figure D.5: Percentage of rejected requests per hour between creation and requested pick-up time from five simulations on instance 1.

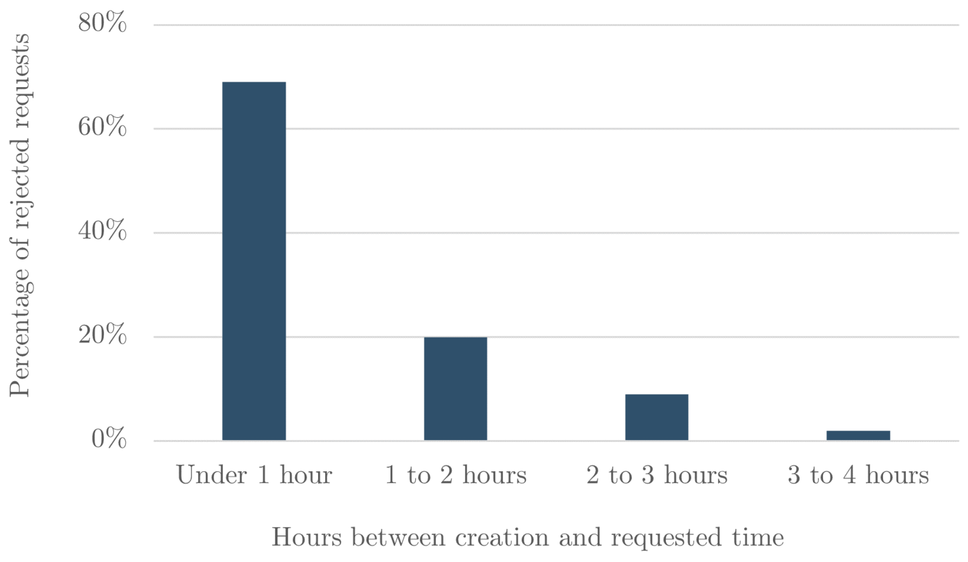


Figure D.6: Percentage of rejected requests per hour between creation and requested pick-up time from five simulations on instance 2.

D.4 Value of Implementing Order Deadline for New Requests

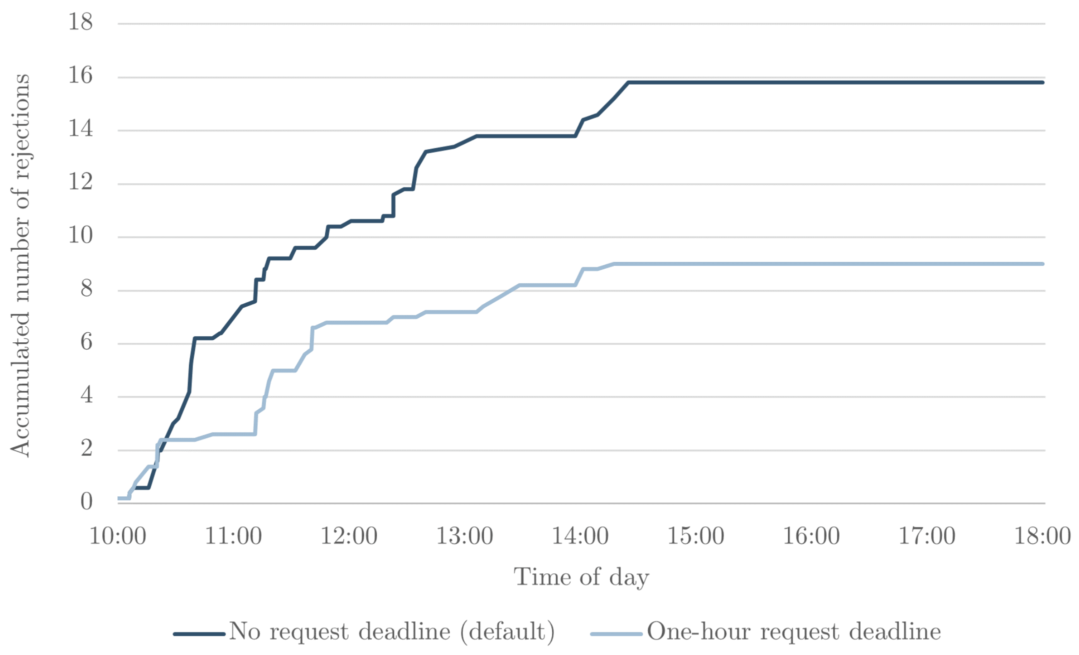


Figure D.7: Average accumulated number of rejections from running five simulations with and without new request deadline on instance 1.

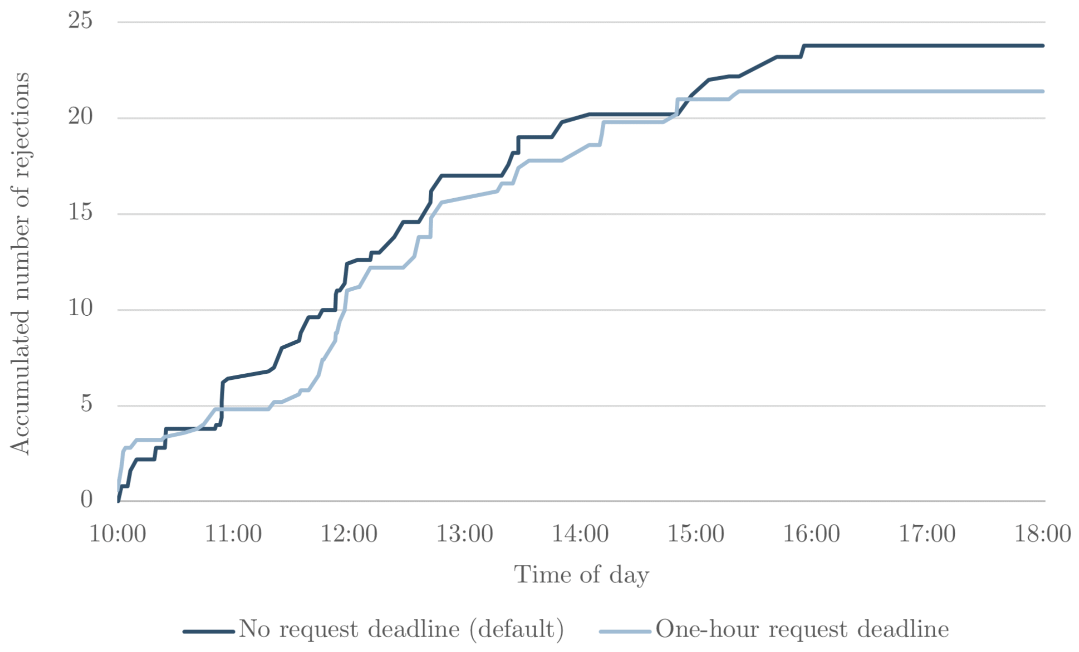


Figure D.8: Average accumulated number of rejections from running five simulations with and without new request deadline on instance 2.

D.5 Analysis of Vehicle Fleet Size

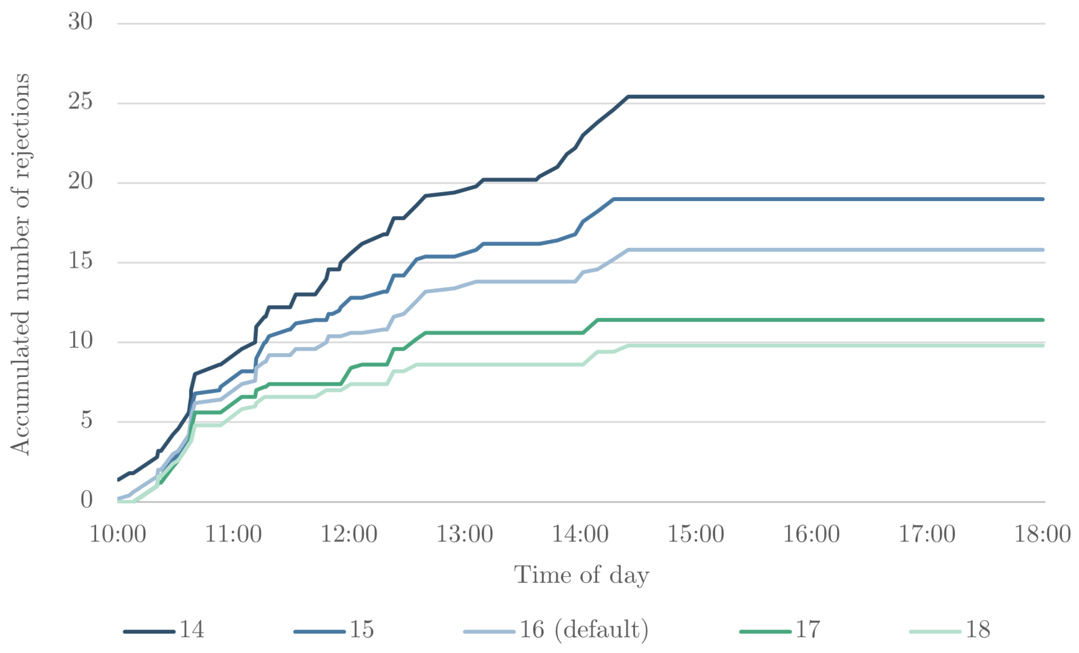


Figure D.9: Average accumulated number of rejections from running five simulations with different fleet sizes on instance 1.

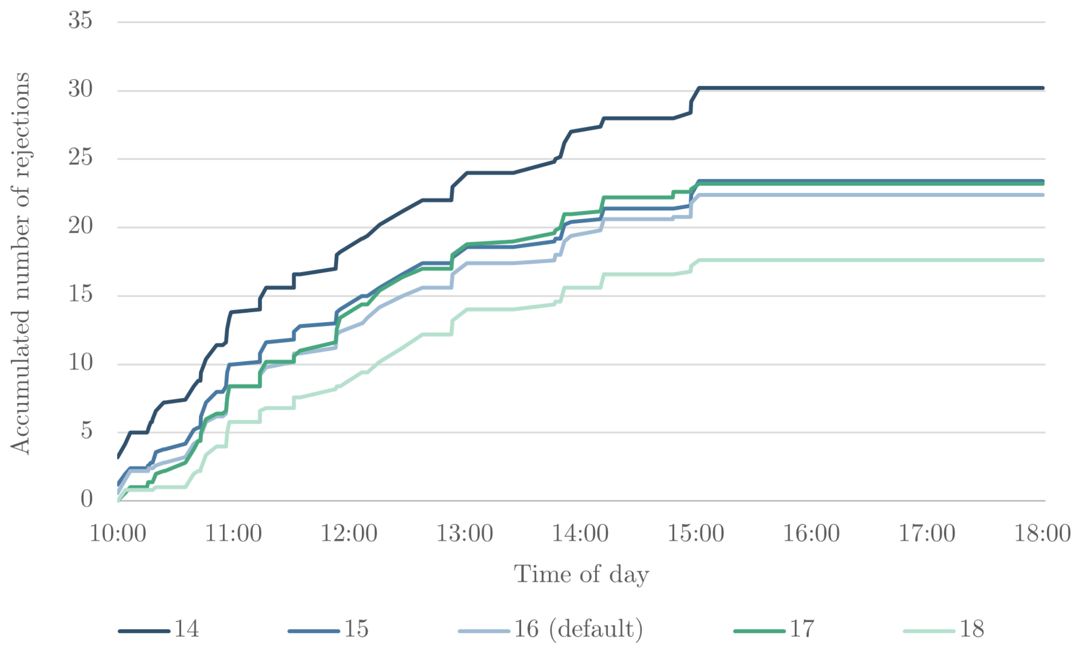


Figure D.10: Average accumulated number of rejections from running five simulations with different fleet sizes on instance 3.

Brenna, M. L., Ellestad, A. O. & Lunde, A. S.

An Adaptive Large Neighborhood Search Heuristic for a Dial-a-Ride Problem with Real-Time Disruptions

NTNU
Norwegian University of
Science and Technology