Master's thesis

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

Ulrik Eilertsen
Olav Mattias Falck-Pedersen
Jone Våge Henriksen

The Stochastic Electric-Vehicle Relocation Problem with Dynamic Pricing

Master's thesis in Industrial Economics and Technology Management
Supervisor: Kjetil Fagerholt
Co-supervisor: Giovanni Pantuso

June 2022

NTNU
Norwegian University of Science and Technology

Sapienza
Università di Roma

Ulrik Eilertsen
Olav Mattias Falck-Pedersen
Jone Våge Henriksen

The Stochastic Electric-Vehicle Relocation Problem with Dynamic Pricing

Master's thesis in Industrial Economics and Technology Management
Supervisor: Kjetil Fagerholt
Co-supervisor: Giovanni Pantuso
June 2022

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

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Department of Industrial Economics and
Technology Management

TIØ4905 - Managerial Economics and Operations Research,
Master's Thesis

The Stochastic Electric-Vehicle Relocation Problem with Dynamic Pricing

Authors:
Ulrik Eilertsen
Olav Mattias Falck-Pedersen
Jone Våge Henriksen

Supervisors:
Kjetil Fagerholt
Giovanni Pantuso
Laura Palagi

June, 2022

Preface

This master's thesis concludes our Master of Science in Industrial Economics and Technology Management at the Norwegian University of Science and Technology, Department of Industrial Economics and Technology Management. The thesis is a continuation of our specialization project written in the fall semester of 2021.

We wish to express our sincere appreciation to our supervisors Professor Kjetil Fagerholt at Norwegian University of Science and Technology, Associate Professor Giovanni Pantuso at University of Copenhagen and Professor Laura Palagi at Sapienza University of Rome. The final results of our thesis would not be the same without your invaluable experience, and the constant feedback and advice throughout the work. We would also like to thank Vybil and Kai Roger Kristoffersen for providing us insight to the problem and valuable discussions of the situation. We are very grateful for all of your help.

Trondheim, June 11, 2022

Ulrik Eilertsen, Olav Mattias Falck-Pedersen and Jone Våge Henriksen

Abstract

Car-sharing services have increased in popularity and size in recent years. These services' positive effects on the environment and several economic aspects are among the reasons for their increased popularity. Additionally, car-sharing services help mitigate issues related to congestion and parking problems.

In station-based car-sharing services, cars can be picked up and dropped off at designated stations, while in free-floating services, cars can be picked up and dropped off anywhere within a predefined service area. One-way station-based services allow customers to drop the car off at a different location than the pick-up location. Free-floating systems and one-way station-based systems lead to cars being differently located at any hour, and thus the challenge of rebalancing the fleet of cars occurs for the car-sharing operator (CSO). This core of the rebalancing problem is that customers drop off cars in low-demand areas and wish to pick up cars in high-demand areas. Further, electric vehicles have become more common in recent years, primarily due to progress in battery technology and increased environmental awareness. For free-floating car-sharing services, the additional challenge of finding the time and place to relocate electric vehicles for charging arises.

Employees of the CSO have traditionally performed relocations of the cars to maintain a balanced fleet. Revenue management via dynamically setting prices can also help redistribute the cars by introducing fees or bonuses to incentivize customers to move cars to preferable destinations. Additionally, the use of dynamic pricing can have a positive impact on the CSO's profit by exploiting the customers' willingness to pay. However, using dynamic pricing leads to new challenges for the CSO in predicting customer preferences and their reactions to different prices.

This thesis presents a model for the Stochastic Electric Vehicle Relocation Problem with Dynamic Pricing (SE-VReP-DP). The problem includes routing and scheduling of relocations performed by employees and dynamically setting prices through fees that incentivize desirable customer trips, with the goal of maximizing the profit of the CSO. We formulate a two-stage stochastic mixed-integer programming model to handle the uncertainty in customer behavior. The model's input includes initial vehicle, employee and customer distributions, initial battery levels of the electric vehicles, and potential customer trip fees. A random term influencing customer behavior introduces stochasticity to the model. The model returns a set of fees for the combinations of pick-up and drop-off areas, in addition to routes and schedules for employee-based relocations. Eilertsen et al. (2021) concluded that the model is too complex for a commercial solver to solve within a reasonable amount of time when using real-sized instances.

A heuristic approach is taken to solve real-sized instances. The heuristic developed in this thesis is an Adaptive Large Neighborhood Search (ALNS) heuristic integrated with two Local Search (LS) algorithms, adapted to solve the two-stage stochastic programming model of the SE-VReP-DP. The heuristic solution method is

decomposed into searching for prices to be assigned to the customers' trips and employee-based relocations separately. A rolling horizon framework is presented to gain managerial insight, using the proposed heuristic to solve the SE-VReP-DP on an hourly basis throughout a work day.

The results presented in this thesis reveal that the proposed heuristic outperforms a commercial solver in solving real-sized instances within a reasonable amount of time. For the largest instances, the commercial solver is unable to find any solution at all, while the heuristic quickly finds a solution. Further, dynamic pricing tends to yield higher profits for the CSO. The average profits gained from satisfying a customer are higher with dynamic pricing, indicating the prices better exploit the customers' willingness to pay. Using the rolling horizon framework reveals that dynamic pricing affects how employees and customers behave, leading to customers performing necessary charging of the fleet to a larger extent than without dynamic pricing.

The results also indicate that the ALNS heuristic performs better when increasing the runtime. This result implies that improvements could be made to the ALNS heuristic to find better solutions in less time. Different implementations and configurations should therefore be considered to further improve the performance of the proposed heuristic solution method.

Sammendrag

Bildelingstjenester har de siste årene blitt stadig mer utbredt. De positive virkningene disse tjenestene har på miljøet og en rekke økonomiske aspekter er noen av årsakene til deres økte populæritet. Bildelingstjenester bidrar også til å unngå problemer knyttet til trafikk og parkering.

I stasjonsbaserte bildelingstjenester blir biler plukket opp og levert til designerte stasjoner. I frittflytende systemer blir biler derimot plukket opp og levert hvor som helst innen et forhåndsdefinert område. Enveis, stasjonsbaserte systemer tillater kunder å levere bilene ved andre stasjoner enn der de ble plukket opp. Frittflytende og enveis, stasjonsbaserte systemer fører til at bildelingsoperatøren må opprettholde en balanse i bil-flåten. Dette problemet oppstår når biler blir avlevert i områder med lav etterspørsel og plukket opp i områder med høy etterspørsel. Elektriske biler har de siste årene blitt mer populært, hovedsakelig grunnet fremskritt innen batteriteknologi og økt fokus på miljø. For frittflytendede bildelingstjenester fører dette med seg utfordringen med å finne sted og tid til å sette de elektriske bilene til lading.

Ansatte av bildelingsoperatøren har tradisjonelt sett vært de som reposisjonerer bilene for å holde flåten balansert. Reposisjoneriner kan også håndteres ved at kunder blir insentivert med bonuser eller avgifter til å plukke opp biler i områder med lav etterspørsel og levere biler i områder med høy etterspørsel. Bruken av slike varierende bonuser og avgifter er det vi kaller dynamisk prising. Dynamisk prising fører til utfordringen knyttet til å forutse kundepreferanser og hvordan kunder reagerer på de forskjellige prisene.

I denne avhandlingen presenteres en modell for det Stokastiske Reposisjonseringsproblemet for Bildelingstjenester med Elektriske Biler og Dynamiske Priser (Stochastic Electric Vehicle Relocation Problem with Dynamic Pricing - SE-VReP-DP). Problemet går ut på å lage ruter og timeplaner for ansatte som skal flytte biler, og å dynamisk sette priser som insentiverer kunder til å bruke tjenesten på et vis som gagner bildelingsoperatøren ved å maksimere profitt. Vi formulerer et tostegs, stokastisk blandet heltallsprogrammeringsproblem for å håndtere usikkerheten knyttet til kundeoppførsel. Modellens input inkluderer initielle distribusjoner av biler, ansatte og kunder, initiellt batternivå for de elektriske bilene og potensielle priser for kundereisene. Usikkerhet blir introdusert i modellen gjennom en tilfeldig parameter. Modellen returnerer et sett av priser for kombinasjonen av områder for henting og levering av biler, i tillegg til ruter og timeplaner for de ansatte. Eilertsen et al. (2021) konkluderer med at modellen er for kompleks til å kunne løses av et kommersielt løsningsprogram når man har realistisk store instanser.

For å kunne løse instanser av reell størrelse foreslås en heuristisk løsningsmetode (en tommelfingermetode). Heuristikken som er utviklet er et adaptivt stort nabolagssøk (Adaptive Large Neighborhood Search - ALNS) integrert med lokalsøkoperatorer (Local Search). ALNS-heuristikken er tilpasset til den stokastiske settingen i SE-VReP-DP. Løsningsmetoden er dekomponert i to separate søk, der det første er for å finne priser og

det andre er for å finne hvilke reposisjoneringer som skal gjøres av de ansatte. Både problemspesifikke og velkjente nedbrytnings- og reparasjonsoperatorer blir brukt i ALNS-heuristikken. Videre blir problemet løst i et simuleringsverktøy (rolling horizon framework), for å skape relevante diskusjoner for problemet i en reell setting.

Resultatene indikerer at innføring av dynamisk prising gir høyere profitt og at ALNS-heuristikken utkonkurrerer en kommersiell løser for instanser av reell størrelse. For de største instansene klarer ikke en kommersiell løser å finne en gyldig løsning i det hele tatt innen kjøretidsbegrensningen. Videre viser resultatene at dynamiske priser kan påvirke oppførselen til både kunder og ansatte, og inviterer til diskusjon rundt rollen til de ansatte i bildefor bør forskjellige implementasjoner og konfigurasjoner vurderes for å videre forbedre resultatene fra den heurstiske løsningsmetoden. lingstjenesten. Resultatene viser også at ALNS-heuristikken gjør det bedre når kjøretiden øker, noe som indikerer at heuristikken kan forbedres for å finne de gode løsningene på kortere tid. Der

Contents

List of Figures

List of Tables

Abbreviations

ALNS	-	Adaptive Large Neighborhood Search
CSO	-	Car-Sharing Operator
DP	-	Dynamic Pricing
DPP	-	Dynamic Pricing Problem
EA	-	Evolutionary Algorithm
EV	-	Electric Vehicle
(E-)VReP	-	(Electric) Vehicle Relocation Problem
LNS	-	Large Neighborhood Search
LOS	-	Levels of Service
LS	-	Local Search
LSP	-	Local Search for Pricing solution
LSR	-	Local Search for Relocation solution
LSO	-	Local Search Operator
MIP	-	Mixed-Integer Programming
SAA	-	Sample Average Approximation
SE-VReP-DP	-	Stochastic Electric Vehicle Relocation Problem with Dynamic Pricing
SP	-	Stochastic Program
TS	-	Tabu Search

Chapter 1

Introduction

This thesis is based upon the same problem put forward by Eilertsen et al. (2021); therefore, some chapters may have similarities. These similarities will be described at the start of the relevant chapters to emphasize the contribution of this thesis. As there have been few changes to the relevant situation, the introduction is based on the work provided by Eilertsen et al. (2021).

Whenever a person wants to get from one location to another, different transportation alternatives are available to them. Traditionally, the person could move by, e.g., by foot, bicycle, public transport, or driving a privately owned car. The convenience of traveling by car is a cause for the increase in the number of privately owned cars (Duncan, 2011). This increase does, however, come with some implications. The dependency on private vehicles has led to congestion and poor air quality in cities due to the high pollution level (Choudhury et al., 2018). Improvements in public transport, toll fees, and other road and fuel pricing measures have not provided the desired level of sustainable solutions (Barnes et al., 2012). Car-sharing is considered a solution to several problems related to pollution (Martin and Shaheen, 2016), traffic (Crane et al., 2012), and transportation costs. Christian and Morin (2005) reveal that the average reduction in carbon dioxide emissions for members of car-sharing services lies in the range of 39% to 54%. Further, car-sharing offers economic advantages as users of car-sharing systems have the benefit of not bearing the costs of having private ownership of the car (Duncan, 2011). The traditional fossil fuel engines used in car-sharing systems are rapidly getting replaced by electric vehicles (EVs) due to the increased focus on sustainability (Cheng et al., 2019).

Car-sharing systems have experienced growth in recent years and are expected to further increase in popularity (Schiller et al., 2017). As major cities expand, the need for innovative, sustainable, and efficient transportation solutions increases and car-sharing has become a viable solution. In recent years, the complexity and size of car-sharing systems have increased, leading to the need for new and advanced solution methods for distributing cars. Today, many modes of transportation are available, contributing to an increased focus on flexibility to attract customers. The high demand for flexibility requires on-demand, short-term and one-way usage in car-sharing systems (Illgen and Höck, 2019). A car-sharing operator (CSO) is responsible for the operation of the car-sharing service. This includes distributing cars to increase availability for users and making decisions regarding the costs and revenue of the service to maximize profits.

Car-sharing systems can be divided into two main categories: *station-based* car-sharing systems and *free-floating* car-sharing systems. Traditionally, car-sharing has been station-based, *one-way* or *two-way* systems

where each car must be picked up from and delivered to specific service stations. This is usually the case for long-term car rentals, e.g., when people rent cars for holidays. A two-way system is the most limiting form of station-based systems, as the car must be picked up and returned at the same station. An important implication of this is that customers will not change the initial distribution set by the CSO. One-way systems are less restrictive, as they do not require the car to be dropped off at the pick-up location. A form of one-way system is the free-floating system. Cars in free-floating car-sharing systems can be located and dropped off by customers anywhere within a predefined operating area. One-way and free-floating systems have gained popularity in recent years due to the high customer utility and level of flexibility. These systems, however, lead to the challenge for the CSO of rebalancing the fleet of cars, i.e., relocating cars to areas where demand is high and away from areas where demand is low (Boldrini et al., 2017).

The first attempt at car-sharing was a project called Sefage in 1948 in Zürich (Shaheen et al., 2015). The project was mainly an offer to people who could not afford cars. One of the earliest car-sharing services with EVs was Praxitèle, dated 1997 in Paris. This service was a technically successful implementation of a one-way station-based system, but the high costs ended up putting an end to Praxitèle's service. A series of similar systems were created, such as Singapore-based DIRACC, a service offered by Honda. DIRACC closed in 2008 due to the non-satisfactory quality of service. Car2Go was founded in 2008 by BMW in Germany and was the first free-floating car-sharing service. Car2Go struggled with profitability and is today merged with Drive Now under the name Share Now.

Share Now is a car-sharing company operating in several countries in Europe, including Germany, Denmark, Italy, and France[1]. They offer a free-floating, on-demand service with short-term and long-term rental options and a wide selection of cars with electric and fuel engines. Depending on the rental option chosen, different fees apply. Similar to GreenMobility operating in Sweden and Denmark[2], Canadian Evo Car Share[3] and Citybee[4] operating in the Baltic countries, fees can be by the minute, hour, or day depending on the selection of short-term or long-term rental. Share Now has incorporated non-dynamic pricing strategies to incentivize customers to drop off their cars in desirable areas with high demand by including additional fees for trips to non-desirable destinations. Such destinations are, for instance, airports or areas far from the city center. They also reward customers for fueling or charging cars, similar to the Italian car-sharing service Enjoy by Eni[5]. Hyre is a new car-sharing service available in the Norwegian cities of Oslo, Bergen, and Trondheim. Customers can choose a short-term rental with an hourly fee or a long-term rental with a daily fee. Hyre includes a maximum travel distance; if the distance is exceeded, a fee per kilometer applies[6]. No companies with dynamic pricing strategies that frequently update prices to manipulate customer behavior were found in the search for related companies.

The work presented in this thesis is based on a collaboration with Vybil. Vy Group, a Norwegian state-owned company, is one of the new companies providing car-sharing services. In 2019, the company launched the EV car-sharing service Vy Bybil before soon changing the name to Vybil. The Vybil cars can be picked up and dropped off by members at any residential parking spot, public parking spot without a time limit, or at private parking stations offered by Vybil. A survey conducted by Vy Group in 2019 indicated that their car-sharing service led to 6% of 8 000 active users selling their car as a consequence of their services[7].

Decisions for the CSO can be made at the strategic, tactical, and operational level (Brandstätter et al., 2016). Strategic decisions often consider a longer time horizon and are not easily changed. The type of car-sharing system, the number of service stations or operating zones, size of the operating area, the partition of area, and capacities are considered strategic decisions. Tactical decisions include the number of employees, number

[1] https://www.share-now.com/dk/en/, accessed 2022-02-06
[2] https://www.greenmobility.com/dk/en/, accessed 2022-30-05
[3] https://evo.ca/, accessed 2022-27-05
[4] https://citybee.lt/en/, accessed 2022-30-05
[5] https://enjoy.eni.com/en/, accessed 2022-02-06
[6] https://www.hyre.no/, accessed 2022-27-05
[7] https://www.tu.no/artikler/vy-s-bildeling-har-fort-til-nesten-500-faerre-privatbiler-sa-langt/472724, accessed 2022-31-05

of cars in the vehicle fleet, reservation policy, and pricing strategy. Reservation policies include on-the-spot reservations or reservations made a particular time in advance. Operational decisions are decisions made more frequently than strategic and tactical decisions. These include decisions regarding relocations, maintenance, recharging, and refueling. The pricing decision becomes operational with a dynamic pricing strategy, where the CSO sets prices to incentivize desired customer behavior. Relocation decisions consist of which cars to relocate, how to relocate them, when to perform the relocations, and where to relocate the cars.

The relocation of cars can be divided into two main categories, namely *employee-based relocation* and *customer-based relocation*. Employee-based relocation is the case where employees of the CSO relocate cars to desired stations or areas. Employees can travel to the cars as a part of the relocation of another car, by designated transportation cars, or by using public transport or folding bicycles that fit in the trunk of the car. This form of relocation gives the CSO a high level of control over the location of cars. It allows strategic relocations to areas with high expected demand and strategic relocations from areas with low expected demand. Implications of this method include high wage costs for employees and that relocations are limited by the number of employees and their available working hours. Customer-based relocation is the case when dynamic pricing incentivizes customers to move cars to and from desired and less desired locations. Customer-based relocation allows the CSO to have fewer employees dedicated to relocation, cutting employee and transportation costs. However, an implication is the lack of control over relocations due to challenges of predicting customer behavior. These two relocation methods can be combined so that the CSO may have direct control over relocations while reducing the need for employee-based relocations with the right pricing strategy.

The CSO is met with the problem of balancing car distribution. This is traditionally solved by hiring employees to relocate cars to high-demand areas. The problem of deciding where and how to relocate the (electric) vehicles is referred to as the (Electric) Vehicle Relocation Problem ((E-)VReP). The task of rebalancing the fleet by using dynamic pricing to incentivize customers to move cars to areas with high demand, and pick up cars from areas with low demand, is called the Dynamic Pricing Problem (DPP). Finding efficient methods to operate car-sharing services is challenging, and CSOs struggle to operate profitably. Therefore, research and improvements in these methods are essential to the industry's survival.

Hellem et al. (2021) and Folkestad and Klev (2021) are among those who have proposed a solution to the problem of employee-based car-fleet rebalancing. Their studies include a mathematical model and solution method to the E-VReP and the Stochastic Electric Vehicle Relocation Problem (SE-VReP), respectively. Folkestad and Klev (2021) have modeled a multi-objective, two-stage stochastic mixed-integer programming model for a free-floating car-sharing system with electric vehicles and on-demand booking in their proposition. As opposed to Hellem et al. (2021), who propose a static optimization model used iteratively in a rolling-horizon framework, Folkestad and Klev (2021) consider the uncertainty in customer demand by making the model stochastic. Pantuso (2020) proposes a solution to the joint DPP and VReP in car-sharing services by simultaneously setting car-sharing prices and deciding on relocations while accounting for uncertainty in customer demand.

This thesis proposes and compares two solution methods to the integrated Stochastic Electric Vehicle Relocation Problem with Dynamic Pricing (SE-VReP-DP). The first method, originally formulated by Eilertsen et al. (2021), is a single-objective, two-stage stochastic mixed-integer programming model for free-floating car-sharing systems with electric vehicles and on-demand booking. Finding a pricing strategy that sufficiently encourages customers to rebalance the fleet is crucial for the CSO to save costs related to employee-based relocation and satisfy demand. To better represent the solution methods' intended use, throughout this thesis, we will refer to the pricing strategy as dynamic pricing, despite it not being dynamic when solved only once. For real-life applications, the problem is meant to be solved over planning horizons in sequence, meaning that the prices can be changed each time the problem is solved, based on the current situation of the car-sharing system. The model should make good relocation and pricing decisions, considering different possible demand outcomes. The possible outcomes introduce uncertainty, making the model more realistic

than a deterministic model. The stochasticity in demand stems from differences in customer preferences and uncertainty in how customers react to the locations of cars and the prices for their intended trips. In addition to handling the uncertain demand, the model makes sure cars in need of charging are moved to charging stations, either by customers or employees. As opposed to Pantuso (2020), who simplified the employee-based relocation by not including the origin of cars, travel times, and employees used to relocate the cars, we have proposed a model that includes the relocation decisions on a more detailed level. By this, the model proposed in this thesis provides solutions to employee-based relocation problem and dynamic pricing problem.

The main contributions of this thesis are a solution method for solving real-sized instances of the SE-VReP-DP, and a rolling horizon framework to properly analyze the effects of dynamic pricing. The MIP model formulated in Eilertsen et al. (2021) could not solve instances of realistic size within a reasonable time, which makes the model poorly suited for practical and frequent use by a CSO. In order to mitigate this problem, a heuristic solution method, partly based on the work of Ropke and Pisinger (2006) and Folkestad and Klev (2021), is proposed. There are, however, many differences in implementation. Our solution method consists of an Adaptive Large Neighborhood Search (ALNS) integrated with two Local Search (LS) heuristics. This integration makes it possible to better gain managerial insights for the CSO, as good solutions can be found within a reasonable time. The heuristic proposed in this thesis differs from heuristic approaches in the literature, as it is adapted to a stochastic setting and includes dynamic pricing. Further, the problem is solved in a rolling horizon framework, simulating a complete work day for a CSO. This allows for more comprehensive analyses concerning the long-term effects of using dynamic pricing.

Chapter 2

Problem Description

This chapter presents the integrated Stochastic Electric Vehicle Relocation Problem with Dynamic Pricing (SE-VReP-DP). This problem is the same as presented by Eilertsen et al. (2021), which is a combination of the Stochastic Electric Vehicle Relocation Problem (SE-VReP), introduced by Folkestad and Klev (2021), and the Dynamic Pricing Problem (DPP), described by Pantuso (2020).

The integrated SE-VReP-DP is an operational planning problem for a free-floating car-sharing system operated by a Car-Sharing Operator (CSO). The CSO operates with a fleet of Electric Vehicles (EV) and is equipped with a staff of employees. The EVs, as well as several charging stations, are located within an operating area. The CSO wishes to serve as many customers as possible during operating hours, out-competing alternative transportation modes in fulfilling the customers' travel requests. This is mainly achieved by assuring the availability of EVs in terms of location and battery level and by applying a suitable pricing strategy.

For a given planning horizon, the CSO makes decisions concerning the relocation and pricing of EVs. The CSO can use its employees to move EVs to locations where higher demand is expected to better meet customer demand. This requires relocation routing and scheduling for each employee, which is a problem similar to the well-known E-VReP. Further, the CSO can use dynamic pricing to manipulate customer demand. By determining vehicle-specific rental fees, the CSO can make certain EVs more (or less) attractive to customers without the need for employee-based relocation. This requires regular decisions of fees for each EV through a pricing strategy, resulting in a DPP. The pricing strategy entails fees for both the pick-up of each EV and the drop-off at each possible destination. The fees will further be applied to incentivize customers to move EVs in need of charging to charging stations.

Based on the abovementioned decisions, the CSO will experience revenue from each satisfied customer. However, the decisions will bear costs in terms of relocation costs and negative fees. The objective is to maximize the expected profit given the expected revenue and the expected costs. A solution to the problem is thus a list of employee routes and schedules, and a pricing strategy for each EV, maximizing the objective.

There are, however, several aspects complicating the problem. Firstly, the CSO must ensure that a certain number of EVs are sufficiently charged. This may require using employees to relocate EVs to charging stations. Secondly, the temporal aspect has a decisive effect on the problem, as the length of the planning horizon may complicate the employee routing and scheduling. Finally, and most importantly, the customer

demand and behavior are uncertain and unknown to the CSO. Information such as expectations concerning demand density, price sensitivity, and customer preferences are available to the CSO. However, customers act in unexpected manners, challenging to fully foresee. Under identical circumstances, the same customer may act differently based on information only known to the customer itself. This introduces stochasticity to the problem, making it a Stochastic Electric Vehicle Relocation Problem with Dynamic Pricing.

Chapter 3

Literature Review

This chapter introduces literature within the formulation of and solution methods to challenges in car-sharing systems and is partly based on the literature review presented by Eilertsen et al. (2021). By combining well-known research studies with more recent ones, we provide insight into the modeling choices made in this report.

Section 3.1 gives an overview of the possible levels of planning for decisions made in car-sharing services. Section 3.2 investigates literature concerning relocation challenges in car-sharing systems. Section 3.3 reviews research within revenue management in general and how this is applied in solution methods to relocation problems. Section 3.3 also discusses how modeling customer behavior is a challenging and important aspect when investigating dynamic pricing strategies. Section 3.4 presents literature related to heuristic solution methods for solving car-sharing problems. Finally, Section 3.5 provides a summary and a visual overview of the presented literature.

The literature review conducted by Folkestad and Klev (2021) has been used as foundation for Sections 3.1 and 3.2. Additional research was found using the academic search engine Google Scholar. Table 3.1 lists the combination of terms used when searching for additional literature.

3.1 Planning Levels in Car-Sharing Services

There are three levels of planning considered relevant for car-sharing operators in the literature, namely strategic, tactical and operational. Interdependencies may exist between the different levels. However, decisions are usually isolated to one level by assuming the decisions on other levels to be fixed. The model presented in this paper primarily considers decisions at the operational level. However, an introduction to the strategic and tactical level provides necessary background information. Figure 3.1 summarizes the decisions that belong to each of the planning levels and their interdependencies, presented in Illgen and Höck (2019) and Hellem et al. (2021).

Table 3.1: Search terms for related literature review.

Search	Terms
1	car-sharing + relocation
2	revenue management + car-sharing + customer behavior
3	revenue management + car-sharing + customer utility + relocation
4	car-sharing + customer utility + dynamic pricing + relocation
5	car-sharing + customer utility + dynamic pricing + relocation + stochastic
6	car-sharing + relocation + revenue management + stochastic demand
7	car-sharing + fleet size + vehicle distribution
8	car-sharing + optimal station locations + number of stations
9	car-sharing + reservation policy + pricing scheme
10	car-sharing + network mode + station-based + free-floating
11	car-sharing + heuristics + neighborhood search
12	car-sharing + heuristics + ALNS + tabu search
13	car-sharing + heuristics + evolutionary algorithm + genetic algorithm

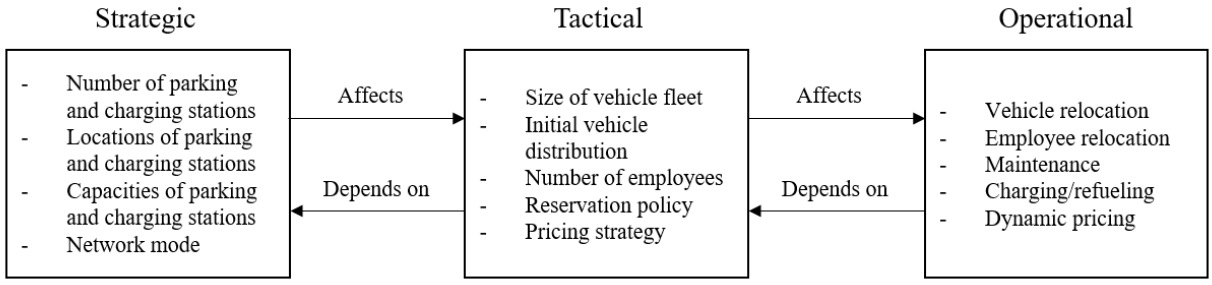


Figure 3.1: Decisions made at each planning level and interdependencies between the different levels.

Decisions regarding the number, locations, and capacities of parking and charging stations and decisions concerning the network mode are strategic. Changing the system's infrastructure is costly, so decisions on the strategic level are made less frequently than tactical and operational decisions. The network mode includes two-way, one-way, and free-floating systems. Two-way systems are the least flexible, as customers must return the cars to the pick-up location. One-way systems have increased flexibility, as the customers can leave the cars at any service station. Alfian et al. (2014) evaluate the performance of two-way and one-way car-sharing systems in a simulation model. The results reveal that the added flexibility of one-way systems, as opposed to two-way systems, increases the number of utilized vehicles and leads to higher profits for the CSO. In order to mitigate the fleet imbalance that may arise in one-way systems, Correia and Antunes (2012) evaluate historical data from Lisbon to find optimal station locations. For free-floating systems, this could be done by analyzing spatial demand patterns in the operating area. A possible approach to this is to perform a cluster analysis by classifying zones with high and low demand as hot and cold spots, respectively (Weikl and Bogenberger, 2013). Another approach is to apply a machine learning method found in more recent research, called the kernel density estimation, in the demand estimation (Li et al., 2020). Kühne et al. (2016) propose a

Mixed-Integer Programming (MIP) model to find the optimal number of, and locations of, charging stations for car-sharing services with Electric Vehicles (EVs). They argue that more charging stations are needed with longer expected travel times due to the increase in vehicles needed.

Decisions regarding the size of the vehicle fleet and employee base are examples of tactical decisions. Nourinejad and Roorda (2014) argue that a larger fleet reduces the need for employee-based relocations. By considering request patterns and user demand, Nourinejad and Roorda (2014) aim to optimize fleet size. Tactical decisions also include initial vehicle distribution and whether or not to allow vehicle reservations made in advance. The study proposes that allowing a 30 minute reservation time can reduce the fleet size of up to 86% for CSOs with an on-demand reservation policy. This proposition is based on running an operational VReP in an optimization-simulation model. However, Hu and Liu (2016) argue that the results obtained are too optimistic when ignoring the loss in efficiency from making cars unavailable when reserved. Related to this, Huang et al. (2020) propose a model for finding the optimal electric vehicle fleet size in a one-way station-based system. The charging station capacities and fleet size are iteratively updated and included in a rolling horizon framework to optimize the operational decisions over a day. Deciding on the pricing scheme can be considered another tactical decision. Waserhole and Jost (2016) propose using dynamic pricing, i.e., using prices to incentivize customers, by including either a fee or a bonus, to relocate cars to areas with high demand and away from areas with low demand. The objective is to reduce the need for employee-based relocation.

Operational planning decisions include relocation, maintenance, and charging or refueling of vehicles. It also includes the pricing strategies used by the CSO. Decisions on the strategic and tactical levels are input to the operational level. Decisions on the operational level are meant to maximize the potential of the car-sharing system on a daily or more frequent basis. This paper primarily concerns decisions on the operational planning level. Therefore, challenges related to the operational planning level is the main topic explored in the remainder of this chapter. Some strategic and tactical decisions were included to create an overview of the situation at hand.

3.2 Car-Sharing Relocation

The rebalancing of cars in a car-sharing fleet is one of the main challenges to a CSO. Being a well-known problem, the research within the field is quite extensive. Most of the earliest research concentrate on station-based car-sharing systems. Barth et al. (2004) introduce a trip joining and splitting method for one-way car-sharing systems. The research objective is to investigate the possibility of reducing the number of relocations performed by employees without lowering the Levels of Service (LOS). They incentivize customers to share a car if the station has too few cars (trip joining) and use one car each if the station has an abundance of cars (trip splitting). Their study shows an impressive reduction in employee-based relocations needed to maintain the same LOS.

While Barth et al. (2004) aim to partially replace employee-based relocations, Kek et al. (2009) and Nourinejad et al. (2015) aim to optimize the relocation process itself. Proposing a two-stage MIP model, Kek et al. (2009) minimize costs related to employee-based relocations and low LOS at the car stations. The first stage is an optimizer, while the second stage includes several heuristics determining operating parameters based on the results from the optimizer. These parameters include the length of employee shifts and the choice of relocation techniques. The research provides good results, but lacks an important practical aspect - the rebalancing of employees. Nourinejad et al. (2015) approach this aspect in an attempted joint optimization of car relocation and employee rebalancing. However, their approach is more on a tactical level, giving support to making decisions concerning the size of the car fleet or the number of employees through sensitivity analyses. Nevertheless, the two studies jointly conclude that the optimization of employee-based relocations is of great

value. Nourinejad et al. (2015) propose that both the number of stations and the number of employees can be reduced without lowering the LOS, while Kek et al. (2009) state that the choice of relocation technique may reduce costs while maintaining the same LOS.

The more recent demand for free-floating systems has caused challenging relocation issues. Weikl and Bogenberger (2013) address these issues by developing algorithms for predicting demand. They partition the operating area into smaller zones, simulating the stations in an otherwise station-based system. Using historical demand data adjusted with real-time demand measurements, they establish an ideal number of cars in each zone. A model to optimize the relocations given the ideal number of cars is presented but not formulated mathematically. The ideal state approach is, however, extended by Hellem et al. (2021) to include a mathematical formulation. Their study proposes a MIP model to solve the Dynamic Electric Vehicle Relocation Problem using a rolling horizon framework. The complexity of the model required the use of heuristic solution methods to find near-optimal solutions for larger instances, which they achieved with an Adaptive Large Neighborhood Search (ALNS) heuristic. Hellem et al. (2021) combine several elements that have been central in car-sharing relocation research the past two decades into one joint model. The study includes aspects from Kek et al. (2009), Nourinejad et al. (2015) and Weikl and Bogenberger (2013) regarding minimizing relocation costs, rebalancing of employees, and meeting expected demand through ideal states, respectively. The model results in schedules and routes for each employee, defining the optimal relocations of both cars and employees for the proposed ideal states.

In most previous studies on employee-based relocation in car-sharing systems, the demand within the planning horizon has been considered known. However, in reality, this demand is stochastic. The uncertainty in demand is often handled using predictive models, reducing the stochastic situation to a deterministic one. This is done by e.g. Weikl and Bogenberger (2013) and Hellem et al. (2021). Another approach is using probability distributions of demand, where customers are looking for a car given a certain probability, as do Brandstätter et al. (2017) and Biondi et al. (2016). The former study proposes a two-stage MIP model to make strategic decisions. The latter study considers a station-based car-sharing system where the objective is to improve the distribution of stations. Huo et al. (2020) propose a data driven approach to characterize vehicle dynamics and customer behaviour in electric car-sharing systems. The results from analyzing a large amount of historical data reveal that the customer behaviour follows a Poisson distribution. The electric car-sharing system relocation problem is then formulated as a data-driven optimization model, combining probability expectation modeling and linear programming with real-time data as input. Li et al. (2019) takes the probability concept further, combining machine learning techniques to construct a spatial demand distribution. The relocation problem is solved daily with a two-stage stochastic programming model. However, as a continuous distribution yields infinitely many scenarios, they use a solution method proposed by Santoso et al. (2005), integrating Bender's decomposition and Sample Average Approximation (SAA). Fan (2013) takes a similar approach, only extended to include relocation routes for the employees. The study formulates a multi-stage model with recourse for car-sharing systems, where small parts of the demand are revealed in each stage. The objective is to decide the relocation strategy, where customer routes are known a priori, and customer reservations are made a day in advance.

None of the studies described in the above paragraph consider free-floating car-sharing systems under uncertainty. Folkestad and Klev (2021) introduce uncertain demand in a free-floating car-sharing system with EVs. They optimize employee routes and schedules based on expected profit under different demand scenarios. The concept of ideal states is used, though the states are made unavailable to the CSO at the time of making relocation decisions, providing a more realistic situation. A similar approach was taken by Nair and Miller-Hooks (2011). They introduce a MIP model aiming at correcting short-term demand asymmetry. They use stochastic demand to provide relocation strategies for circumstances when demand outstrips supply. The model further includes chance constraints for finding optimal initial vehicle distributions, reducing the need for relocations later in the modeling process. The relocation strategies do, however, not include routing and scheduling of employees, as do Folkestad and Klev (2021).

Another critical aspect of the relocation problem is the recharging of EVs. Many CSOs today use EVs in their fleet, where employees perform the charging as a part of the relocations. Boyacı et al. (2015) introduce a multi-objective MIP model for relocation of EVs in a one-way car-sharing system. They find the efficient frontier of maximizing CSO profit and customer utilities using the weighted sum model. Boyacı et al. (2017) further extend the model to include rebalancing of employees using a lexicographical approach. Bruglieri et al. (2014) propose that employees use folding bicycles when moving between cars to increase efficiency, while Boyacı et al. (2017) use service cars to move employees in larger groups. Xu and Meng (2019) address another possible issue that may arise when cars need to be charged. They consider that the time it takes to recharge car batteries is a non-linear process when determining the vehicle fleet size. Cepolina and Farina (2012) and George and Xia (2011) propose methods for improving already existing car-sharing systems by looking at the cost of waiting for cars to charge while considering imbalances in demand. Folkestad et al. (2019) introduce a MIP model specifically for optimal recharging and relocation of EVs in a free-floating car-sharing system. The model does not necessarily relocate cars needing charging to the nearest charging stations but strategically selects charging stations to meet expected demand. The demand is depicted through ideal states, as done by Hellem et al. (2021). Hellem et al. (2021) also include logic for handling charging of the fleet. They approach the problem by rewarding relocation of cars in need of charging to charging stations. However, this approach requires measures of the long-term benefits from charging, which is challenging to estimate as the supply of cars affects both the short-term and long-term demand. Folkestad and Klev (2021) remove the charging benefit estimation from the objective function, implementing a multi-objective model following the lexicographic method. The model was first solved to optimize the number of cars to charge and then to maximize the profit.

3.3 Revenue Management

Another interesting approach to the relocation problem is using revenue management. McGill and Van Ryzin (1999) defines revenue management (also known as *yield management*) as the "practice of controlling the availability and/or pricing of travel seats in different booking classes with the goal of maximizing expected revenues or profits." Since 1999 this term has been used more widely, and is now in use in many businesses, e.g., hotels, car-rental, car-sharing and manufacturing (Klein et al., 2020). One of the key areas within revenue management is pricing strategies.

McGill and Van Ryzin (1999) present a research overview on revenue management with references to more than 190 different studies. They describe how the airline market was the first market where revenue management gained attention, particularly by using seat allocation. Although this seating strategy is the most visible today in the airline market, the use of dynamic pricing has proven critical for the airlines to maximize their profit. Selcuk and Avşar (2019) work on this problem with the motivation of determining whether dynamic pricing can increase profit. They use disaggregated demand at individual customer levels to evaluate how the customers would react to different prices of fares, and the model allows different reservation prices for different customers.

Revenue management has been a research topic within the car-sharing business since about 2013, where Waserhole et al. (2013) was one of the first to try to balance the fleet of cars in a car-sharing system using pricing incentives. Comparing the situation to a bicycle-sharing system, the authors made a model for a station-based one-way car-sharing system to maximize the number of trips sold. A customer is offered a price determined by a Markov Decision Process, handling the demand stochasticity. The model is solved using different heuristics due to the complexity of the problem, producing results with significant gaps in the upper and lower solution bounds.

Most car-sharing services operate with a First-Come-First-Served (FCFS) mechanism. Wang and Liao (2021)

argue that the FCFS mechanism significantly influences the supply-demand dynamics in the car-sharing system. They discuss related mechanisms and analyze the impact these have on the use of shared vehicles and thus the profit of the CSO. They model the supply-demand interaction in the situation of supply insufficiency, proposing mechanisms for queuing to utilize the service more efficiently. The analysis considers supply-demand dynamics under different FCFS mechanisms and embeds these in a Boundedly Rational Dynamic User Equilibrium Problem. Wang and Liao (2021) cope with different possible waiting times by proposing a path expansion strategy to accurately capture the choice of car-sharing service in space and time. They solve this by column generation in a bi-modal supernetwork.

Jorge et al. (2015) address the problem of an unbalanced car fleet in a one-way station-based car-sharing network. The authors implement a mixed-integer non-linear programming model where the prices between pairs of zones are set to optimize the profit. It is non-linear as the demand depends on the price, leading the authors to use an Iterated Local Search (ILS) metaheuristic to solve the model. Their case study in Lisbon found that using the ILS to solve the model had a substantial positive impact on the profit. Even though the optimal pricing strategy led to 18% less demand served and resulted in an unbalanced fleet, the profit was substantially better than when no price-based balancing strategy was applied. Kamatani et al. (2019) address the same problem as Jorge et al. (2015). They distinguish themselves by making decisions based on the machine learning method called Reinforcement Learning (RL), suitable for multi-step decision-making. A customer has different preferences regarding price and distance. Using these preferences and making the RL model learn in the environment changes, the authors find that the number of zones with no cars was reduced.

Avenalia et al. (2019) try to solve an extended version of the problem addressed by Jorge et al. (2015) and Kamatani et al. (2019), by also considering different relocation strategies. The relocation strategies are *uniform* (equal number of cars in each zone), *threshold* (number of cars in a zone in a threshold interval) and *free* (unregulated number of cars in each zone). The prices are set for each origin-destination zone pair, decided by the availability of cars in the zones. The problem is formulated as a mixed-integer linear programming model to determine the optimal pricing discrimination under different relocation strategies. The study found that the less we influence the system's structure, the higher the profit. Price discrimination was in all cases found to be better than a fixed price strategy. Kikuchi and Miwa (2021) also argue that dynamic pricing models yield higher profit than models with fixed pricing. They consider the effect of changing the prices based on demand at given stations in one-way and round-trip station-based systems by formulating an optimization problem to determine the price between any two stations at each time slot. For smaller instances, results are obtained by solving a mixed-integer non-linear programming model. For larger instances, the problem is solved by implementing a network flow problem where integer constraints are relaxed. The method is evaluated by performing numerical experiments for the network flow problem on a time and space network.

When modeling dynamic pricing strategies to increase profit, realistic customer behavior is challenging to imitate. de Donnea (1972) discusses how customer behavior usually was modeled in microeconomics through one constraint only, namely the budget constraint. This modeling indicates that whether a customer buys a good or not depends solely on the price of the good. However, when it comes to services within transportation, de Donnea (1972) argues that time is an essential factor in addition to the price. Therefore, he introduces a utility function taking both time and price as arguments to model customer behavior. Modesti and Sciomachen (1998) take the concept of utility measurement a step further. Their research describes a set of utility weights assigned to each customer. The weights represent the customer's preferences concerning specific characteristics of a trip between two geographical points by a specific mode of transport. For instance, a trip by bus and a trip by car may have different comfort levels and prices and may vary in travel time. With their respective weights, these characteristics are given as input to a utility function, providing an estimated utility of each mode of transportation for a customer.

Another approach to modeling customer behavior is using reinforcement learning, as do Esmaeilpour et al.

(2016). They try to model whether or not a customer would purchase a specific good given its price, degree of luxury, and availability. Having access to an extensive database containing customer-specific information, they train customer agents using Q-learning. The results support the possibility of modeling customer behavior through reinforcement learning. However, the accuracy depends on the amount and specification of customer data.

Kamatani et al. (2019) use the concept introduced by Modesti and Sciomachen (1998), using only two weights to balance the trade-off between rental charge and walking distance. Pantuso (2020) applies a more extensive version, including weights for walking distances and waiting time. Like Avenalia et al. (2019), Pantuso focuses on the use of dynamic pricing on car-sharing trips between pairs of geographical zones. However, this is a study on a free-floating car-sharing system, as opposed to the one-way system studied by Avenalia et al. (2019). Furthermore, Pantuso (2020) includes simple relocation logic in the model, making the research more realistic and more complex. Decisions concerning which cars to move where and what prices to apply at which travel routes gave promising results on his case study in Milan.

3.4 Heuristics Relevant to Car-Sharing

Finding exact solutions to real-sized instances of the SE-VReP-DP has proven to be very time-consuming (Eilertsen et al., 2021). On a practical level, this problem faces time constraints as it is meant to be solved frequently throughout the day. In order to find good solutions within a reasonable time, different heuristic approaches have been explored. Several heuristic methods for car-sharing problems have been proposed in the literature. Neighborhood Search and Evolution-Inspired Search arise as the two most popular categories for these kinds of problems. Sections 3.4.1 and 3.4.2 present literature on Neighborhood Search heuristics and Evolution-Inspired Search heuristics relevant to car-sharing problems, respectively.

3.4.1 Neighborhood Search

The two most common neighborhood search methods found in the literature for car-sharing problems are Tabu Search (TS) and Adaptive Large Neighborhood Search (ALNS). Local Search (LS) is also a common neighborhood search method. However, LS is often used in combination with either ALNS or TS.

Tabu Search

TS is a local neighborhood search, meaning it only explores local neighbor solutions. TS greedily chooses solutions that yield the highest increase in objective value, similar to LS. However, to avoid getting stuck in a local optimum, TS stores a predefined number of previous solutions that are not allowed to be revisited for a given number of iterations. This is referred to as the *tabu tenure*.

Bruglieri et al. (2019) compares solutions from a TS and an ALNS to the E-VReP in a one-way electrical car-sharing system with the exact solution from mixed-integer linear programming. They tested their approach on three real-like sets of test instances. Similarly, Amine Ait-Ouahmed (2018) proposes a TS for finding solutions to E-VReP in a one-way electrical car-sharing system. Unlike Bruglieri et al. (2019), where the TS is performed on a complete solution constructed greedily, Amine Ait-Ouahmed (2018) divides the problem into two separate sub-problems, namely car routing, and employee routing. The car routing problem is solved first, and the employee routing is solved based on the solution from the car routing problem. Numerical results show that the TS can produce near-optimal solutions with decreased computational time for larger

test instances.

Adaptive Large Neighborhood Search

The general Large Neighborhood Search (LNS) is introduced by Shaw (1998), where so-called *destroy* and *repair* operators are used to break down and rebuild the solutions. This is to explore a larger share of the search space than an LS, where only marginal changes to the solution are performed within each iteration. ALNS was first introduced by Ropke and Pisinger (2006), and it is based on the LNS from Shaw (1998). In an ALNS, weights are included for each destroy and repair operator, rewarding operators that have produced good solutions in previous iterations. For the subsequent iterations, the choice of an operator is based on their weights instead of chosen at random, as done by LNS. This constitutes the adaptive part of the ALNS.

The ALNS is a widely used heuristic for car-sharing problems. As mentioned above, Bruglieri et al. (2019) propose an ALNS for solving an E-VReP in a one-way electrical car-sharing system. The ALNS proved to be an effective heuristic for solving the E-VReP, and performed better than the TS. Hellem et al. (2021) also propose using ALNS for solving a car-sharing problem and reach the conclusion that the ALNS produces solutions of high quality on real-sized instances within a reasonable time. There is also some research on combining different heuristic methods to solve variations of the VReP. Cai et al. (2022) propose a hybrid ALNS and TS algorithm for the E-VReP where time windows, limited durations, and charging requirements are considered. Computational experiments show the effectiveness of the proposed method in solving the E-VReP.

ALNS in Stochastic Formulations

In the literature, few cases of heuristics are implemented in the stochastic formulation of the VRP and VReP. One of the studies on the topic is done by Lei et al. (2011), where the capacitated vehicle routing problem with stochastic demands and time windows (CVRPSDTW) is considered. Lei et al. model this as a stochastic program with recourse, with demand as the stochastic factor. By using a recourse policy, the second-stage solution of the problem is the actions to take in different outcomes of the stochastic demand. The objective is to minimize the cost of the first-stage relocations and the expected cost of the second-stage solution, taking into account the different possible demand realizations. The probability of each second-stage outcome is multiplied with the cost of their respective recourse actions. In the paper, an ALNS is proposed as a solution method. Similarly, Luo and Lim (2016) propose a strategy implementing three ALNS heuristics to solve a VRP with stochastic demands and weight-related cost. The problems and solution methods in these papers differ from our problem and strategy as we are considering profit maximization in the second stage. Folkestad and Klev (2021) propose an ALNS adapted to a stochastic setting to solve a Stochastic Electric Car Relocation Problem (SE-CReP). As a part of their ALNS implementation, LS is included to explore the local neighborhood to see if that can improve the solution further. The work by Folkestad and Klev (2021) differs from this thesis, where finding the optimal pricing policy is the main objective for the model and the proposed heuristic.

Combined Heuristics and Machine Learning Methods for Car-Sharing Problems

Recent studies within using machine learning methods combined with existing optimization techniques and heuristics show promising results. Bogyrbayeva et al. (2021) propose a reinforcement learning approach for rebalancing free-floating electrical car-sharing systems. The objective is to rebalance the fleet of cars in a minimal amount of time. As a solution method, they deploy a policy gradient method for training recurrent neural networks and compare the obtained policy results with heuristic solutions. The learned policies offer flexibility, leading to results significantly reducing the time needed to rebalance the network. Hottung and Tierney (2019) propose a Large Neighborhood Search (LNS) that integrates trained heuristics for solving vehicle routing problems. The learning mechanism for generating new solutions is based on a deep neural

network with an attention mechanism, where the repair operator is chosen based on the characteristics of the instance and the chosen destroy operator. Results from testing on the capacitated vehicle routing problem (CVRP) and the split delivery vehicle routing problem (SDVRP) show that the approach outcompetes other existing machine learning approaches and comes close to the performance of state-of-the-art optimization methods.

Pricing Policy Heuristics

Research on and development of heuristics for dynamic pricing in car-sharing systems are limited. Waserhole et al. (2013) propose using pricing policies to improve the efficiency of car-sharing systems modeled as a Markovian formulation of a closed queuing network with finite buffer and time-dependent service time. A heuristic approach is taken as the model is intractable for real-sized instances. This approach combines scenarios, fluid approximation, simplified stochastic models, and asymptotic approximations. Similarly, Waserhole and Jost (2016) aim to optimize the number of trips taken in a one-way car-sharing system by using prices to incentivize customers. The car-sharing system is modeled as a closed queuing network with an infinite buffer capacity and Markovian demands. The paper proposes a heuristic based on computing a maximum circulation on the demand graph and a convex integer program solved optimally by a greedy algorithm. The heuristic approaches found in the literature differ from the approach taken in this thesis, as we are considering dynamic, not static, pricing, and we are using a combination of LS and ALNS instead of a maximum flow relaxation of the problem. Literature specifically related to neighborhood search for car-sharing systems is limited, so the approach taken in this thesis is an adaptation and combination of research within ALNS for relatable problems.

3.4.2 Evolution-Inspired Search

Genetic algorithms are based on a natural selection process that mimics biological evolution and is a part of the larger class of evolutionary algorithms. In a genetic algorithm, a set of candidate solutions, called the population, evolves toward better solutions by mutating or altering a set of properties to fit the candidate solution. The fitness of the solutions is calculated and the good candidate solutions are used for the next iteration of the algorithm. Prince (2004) is the first to present an evolutionary algorithm for VRPs that could compete with TS. Tan et al. (2007) study a multi-model type of the Capacitated Vehicle Routing Problem with Stochastic Demand (CVRPSD), and presents a multi-objective evolutionary algorithm that incorporates two VRPSD-specific heuristics for local exploration. A route simulation method is used to evaluate the fitness of the solutions. Santos et al. (2017) formulate an Integer Linear Programming (ILP) model and create an evolutionary algorithm (EA) in order to solve a Vehicle Relocation Problem in a free-floating car-sharing system. The system was based around the city of Cortagem in Brazil. Results revealed that the EA performed well compared to the ILP and found the optimal solution in some cases. However, the computation times were high for the EA, making it unsuitable for practical use. Similarly, Folkestad et al. (2019) consider the problem of charging and repositioning a fleet of EVs in a free-floating car-sharing system. They developed a Hybrid Genetic Search with Adaptive Diversity Control to solve the problem. In contrast to Santos et al. (2017), the genetic algorithm generates high-quality solutions within a reasonable time for problem instances of realistic size.

3.5 Summary

The research on both car-sharing fleet relocation and revenue management is extensive. The intentions of Chapter 3 were to give insight into previous studies and help clarify the choices made further in this thesis.

Table 3.2 summarizes literature related to decisions at the strategic and tactical planning level and literature relevant to the context of the problem at hand. Table 3.3 provides an overview of the more directly related papers discussed in Sections 3.2 and 3.3, and at which topics they coincide with this report. Table 3.4 summarizes literature related to heuristics for car-sharing systems and pricing policies, discussed in Section 3.4.

Most of the research studies discussed propose solution methods to problems related to the relocation of cars and pricing strategies for CSOs. Most relocation-focused studies treat demand as a deterministic factor, while most pricing-focused studies ignore the routing and scheduling details behind relocation. Folkestad and Klev (2021) study the relocation problem at a high level of detail while using stochastic demand. Pantuso (2020) formulates an extensive stochastic dynamic pricing model while ignoring the details of employee-based relocations. As mentioned in Chapter 1, this report draws distinguished lines to these two studies. By combining and extending important aspects from both studies, we aim to maintain high levels of detail within solution methods for both employee-based relocations and dynamic pricing, including the uncertainty in demand. To our knowledge, there are no other studies on integrated employee-based relocation and dynamic pricing in car-sharing systems with a matching level of details. Moreover, an integrated ALNS and LS heuristic is implemented to be able to solve real-sized instances of the problem. The heuristic approach proposed in this thesis differs from the approaches found in the literature, including Hellem et al. (2021) and Folkestad and Klev (2021), as they only consider employee-based relocation and do not perform a search for the pricing policy. The heuristic is tested on real-sized instances based on data from the Norwegian car-sharing company Vybil, and is used to provide managerial insight in combination with a rolling horizon framework.

Table 3.2: Overview of articles related to decisions on the strategic and tactical planning levels, discussed in Section 3.1.

Research paper	**EVs**	**Station-based systems**	**Free-floating system**	**Station location**	**Charging station locations**	**Fleet size**	**Reservation time**	**Dynamic pricing**
Alfian et al. (2014)		✓				✓		
Correia and Antunes (2012)		✓		✓	✓			
Weikl and Bogenberger (2013)			✓					
Li et al. (2020)			✓					
Kühne et al. (2016)					✓			
Nourinejad and Roorda (2014)						✓	✓	
Hu and Liu (2016)						✓	✓	
Waserhole and Jost (2016)								✓
Xu and Meng (2019)						✓		
Cepolina and Farina (2012)	✓					✓		
George and Xia (2011)	✓					✓		

Table 3.3: Overview of articles discussed in Sections 3.2 and 3.3, and how they coincide with the topics presented in this thesis.

Research paper	**EVs**	**Free-floating system**	**Employee-based relocation**	**Employee routing**	**Customer-based relocation**	**Dynamic pricing**	**Stochastic model**	**Customer behavior**
Kek et al. (2009)			✓					
Nourinejad et al. (2015)			✓	✓				
Weikl and Bogenberger (2013)		✓	✓					
Folkestad et al. (2019)	✓	✓	✓	✓				
Hellem et al. (2021)	✓	✓	✓	✓				
Huo et al. (2020)	✓		✓				✓	✓
Wang and Liao (2021)							✓	✓
Kikuchi and Miwa (2021)						✓	✓	✓
Barth et al. (2004)					✓			✓
Selcuk and Avşar (2019)						✓		✓
Esmaeilpour et al. (2016)							✓	✓
Jorge et al. (2015)					✓	✓	✓	
Modesti and Sciomachen (1998)						✓	✓	✓
Waserhole et al. (2013)					✓	✓	✓	✓
Kamatani et al. (2019)					✓	✓	✓	✓
Avenalia et al. (2019)			✓		✓	✓		✓
Nair and Miller-Hooks (2011)	✓	✓	✓				✓	
Folkestad and Klev (2021)	✓	✓	✓	✓	✓		✓	
Pantuso (2020)	✓	✓	✓		✓	✓	✓	✓
This report	✓	✓	✓	✓	✓	✓	✓	✓

Table 3.4: Overview of articles related to heuristics for car-sharing systems discussed in Section 3.4

Research paper	(A) LNS	TS	EA	Other Heuristics	VRP/ VReP	EVs	Stochastic Demands	Pricing Policy
Bruglieri et al. (2019)	✓	✓			✓	✓		
Cai et al. (2022)	✓	✓			✓	✓		
Amine Ait-Ouahmed (2018)		✓			✓	✓		
Shaw (1998)	✓							
Ropke and Pisinger (2006)	✓							
Hellem et al. (2021)	✓				✓			
Folkestad and Klev (2021)	✓				✓	✓	✓	
Lei et al. (2011)	✓				✓		✓	
Hottung and Tierney (2019)	✓			✓	✓	✓		
Bogyrbayeva et al. (2021)				✓	✓	✓		
Luo and Lim (2016)	✓				✓		✓	
Waserhole et al. (2013)				✓	✓			✓
Waserhole and Jost (2016)				✓	✓			✓
Prince (2004)			✓		✓			
Tan et al. (2007)			✓		✓		✓	
Santos et al. (2017)			✓		✓		✓	
Folkestad et al. (2019)			✓		✓	✓	✓	

Chapter 4

Model Formulation

This chapter provides a detailed mathematical formulation of the SE-VReP-DP. The model is a modified version of the formulation of Eilertsen et al. (2021), and includes important aspects from both Folkestad and Klev (2021) and Pantuso (2020), concerning the routing problem and the pricing problem, respectively. Section 4.1 presents the reasoning behind our modeling approach. General assumptions are enlightened in Section 4.2, before Section 4.3 introduces the notation used. The mathematical formulation is explained in detail in Section 4.4. A visual example problem with a step-by-step solution method is provided in Appendix C.

4.1 Modeling Approach

We approach the problem by implementing a two-stage linear stochastic mixed-integer programming model. Both the choice of which vehicles to relocate by which employees and the choice of prices are first-stage decisions. The relocations take place in the first stage, while the pricing decisions take effect in the second stage, when the customers are choosing the transportation mode. The choice of prices is denoted as the CSO's pricing strategy. First-stage decisions are made before the CSO knows the customer preferences concerning different attributes of the car-sharing service, such as distance to the cars and rental price. In the second stage, customer preferences manifest themselves through scenarios, and customer behavior and choice of transportation mode are realized. We are approximating the stochastic element of customer behavior by using independent and identically distributed samples from the original distribution. This approximation method is called Sample Average Approximation (SAA). The method produces an expected revenue, resulting in the CSO's expected profit when taking the first-stage decisions into account. The goal is to maximize the expected profit while considering the cost of relocating cars and the expected revenue made by the pricing strategy.

The first stage starts at time T_0, the beginning of the planning horizon. Here, the decisions are to determine which cars to be relocated and which employees to relocate them. Folkestad and Klev (2021) argue that a so-called *task-based* model outperforms the *flow-based* approach in both solution quality and computation time. We, therefore, pursue the task-based approach, referring to the relocation of a car as a *car-move*, where one car-move is defined as a *task* when assigned to an employee. Each employee is given a list of tasks to complete within the first stage of the planning horizon. The task list contains information regarding both

the route and schedule for each employee. The timeline of input to and decisions of the different stages is illustrated in Figure 4.1.

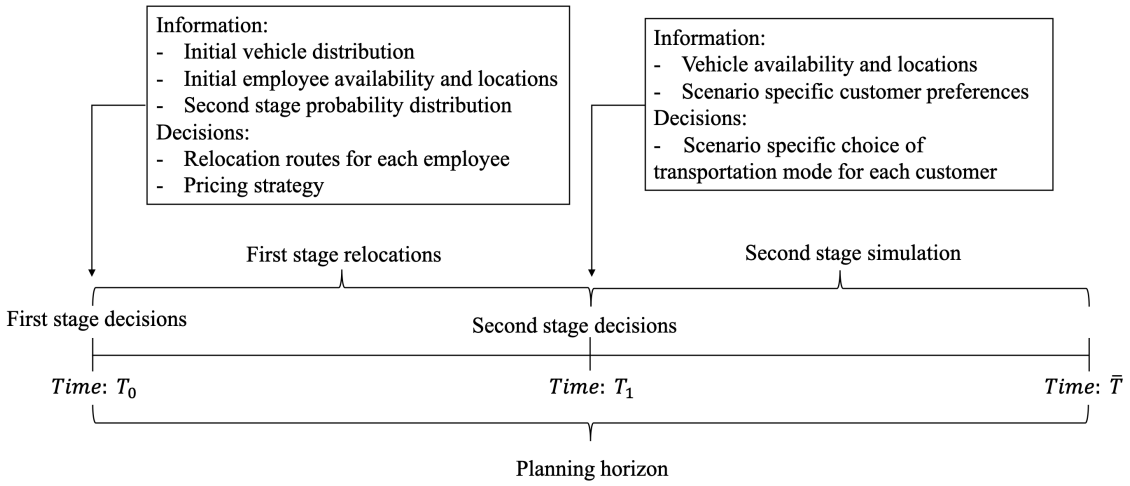


Figure 4.1: Timeline of information and decisions over the planning horizon.

An essential extension of the model of Folkestad and Klev (2021) is the inclusion of pricing strategies. A specific fee or bonus is assigned to each customer trip with an EV between two zones, as done by Pantuso (2020). This means that there is one combined pick-up and drop-off fee from all zones to all zones. Henceforth this combined pick-up and drop-off fee or bonus from all zones to all zones is referred to as the *fee*, or described accordingly. The value of this fee will play an important role in whether or not a customer wishes to travel by EVs. The fee can be positive, zero or negative and is assigned during the first stage.

The demand is modeled as the customers that prefer the use of the service provided by the CSO instead of the other available transportation modes. The available transportation modes for the customers are car-sharing, public transport and bicycling. Each transportation mode has different characteristics associated with it, e.g., travel time, distance, price, and comfort. Based on customer preferences in relation to these characteristics, each available transportation mode yields a different utility for each of the customers. As we assume to know the positions and the desired destinations of all the customers, as well as the position of the cars, the only stochastic part is the customers behaviors. Each customer has a specific set of preferences when it comes to the characteristics a transportation mode yields, and the CSO wants to take decisions based upon these preferences. These preferences, however, are not fully known to the CSO. Therefore, the preferences are approximated by a quantification of some characteristics. The approximation of the preferences does not capture all of the characteristics affecting the customer, as the customer could be influenced by the weather, gas prices, the purpose of the trip, etc. Because of this, there is included a stochastic element to the behavior of each customer, i.e., a preference unmeasurable to the CSO. This element represents an error in the approximation.

The CSO needs to operate with imperfect information regarding the customer's utility towards the different transportation modes. Parts of the customer preferences, and therefore utilities, are not known to the CSO. This leads to uncertainty in the customer behavior and in the prediction of how customers react to the car distribution and the prices for the car-sharing options set in the first stage. A Random Utility model is used to handle this uncertainty. A Random Utility model assumes a behavior from the decision-maker where "the decision-maker has a perfect discrimination capability" (Bierlaire and Sharif Azadeh, 2016). This coincides with the customer being able to choose the transportation mode that yields the highest utility.

In the second stage of the model, starting at time T_1 in Figure 4.1, the uncertain customer demand for EVs

is realized, i.e., becomes known to the CSO. We model this demand as the set of potential customers who prefer traveling by the car-sharing option. Based on the available transportation modes, the customers decide upon their preferred mode of transport, i.e., the one that yields the highest utility for the specific customer. Scenarios, i.e., different realizations of the stochastic customer behavior, and thus also realizations of the stochastic demand, are included to make the model approximate the real problem. The use of scenarios tries to approximate the true behavior of the customers, as discussed by Eilertsen et al. (2021). Each scenario handles different realizations of the stochastic customer behavior, and each of these realizations results in a set of specific customer requests. The set of customer requests is a set of the customers who prefer to get from their origin to their desired destination using the car-sharing option. The set of requests is preprocessed for each scenario and is then given directly as input to the model, specifically in the second stage. The revenue generated in the second stage of the problem is thus an approximation of the revenue of the stochastic problem with the real-life distribution of customer behavior.

In this thesis, we propose a single-objective model formulation, as opposed to Folkestad and Klev (2021), where maximizing the number of cars moved to charging stations was a separate objective. In our model, the charging of cars is handled with constraints and pricing strategies. Cars in need of charging are not available to employees or customers unless the destination of the relocation or trip is a charging station. Further, strategic pricing incentivizes customers to move cars in need of charging to charging stations. In Eilertsen et al. (2021), a necessary share of cars has to be charged in every possible scenario, resulting in a set of worst-case constraints. This formulation proved to be too restrictive. In order to avoid an infeasible problem for real-sized instances, the constraints were modified to only require an expected number of cars to be charged, averaged over all scenarios.

4.2 Assumptions

The CSO faces the problem of jointly deciding the optimal relocations and prices to charge in order to maximize expected profit. To clarify the characteristics and perimeter of the problem, the following assumptions are stated.

Operating area

The business area for a free-floating service is divided into a finite set of zones. Suitably partitioned geographical locations represent the zones.

Stages of the planning period

The CSO will make all necessary decisions in the first stage, which will form the foundation for the second stage. The first stage is the period leading up to the second stage, as illustrated in Figure 4.1.

Fees and Bonuses

The price faced by the customers of the car-sharing service is made as a combination of a combined pick-up and a drop-off fee and a per-minute rental fee multiplied by the travel time between the customer's origin and desired destination. The per-minute rental fee is independent of the pick-up and drop-off location and is valid at any time of the day. The per-minute rental fee is fixed and not a part of the decision variables. The combined pick-up and drop-off fee from all zones to all zones can be either positive, zero or negative. Negative fees are available to incentivize desirable customer-based relocations, and positive fees are available to discourage less desirable customer-based relocations. The assignment of fees to the customer requests is a

part of the first-stage decisions to be made.

Different transportation options

Each zone within the business area offers several different transportation options outside the control of the CSO. The available transportation modes are the same in every zone, except for the possibility of the CSO having no EVs in a particular zone, leading to car-sharing not being an option in that particular zone. In every zone, public transport and bicycling are always available. Walking is not included as a transportation mode between zones, as we assume the customer to choose to travel by bicycle instead. A transportation option is only viable if it can take the customer from its origin zone to the desired destination zone.

Customer behavior

The CSO informs customers about the pricing scheme, i.e., the combined pick-up and drop-off fees, for their desired travel route. Customers are also informed about other available transportation modes and know their respective prices and characteristics, e.g., travel time, waiting time, and comfort level.

Every potential customer has a specific origin and desired destination zone and must complete this trip. The demand is fulfilled by using the car-sharing service or an alternative transportation mode. The choice of transportation is dependent on the utilities yielded by the different transportation modes.

Each possible transportation mode yields a personal utility for a specific customer. The customer is assumed to be choosing the transportation mode that yields the personally highest utility.

All customers have preferences for the different available means of transport. These preferences can be related to travel time, waiting time, walking distance, and price. Most of these preferences are observable to the CSO. There are, however, unobservable elements affecting the customers' behavior, and by that, the problem is stochastic.

When customers desire to travel, they always move directly from their origin zone to their desired destination zone. Consequently, a customer cannot travel to another zone and use the transportation modes in that zone to get to the desired destination zone.

Homogeneous fleet

We assume that the customer is indifferent to the choice of car, and the fleet is therefore considered homogeneous.

CSO desires profit maximization

The CSO could have different objectives, such as revenue maximization, satisfied demand maximization, minimization of zonal deficiency of cars, or profit maximization. In this thesis, the CSO is assumed to desire profit maximization.

Always available parking

There is always parking space available in each zone, and thus no limitation concerning the possible numbers of cars or customer trips to this zone.

Fixed number of employees

The CSO has a fixed number of employees available for relocating cars. The employee salaries are considered fixed, and excluded from the model. An employee can be assigned to relocate one or more cars during the first stage of the planning horizon. All relocations must be completed within the first stage.

Employee transportation

Employees use folding bicycles when traveling to the cars they are assigned to relocate. The bicycles fit in the trunk of the cars.

Fixed number of charging stations

We assume the CSO has a fixed number of charging stations in specified zones in the operating area. We further assume that each station has a limited capacity.

A car can only be moved once in each stage

Each car can be relocated by an employee at most once during the planning horizon. Further, a car can serve customer demand at most once within the planning horizon.

4.3 Notation

The notation presented in this subsection is used in the two-stage stochastic programming model for the integrated SE-VReP-DP. Notation for both the first and the second stages of the model is included. A summary of all sets, parameters and decision variables can be found in Table 4.1, Table 4.2 and Table 4.3, respectively.

4.3.1 Sets

This section introduces the different sets of data used in the two-stage stochastic programming model. These include the operating area, the vehicles with associated sets, employees with associated sets, and customers with associated sets.

The operating area is partitioned into a set of zones, $\mathcal{I}$. For modeling purposes, we introduce subsets of zones containing charging stations, $\mathcal{I}^{CS} \subset \mathcal{I}$. These zones can be used for normal parking as well as for charging cars in need of charging.

The set of employees making the relocation of the cars for the CSO is given by $\mathcal{E}$. A set containing abstract tasks to be done by the employees is given by $\mathcal{M}$. This set is solely used to keep track of the task number an employee is performing at each time.

The set of car-sharing vehicles is given by $\mathcal{V}$. We assume vehicles put to charging are excluded from this set, as they are unavailable to customers. Further, a subset of vehicles in need of charging, $\mathcal{V}^B \subset \mathcal{V}$, is defined. These cars are only available for relocations or trips ending in a charging zone. The set of possible relocations, called car-moves, performed by employees for all cars is given by $\mathcal{R}$. There are also subsets of $\mathcal{R}$ that can be indexed by car, $\mathcal{R}_v^V$, and destination zone, $\mathcal{R}_i^N$. Indexation by car v gives all possible destinations

for the specified car, and indexation by zone i gives all cars that can get to the specified destination zone. $\mathcal{R}_i^{CD} \subset \mathcal{R}_i^N$ contains the set of charging moves with destination zone i.

The set of customers is given by $\mathcal{K}$. Subsets $\mathcal{K}_i$ and $\mathcal{K}_{ij}$ contain customers travelling from zone i, and from zone i to j, respectively. The customers are indexed in ascending order regarding the time they make a request. To handle the uncertainty in customer behavior, a set of scenarios, $\mathcal{S}$, is introduced. A random variable $\tilde{\xi}_{kv}$ is used to capture the uncertainty of the utility yielded for customer k with vehicle v. In the model, each scenario s is a given realization of $\tilde{\xi} := (\tilde{\xi}_{kv})_{k\in\mathcal{K},v\in\mathcal{V}}$, $\xi_s := (\xi_{kvs})_{k\in\mathcal{K},v\in\mathcal{V}}$. The scenarios capture different possibilities of how customers react to the prices and locations of cars from the first stage and other available transportation modes.

A customer trip with a car is associated with a combined pick-up and drop-off fee and a per-minute fee. The set of possible combined pick-up and drop-off fee levels that are available to the CSO is given by the set $\mathcal{L}$.

We initialize a set of customer requests, $\mathcal{D}(\xi_s)$, for a given realization ξ_s in scenario s. The set $\mathcal{D}(\xi_s)$ contains a request for each customer k who prefers using car-sharing for at least one fee level l over the other available transportation modes. The origin, destination and customer of request d is represented by $i(d)$, $j(d)$ and $k(d)$, respectively, and the highest fee level which customer $k(d)$ would prefer car-sharing to other transportation modes is represented by $l(d)$. We further create a subset of $\mathcal{D}^{CD}(\xi_s) \subset \mathcal{D}(\xi_s)$ containing all requests with destination in a charging zone, and the requests going to charging zone $i \in \mathcal{I}^{CS}$ is given by $\mathcal{D}_i^{CD}(\xi_s) \subset \mathcal{D}^{CD}(\xi_s)$. The requests that has precedence over request d is given by $\mathcal{D}_d(\xi_s) = \{p \in \mathcal{D}(\xi_s) : i(p) = i(d), k(p) < k(d)\} \subset \mathcal{D}(\xi_s)$, i.e., requests originating in the same zone as d with higher priority than d. The priority is determined by the time the customer makes the request, where the first customer to make a request is chosen. The requests going from zone i to zone j is given by $\mathcal{D}_{ij}(\xi_s) \subset \mathcal{D}(\xi_s)$.

Finally, $\mathcal{L}_d \subset \mathcal{L}$ is the subset of fees that are less than the highest fee level acceptable for request d, $l(d)$.

4.3.2 Parameters

The parameters in the model's objective function are associated with either cost, revenue, or time. The toll, maintenance, and energy (electricity) cost of driving the car are accumulated to C^R per time unit. The salaries for employees are assumed fixed, and the number of employees will remain the same, and their salaries are therefore not a part of the model. The time it takes to do the relocation car-move r is T_r^H. This parameter includes the time it takes to get from the origin to the destination zone of the car-move, in addition to the time it takes to find a parking spot or initiate charging. P_s gives the probability that scenario s will occur. The profit of satisfying request d with the combined pick-up and drop-off fee level l is given by R_{dl}. The profit represents the revenue and cost related to customers using the cars.

The origin of car v, i.e., where the car is located at the beginning of the planning horizon, is given by the parameter $o(v)$. Likewise, the origin and the destination of car-move r is given by the parameters $o(r)$ and $d(r)$.

The earliest start time an employee e is available for performing a task is given by T_e^{SO}. This parameter handles the case when employees experience possible delays due to traffic, other ongoing relocations, or personal matters, leading to the employee not being available at the start of the planning horizon. The origin of the employee e is given by $o(e)$.

The time length of the planning horizon is the time of the first stage and the time of the second stage. The time length of the first stage is given by $\overline{T}^1$. We assume that all customer requests will be initiated within the

time period of the second stage and further served immediately after initiation. The first time point when a car is available for a car-move r is T_r^{SC}. This value can be non-zero, as a car can be in use from previous relocations or used to serve a customer prior to the start. The time it takes to travel between zones i and j by bicycle is given by T_{ij}.

The value of fee level l is given by the parameter L_l.

The number of available charging spots in zone $i \in \mathcal{I}^{CS}$ is given by the parameter N_i^{CS}.

The share of cars in need of charging, which needs to be charged in the planning horizon, is given by the parameter N^B.

In the second stage a request d has an origin and destination represented by $i(d)$ and $j(d)$, respectively. The customer of the request is given by $k(d)$. The highest fee level which customer $k(d)$ would prefer car-sharing to other transportation modes is represented by $l(d)$.

M_r is a Big-M notation, which is used to handle the logic in the mathematical model presented in Section 4.4, and is defined for each car-move r. Further explanations concerning M_r are provided in Appendix B

4.3.3 Decision Variables

The binary variable z_{iv} takes the value one if vehicle v is available to customers in zone i at the beginning of the second stage, at time $\overline{T}^1$, after the employee-based relocations in stage one have been done, otherwise it is set to 0.

The binary variable λ_{ijl} takes the value one if the combined pick-up and drop-off fee level l is used between zone i and j, otherwise it is set to 0.

The binary variable x_{erm} takes the value one if employee e makes relocation move r as her task number m; otherwise, it is set to 0. x_{erm} will contribute to the ordering of the tasks to be done by the employee. The time when employee e will start doing task m is given by t_{em}.

In the second stage, the binary variable y_{vdls} takes the value 1 if vehicle v satisfies request d with the fee level l in scenario s, otherwise it is set to 0. If a request has its destination in a zone without a charging station, it cannot be served by a car in need of charging. In these situations, y_{vdls} is not defined.

Table 4.1: Sets used in the model.

Notation	Explanation
$\mathcal{I}$	Set of zones
$\mathcal{I}^{CS}$	Set of charging zones
$\mathcal{E}$	Set of employees
$\mathcal{M}$	Set of abstract employee tasks
$\mathcal{V}$	Set of car-sharing vehicles
$\mathcal{V}^B$	Set of vehicles in need of charging
$\mathcal{R}$	Set of car-moves
$\mathcal{R}_v^V$	Set of car-moves for vehicle v
$\mathcal{R}_i^N$	Set of car-moves with destination zone i
$\mathcal{R}_i^{CD}$	Set of car-moves with destination zone $i \in \mathcal{I}^{CS}$
$\mathcal{K}$	Set of customers
$\mathcal{K}_i$	Set of customers travelling from zone i
$\mathcal{K}_{ij}$	Set of customers travelling from zone i to j
$\mathcal{S}$	Set of scenarios
$\mathcal{L}$	Set of possible combined pick-up and drop-off fee levels
$\mathcal{D}(\xi_s)$	Set of requests for a given realization of ξ_s in scenario s
$\mathcal{D}^{CD}(\xi_s)$	Set of requests with destination in a charging zone for a given realization of ξ_s in scenario s
$\mathcal{D}_i^{CD}(\xi_s)$	Set of requests with destination in charging zone i for a given realization of ξ_s in scenario s
$\mathcal{D}_d(\xi_s)$	Set of requests that has precedence over request d in scenario s
$\mathcal{D}_{ij}(\xi_s)$	Set of requests going from zone i to j in scenario s
$\mathcal{L}_d$	Set of combined pick-up and drop-off fee levels that are less than or equal to the highest fee accepted for request d

Table 4.2: Parameters used in the model.

Notation	Explanation
C^R	Accumulated toll, maintenance and energy (electricity) cost per minute of driving a car
T_r^H	The time it takes to perform the car-move r
P_s	Probability of scenario s occurring
R_{dl}	Profit of satisfying request d with pick-up fee level l
$o(v)$	Origin of car v at start of planning horizon
$o(r)$, $d(r)$	Origin and destination of car-move r, respectively
T_e^{SO}	Earliest start time employee e is available to perform a task
$o(e)$	Origin of employee e at start of planning horizon
$\overline{T}^1$	Time length of the first stage
T_r^{SC}	First available time a car is available for car-move r
T_{ij}	The time it takes to travel between zone i and j by bicycle
L_l	Value of a fee at fee level l
N_i^{CS}	Number of available charging slots in zone $i \in \mathcal{I}^{CS}$
N^B	Share of cars in need of charging which needs to be charged in the planning horizon
$i(d)$, $j(d)$	Origin and destination of customer request d, respectively
$k(d)$	Customer of customer request d
$l(d)$	The highest fee level at which customer $k(d)$ would prefer car-sharing to other transportation modes
M_r	Big-M notation for each car-move r

Table 4.3: Decision variables used in the model.

Notation	Explanation
z_{iv}	1 if vehicle v is available to customers in zone i at the beginning of the second stage, at time $\overline{T}^1$, 0 otherwise.
λ_{ijl}	1 if fee level l is used between zones i and j, 0 otherwise.
x_{erm}	1 if employee e performs car-move r as her task number m, 0 otherwise.
y_{vdls}	1 if vehicle v satisfies request d with the fee level l in scenario s, 0 otherwise.

4.4 Mathematical Model

In this section, we present the mathematical model formulation. For simplicity, we present the first and second stages of the model separately. Note, however, that the two stages are connected. Minor changes have been made to the model presented by Eilertsen et al. (2021). Furher, one set of second-stage constraints is

logically modified from the original formulation in Eilertsen et al. (2021), as it was too restrictive.

4.4.1 First Stage

Objective Function

$$\max z = -\sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}}\sum_{m\in\mathcal{M}} C^R T_r^H x_{erm} + \sum_{s\in\mathcal{S}} P_s Q_s(z,\lambda,\xi_s) \tag{4.1}$$

The first-stage objective function (4.1) contains two main elements: a cost term and a revenue term. The former represents the cost related to the employee-based relocation of EVs. This includes the relocation cost, the time spent on relocation, and the binary relocation variable. The revenue term represents the expected revenue realized in the second stage. $Q_s(z,\lambda,\xi_s)$ is the expected revenue generated for each scenario s, given the first-stage relocations defining z, pricing decisions λ, and the stochastic, scenario-dependent element ξ_s. In order to calculate the expected revenue, the Sample Average Approximation (SAA) method is used. The SAA method approximates the real solution by averaging the results from the different scenarios, where the probability of a scenario P_s occuring is equal for all scenarios s, $P_s = \frac{1}{|\mathcal{S}|}$. The objective is to maximize the expected profits, consisting of the difference between expected revenues from second-stage rentals and the costs from performing the first-stage relocations.

Pricing of EVs

$$\sum_{l\in\mathcal{L}} \lambda_{ijl} = 1, \qquad i,j \in \mathcal{I} \tag{4.2}$$

Constraints (4.2) assure there is precisely one fee l assigned to any trip between two zones i and j. This is the only requirement needed for the pricing strategy to be valid.

Availability of EVs in the Second Stage of the Model

$$1 - \sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} = \sum_{i\in\mathcal{I}} z_{iv}, \qquad v \in \mathcal{V}^B \tag{4.3}$$

$$\sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_i^N\cap\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} = z_{iv}, \qquad i \in \mathcal{I}\backslash\{o(v)\}, v \in \mathcal{V}\backslash\{\mathcal{V}^B\} \tag{4.4}$$

$$1 - \sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} = z_{o(v),v}, \qquad v \in \mathcal{V} \tag{4.5}$$

$$\sum_{i\in\mathcal{I}} z_{iv} \leq 1, \qquad v \in \mathcal{V} \tag{4.6}$$

Constraints (4.3)-(4.6) handle the logic concerning the availability of EVs in each zone for the second stage of the model. Availability in this paragraph refers to availability in the second stage of the model. Constraints (4.3) state that an EV in need of charging is made unavailable in all zones if it is relocated by an employee, as this relocation would imply the car being put to charging. If it is not relocated, it remains available in exactly one zone, as customers have the option to take the car to a charging station. Constraints (4.4) assure that if an EV is relocated to zone i, it is made available for rentals in zone i. Constraints (4.5) force an EV to be available in its original zone if it is not relocated and similarly unavailable in its original zone if it is moved. Finally, Constraints (4.6) make sure an EV is available for rental in at most one zone. An EV can be unavailable if relocated to a charging station, explaining the less than or equal sign.

Relocation of EVs

$$\sum_{r\in\mathcal{R}} x_{e,r,(m+1)} \leq \sum_{r\in\mathcal{R}} x_{erm}, \qquad e \in E, m \in \mathcal{M}\backslash\{|\mathcal{M}|\} \tag{4.7}$$

$$\sum_{r\in\mathcal{R}} x_{erm} \leq 1, \qquad e \in \mathcal{E}, m \in \mathcal{M} \tag{4.8}$$

$$\sum_{e\in\mathcal{E}} \sum_{r\in\mathcal{R}_v^V} \sum_{m\in\mathcal{M}} x_{erm} \leq 1, \qquad v \in \mathcal{V} \tag{4.9}$$

$$\sum_{e\in\mathcal{E}} \sum_{i\in\mathcal{I}^{CS}} \sum_{r\in\mathcal{R}_i^{CD}} \sum_{m\in\mathcal{M}} x_{erm} + \sum_{s\in\mathcal{S}} P_s W(z,\lambda,\xi_s) \geq N^B \cdot |\mathcal{V}^B| \tag{4.10}$$

Constraints (4.7)-(4.10) handle the logic concerning the relocation of EVs. Constraints (4.7) make sure that an employee e cannot perform any relocations if she did not complete the previous task. Constraints (4.8) state that any task for any employee can contain at most one request. Finally, Constraints (4.9) make sure each vehicle can be relocated at most once. Constraint (4.10) ensures that the expected number of cars in need of charging that are moved to charging zones either by employees or customers is more than the necessary share of cars to charge. Here, $W(z,\lambda,\xi_s)$ is the expected number of customer-based charging-moves, defined by (4.22) in the second stage of the model. This differs from the formulation in Eilertsen et al. (2021), where the constraints were part of the second stage, and the number of cars charged had to fulfill the necessary share in each scenario. The reason for this change is that the constraints were too restrictive, as it only considered worst-case scenarios.

Time Usage

$$t_{em} + T_r^H \cdot x_{erm} + \sum_{r_1\in\mathcal{R}\backslash\{r\}} T_{d(r),o(r_1)} x_{e,r_1,(m+1)} - M_r(1 - x_{erm}) \leq t_{e,(m+1)},$$

$$e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M}\backslash\{|\mathcal{M}|\} \tag{4.11}$$

$$t_{em} \leq t_{e,(m+1)}, \qquad e \in \mathcal{E}, m \in \mathcal{M}\backslash\{|\mathcal{M}|\} \tag{4.12}$$

$$(T_e^{SO} + T_{o(e),o(r)}) \cdot x_{er1} \leq t_{e1}, \qquad e \in \mathcal{E}, r \in \mathcal{R} \qquad (4.13)$$

$$T_r^{SC} x_{erm} \leq t_{em}, \qquad e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M} \qquad (4.14)$$

$$t_{e|\mathcal{M}|} + \sum_{r \in \mathcal{R}} T_r^H x_{er|\mathcal{M}|} \leq \overline{T}^1, \qquad e \in \mathcal{E} \qquad (4.15)$$

Constraints (4.11)-(4.15) handle the logic concerning temporal aspects for the employees. Constraints (4.11) assure that a new task cannot start until the previous task is complete, and the employee performing the tasks has traveled from the destination of the previous task to the origin of the new task. Constraints (4.12) force the start time of a new task to not be before the start time of the previous task for each employee. Constraints (4.13) make sure that the first task of each employee starts after the earliest start time for the employee and after the employee has traveled from its origin to the origin of the relocation. Constraints (A.14) ensure that the start time of each task for each employee is after the earliest start time for the specific relocation. Finally, Constraints (4.15) assure that the last task for each employee is complete within the duration of the first stage.

Binary and Non-Negativity Definitions

$$\lambda_{ijl} \in \{0, 1\}, \qquad i, j \in \mathcal{I}, l \in \mathcal{L} \qquad (4.16)$$

$$z_{iv} \in \{0, 1\}, \qquad i \in \mathcal{I}, v \in \mathcal{V} \qquad (4.17)$$

$$x_{erm} \in \{0, 1\}, \qquad e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M} \qquad (4.18)$$

$$t_{em} \geq 0, \qquad e \in \mathcal{E}, m \in \mathcal{M} \qquad (4.19)$$

Constraints (4.16)-(4.19) define the domain of the variables introduced in the first stage of the model.

4.4.2 Second Stage

Objective Function

$$Q_s(z, \lambda, \xi_s) = \max \sum_{d \in \mathcal{D}(\xi_s)} \sum_{v \in \mathcal{V}} \sum_{l \in \mathcal{L}_d} R_{dl} y_{vdls} \qquad (4.20)$$

The second-stage objective function is to maximize the revenue of satisfied requests. This is given by Equation (4.20), iterating over the possible scenarios s, multiplying the probability of each scenario P_s with the revenue of satisfied requests in this scenario. The revenue is obtained by multiplying the revenue made by completing a request at a specific fee with the corresponding binary decision variable y_{vdls}. The first-stage objective function (4.1) iterates over the entire set of scenarios. For a given scenario s, the following constraints apply.

Request Handling

$$\sum_{e\in\mathcal{E}}\sum_{v\in\mathcal{V}^B}\sum_{r\in\mathcal{R}_i^{CD}\cap\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} + \sum_{v\in\mathcal{V}^B}\sum_{d\in\mathcal{D}_i^{CD}(\xi_s)}\sum_{l\in\mathcal{L}} y_{vdls} \leq N_i^{CS}, \qquad i\in\mathcal{I}^{CS} \tag{4.21}$$

$$W(z,\lambda,\xi_s) = \sum_{v\in\mathcal{V}^B}\sum_{d\in\mathcal{D}^{CD}(\xi_s)}\sum_{l\in\mathcal{L}} y_{vdls} \tag{4.22}$$

$$\sum_{v\in\mathcal{V}}\sum_{l\in\mathcal{L}_d} y_{vdls} \leq 1, \qquad d\in\mathcal{D}(\xi_s) \tag{4.23}$$

$$\sum_{d\in\mathcal{D}(\xi_s)}\sum_{l\in\mathcal{L}_d} y_{vdls} \leq 1, \qquad v\in\mathcal{V} \tag{4.24}$$

$$\sum_{l\in\mathcal{L}_{d_1}} y_{vd_1ls} + \sum_{d_2\in\mathcal{D}_{d_1}(\xi_s)}\sum_{l\in\mathcal{L}_{d_2}} y_{vd_2ls} \leq z_{i(d_1),v}, \qquad d_1\in\mathcal{D}(\xi_s), v\in\mathcal{V} \tag{4.25}$$

$$y_{vd_1l_1s} + \sum_{d_2\in\mathcal{D}_{d_1}(\xi_s)}\sum_{l_2\in\mathcal{L}_{d_2}} y_{vd_2l_2s} + \sum_{v_1\in\mathcal{V}:v_1\neq v} y_{v_1d_1l_1s} \geq \lambda_{i(d_1),j(d_1),l_1} + z_{i(d_1),v} - 1,$$

$$d_1\in\mathcal{D}(\xi_s), v\in\mathcal{V}, l_1\in\mathcal{L}_{d_1} \tag{4.26}$$

$$\sum_{v\in\mathcal{V}} y_{vdls} \leq \lambda_{i(d),j(d),l}, \qquad d\in\mathcal{D}(\xi_s), l\in\mathcal{L}_d \tag{4.27}$$

Constraints (4.21)-(4.27) handle the logic concerning the customer requests. Constraints (4.21) make sure the capacities of the charging zones are not exceeded by only considering moves of cars in need of charging, $v\in\mathcal{V}^B$, as other cars also can be relocated to charging zones for purposes other than charging. Constraints (4.22) assure that $W(z,\lambda,\xi_s)$ equals the number of customer charging-moves in each scenario. This constraint is in place to make sure that (4.10) in the first stage works as intended. Constraints (4.23) ensure that each

request is at most satisfied by one vehicle at one possible fee level, and Constraints (4.24) ensure that each car satisfies at most one request at a possible fee level. Constraints (4.25) say that a vehicle v can satisfy a request d_1 only if it is available in the zone $i(d_1)$ and it is not a part of a request of a customer arriving earlier, i.e., a customer with a lower index. Constraints (4.26) make sure that if a request d_1 at level l_1 must be satisfied by vehicle v if fee level l_1 is chosen for that request and the vehicle is available in the origin zone, unless the car has been used to satisfy a higher priority request d_2 or the request d_1 has been satisfied by another vehicle v_1. Lastly, Constraints (4.27) state that if a request is satisfied by a vehicle at a certain fee level l, then the fee must be chosen for that request ($\lambda_{i(d),j(d),l} = 1$). Further, if fee l is not assigned to request d ($\lambda_{i(d),j(d),l} = 0$), then no vehicles can satisfy the request at fee level l ($y_{vdls} = 0$).

Binary

$$y_{vdls} \in \{0, 1\}, \qquad v \in \mathcal{V}, d \in \mathcal{D}(\xi_s), l \in \mathcal{L} \tag{4.28}$$

Constraints (4.28) define the domain of the variables introduced in the second stage of the model.

Chapter 5

A Heuristic Approach for Solving SE-VReP-DP

Eilertsen et al. (2021) conclude that a commercial MIP-solver cannot solve real-sized instances of the Stochastic Electrical Vehicle Relocation Problem with Dynamic Pricing (SE-VReP-DP) in a reasonable time. In order to find good solutions to real-sized instances, a heuristic approach is proposed. Eilertsen et al. (2021) also found that changes in the pricing solution yield significant improvements in the objective function, as opposed to changes in the employee-based relocations. Therefore, the proposed heuristic focus more on finding good pricing solutions than relocation solutions. An Adaptive Large Neighborhood Search (ALNS) with integrated Local Search (LS) heuristic, partly based on Ropke and Pisinger (2006) and Folkestad and Klev (2021), is presented as a solution method for the SE-VReP-DP in this chapter. Folkestad and Klev (2021) demonstrated that the ALNS is an efficient solution method for the SE-VReP, which is a problem with similarities to the SE-VReP-DP.

The problem and method presented in this thesis differ from similar studies proposed in the literature in Section 3.5, as dynamic pricing is included. Recall from Chapter 1 that we refer to the pricing strategy as dynamic pricing, as it is meant to be solved frequently where prices are dynamically modified based on the current situation in a real-life setting. The goals of the approach proposed in this chapter are to keep the cars sufficiently charged to cover future demand, allocate cars to meet expected demand, and set the fee levels optimally to achieve high revenues and, thereby, profits.

In Section 5.1, a high level overview of the heuristic is introduced. Section 5.2 presents the representation of a solution. Section 5.3 and 5.4 describe how feasibility is handled and how the heuristic calculates the objective value, respectively. In Section 5.5, the construction heuristic used for constructing an initial solution is described. Section 5.6 introduces the ALNS. In Section 5.7, all Local Search Operators, both for relocation and pricing are introduced. Lastly, Section 5.8 presents the different methods of reducing the solution space, and consequently the computation time for the heuristic.

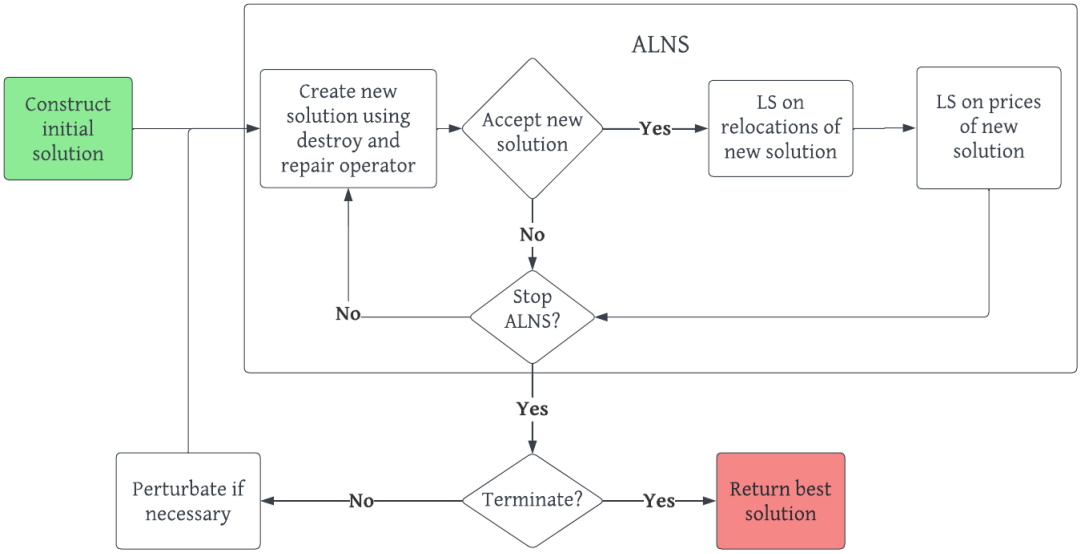


Figure 5.1: Flowchart presenting the different processes of the ALNS heuristic. It is a visualization of the more detailed Algorithm 1.

5.1 High-level overview of the algorithm

An overview of the algorithm for solving the SE-VReP-DP, called the *algorithm* in the remainder of this chapter, is presented in Algorithm 1. A flowchart showing the separate steps of the algorithm is presented in Figure 5.1. The algorithm starts with the solution generated by the construction heuristic, modifies it, and returns the best solution found. There are three main components, namely the construction heuristic, the Adaptive Large Neighborhood Search (ALNS), which is integrated with two Local Seach (LS) algorithms, and the *perturbation* of the current solution. These and the surrounding aspects will be explained in the remainder of this chapter.

Algorithm 1 Heuristic Framework, the *algorithm*

- **function** *main*
  InitialSolution = Solution found by $ConstructionHeuristic(\gamma_\emptyset)$
  CurrentSolution = *InitialSolution*
  BestSolution = *InitialSolution*
  while termination condition is **not** met **do**
    CurrentSolution = ALNS(*CurrentSolution*)
    if *CurrentSolution* is better than *BestSolution* **then**
      BestSolution = *CurrentSolution*
    end if
    if condition for applying a *perturbation* on the *CurrentSolution* is met **then**
      apply *perturbation* to *CurrentSolution*
    end if
  end while
  return *BestSolution*
end function

As mentioned above, the algorithm starts by constructing a solution using a construction heuristic, the initial solution. This accounts for both the pricing decisions as well as the relocations which should be done. The details surrounding the construction heuristic is described in Section 5.5. The construction heuristic takes an empty solution as input and modifies it to be feasible if it is not already. This solution found by the construction heuristic is assigned to the variables called *CurrentSolution* and *BestSolution* in Algorithm 1.

The initial solution is continuously improved based on the principles of ALNS presented by both Ropke and Pisinger (2006) and Folkestad and Klev (2021). Our implemented ALNS, which is integrated with different Local Search (LS) heuristics, is described in detail in Sections 5.6 and 5.7. The ALNS updates the *CurrentSolution* for every iteration. The ALNS is integrated with two LS heuristics, and they are working on different solution spaces. The ALNS is making changes to the pricing solution by applying different destroy and repair operators, while one LS heuristics is also making smaller changes to the pricing solution. On the other hand, the other LS heuristic is making changes to the employee-based relocation solution. Recall from Section 4.1 that this problem is a two-stage problem, where the pricing and relocation decisions in the first stage impact the second-stage approximation of the profit of the CSO. Whenever the ALNS finds a new global best solution, the *BestSolution* in Algorithm 1 is updated accordingly. This process continues until the termination condition is met, which is described at the end of this section.

The final element of the algorithm is the *perturbation* process. To help the algorithm escape local optima, a share of κ of the pricing solution is set to the initial values, guiding the heuristic into a different area of the solution search space. This principle is based upon the work of Lourenço et al. (2003).

The termination condition for the algorithm consists of two conditions. The first one is a limitation on the maximum number of iterations, I^{main}, to run the ALNS (the while-loop in Algorithm 1). The second is a time limit, T^{limit}. Notice that as we are checking the stopping criteria after a run of the ALNS, the algorithm might run slightly longer than T^{limit}.

5.2 Solution Representation

A solution γ consists of both a relocation solution, γ^E, and a pricing solution, γ^P. They represent the same as the x-variables and the λ-variables, respectively, presented in Section 4.3.3.

The relocation solution, γ^E, consists of a set of tasks each employee needs to perform and in which order. Recall that the employees can perform car-moves from $\mathcal{R}$. If one of these car-moves, r, is performed, then r is associated with one employee, e, i.e., e will perform r. Each employee may have multiple car-moves to perform, so when r is associated with e it is also associated with a task number. This makes the task list an ordered list of car-moves. Thus γ^E contains an ordered list of car-moves to be performed by each employee.

Table 5.1 shows an example of a relocation solution, γ^E. Here we see that there are two employees, $e1$ and $e2$. $e1$ is associated with two tasks, i.e., two car-moves to be performed during the planning horizon. $e1$ will perform $cm1$ first and $cm2$ thereafter. $e2$ is associated with three tasks, i.e., three car-moves to be performed during the planning horizon, where the first task to be performed is $cm3$, then $cm4$ and lastly $cm5$.

Table 5.1: Example of a relocation solution, γ^E.

Employee	**Task 1**	**Task 2**	**Task 3**	**Task 4**
e1	cm1	cm2		
e2	cm3	cm4	cm5	

The pricing solution, γ^P, is a matrix of assigned fees for every pair of zones. Recall that when a customer request is satisfied, the price to be paid is the per-minute fee in addition to the assigned fee of the origin-destination zone pair. There has to be exactly one fee between every pair of zones, and this fee can be either positive, zero or negative, selected from a finite set of values.

Table 5.2 shows an example of a pricing solution, γ^P. Here we have an example situation where the operating area is split into four zones. The price between the origin zone o and the destination zone d is given in the matrix as p_{od}.

Table 5.2: Example of a pricing solution, γ^P.

$o \backslash d$	**1**	**2**	**3**	**4**
1	p_{11}	p_{12}	p_{13}	p_{14}
2	p_{21}	p_{22}	p_{23}	p_{24}
3	p_{31}	p_{32}	p_{33}	p_{34}
4	p_{41}	p_{42}	p_{43}	p_{44}

An empty solution, $\gamma_\emptyset$, is to be sent into the construction heuristic, introduced in Section 5.1 and further discussed in Section 5.5. In the relocation part of the empty solution, $\gamma^E_\emptyset$, none of the employees have any tasks to perform. In the pricing part of the empty solution, $\gamma^P_\emptyset$, the fee assigned for all pairs of zones is the lowest fee level possible.

5.3 Feasibility Function

The ALNS and LS heuristic operations are to find the best possible solution, provided it is applicable to all constraints presented in the mathematical formulation in Section 4.4. To assure a solution's feasibility, the heuristic performs feasibility checks after a new solution is proposed by any of the three heuristic operations, namely the ALNS on prices, the LS on prices, and the LS on employee-based relocations.

The pricing solution, γ^P, is mainly constrained by Constraints (4.2), stating that each pair of zones must be assigned exactly one fee from the set of fees, $\mathcal{L}$. This is handled implicitly in the ALNS itself and the LS for prices, as only fees in $\mathcal{L}$ are explored, and it ensures that all the zone pairs have an associated fee level. Recall from Section 5.2 that the empty pricing solution, $\gamma^P_\emptyset$, assigns the lowest fee to all zone pairs, and is thus preventing having zone pairs without fees, thereby handling Constraints (4.2).

The relocation solution, γ^E, is constrained more complexly, as this part of the solution includes spatial and temporal aspects. Firstly, an employee must be able to complete all its assigned car-moves in a given order

within the time length of the first stage, $\overline{T}^1$. This is represented by Constraints (4.11) - (4.15). The ordering of car-moves is handled in the solution representation of γ^E, given as an ordered list of car-moves for each employee, where the index of the car-move represents the task number used in the mathematical formulation. The time needed for an employee to travel by bicycle between the car-moves, and the time needed to perform all car-moves in the order provided, are calculated using the distance matrix for traveling by bicycle, T_{ij}, and the relocation time parameter, T_r^H. If the calculated time exceeds $\overline{T}^1$ for an employee when considering adding a car-move to the employee's task list, the car-move is not added.

Secondly, the number of cars set to charging may either be too low or exceed the charging capacity at certain charging stations. The charging capacity can only be exceeded when considering adding a car-move to an employee's task list. When performing an LS on employee-based relocations, a car-move where the car is in need of charging will only be considered if there is an available charging slot at the charging station associated with the car-move. Thereby Constraints (4.21) are handled. To assure the number of cars set to charging is sufficiently high, we must consider both the number of charging moves in the relocation solution and the number of customer requests to charging stations satisfied in each scenario. An algorithm iterates over all scenarios and all potential customer requests to find the expected number of cars in need of charging moved to a charging station by a customer. The solution is marked as infeasible if the sum of the expected number of cars put to charging by customers and the number of employee-based car-moves of cars in need of charging is lower than the required amount. This assures Constraints (4.10) are held.

Finally, the mathematical model includes general constraints considering the locations of cars and their availability to customers (Constraints (4.3) - (4.6)), relocation logic (Constraints (4.7) - (4.9)), and the allocation of cars to satisfy customer requests (Constraints (4.23) - (4.27). The location of cars are handled in the heuristic logic, keeping track of the origin zone of each car and their new location zone if moved by an employee. A car is made unavailable to customers if it is set to charging by an employee. The relocation logic concerns only allowing one employee to perform each car-move, and assuring each car is relocated at most once. This logic is incorporated in the LS of the relocations when determining the search space for potential car-moves to add to an employee's task list; Only unassigned car-moves are considered possible candidates. The allocation of cars to customers follows an iterative process over scenarios, assigning available cars to willing customers in a prioritized order based on the customers' request IDs. The process further assures that a customer request must be fulfilled if the current pricing solution is accepted by the customer and there is an available car, and that each request at most can be satisfied by one car.

5.4 Heuristic Objective Function

The calculation of the value of a heuristic solution is based on the objective function provided in the mathematical formulation, namely Equation (4.1). As the representation of the model is different in the heuristic model than in the mathematical formulation, we use a different notation to formulate the *heuristic objective function*.

Recall that γ^E is the relocation part of the heuristic solution γ, being an ordered list of car-moves for each employee. Let γ_e^E denote the ordered list of car-moves to be performed by employee e in the relocation solution, and r denote a car-move in this list. Further, recall that the per-minute cost of performing a car-move is C^R, and that the time in minutes to complete car-move r is denoted as T_r^H. The heuristic evaluation of the relocation solution, denoted as f_E, can be written

$$f_E = -\sum_{e \in \mathcal{E}} \sum_{r \in \gamma_e^E} C^R T_r^H \tag{5.1}$$

Recall that γ^P is the pricing solution of the heuristic solution γ, being a matrix of assigned fees between each pair of zones. Let γ_{ij}^P denote the assigned fee l between zones i and j in the pricing solution. Further recall that $\mathcal{D}(\xi_s)$ is the set of requests d for a given realization of ξ_s in scenario s, and that $\mathcal{L}_d$ is the set of fees l sufficiently low to satisfy request $d \in \mathcal{D}(\xi_s)$ for a given realization of ξ_s in scenario s. Each request d has an origin zone $i(d)$, and a destination zone $j(d)$. For a given pricing solution γ_{ij}^P, we create a subset of $\mathcal{D}(\xi_s)$ with all potential requests $d \in \mathcal{D}(\xi_s)$ with origins $i(d)$ and destinations $j(d)$ which could be satisfied by the fee $l = \gamma_{i(d),j(d)}^P$. Mathematically, this subset, denoted as $\mathcal{D}^H(\xi_s)$, is

$$\mathcal{D}^H(\xi_s) = \{d \mid d \in \mathcal{D}(\xi_s) \ \wedge \ \gamma_{i(d),j(d)}^P \in \mathcal{L}_d\}$$

Algorithm 2 Generation of the set of Satisfied Requests $\mathcal{D}^{SR}(\xi_s)$ in scenario s.

input : ξ_s
input : γ^P
$\mathcal{D}^{SR}(\xi_s) = \emptyset$
$\mathcal{V}_i^A$ is the set of all vehicles available in zone i
$\mathcal{V}_i^{BA}$ is the set of vehicles in need of charging available in zone i
for request $d \in \mathcal{D}^H(\xi_s)$ **do**
  $i = i(d)$ ▷ $i(d)$ is the origin zone of request d
  $j = j(d)$ ▷ $j(d)$ is the destination zone of request d
  for available vehicle $v \in \mathcal{V}_i^A$ **do**
    if vehicle v is in need of charging **then**
      if the destination zone j has no charging station **then**
        continue to next vehicle
      end if
      if the destination zone j has a charging station, but the capacity is maxed **then**
        continue to next vehicle
      end if
      Reduce the number of available charging slots at the destination zone
    end if
    $\mathcal{D}^{SR}(\xi_s) = \mathcal{D}^{SR}(\xi_s) \bigcup \{d\}$ ▷ Assign vehicle v to request d by adding d to the set of satisfied requests
    $\mathcal{V}_i^A = \mathcal{V}_i^A \setminus \{v\}$ ▷ Remove vehicle v from the set of available vehicles
  end for
end for
return $\mathcal{D}^{SR}(\xi_s)$

The set $\mathcal{D}^H(\xi_s)$ is ordered by customer priority, so a request in the set has precedence over all subsequent requests. To determine whether a request $d \in \mathcal{D}^H(\xi_s)$ is allocated a vehicle in scenario s, we must consider the following three requirements: There is a vehicle in the request origin zone, $i(d)$; If the vehicle is in need of charging, the destination zone, $j(d)$, must have a charging station with available capacity; The vehicle has not previously been allocated to serve another request with precedence over the current request. The heuristic handles this by iterating over the ordered set of requests $\mathcal{D}^H(\xi_s)$ for each scenario s while checking the three requirements. This process is demonstrated in Algorithm 2, resulting in the set of *satisfied requests* for each realization of ξ_s for all scenarios s, $\mathcal{D}^{SR}(\xi_s)$. We can therefore express the heuristic evaluation of the pricing solution, denoted f_P, as

$$f_P = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \left(\sum_{d \in \mathcal{D}^{SR}(\xi_s)} R_{d,\gamma^P_{i(d),j(d)}} \right). \tag{5.2}$$

Here $\mathcal{S}$ is the set of scenarios and $R_{d,\gamma^P_{i(d),j(d)}}$ is the profit of satisfying request d with fee level $l = \gamma^P_{i(d),j(d)}$, as described in Section 4.3. Combining and maximizing the expressions in Equations (5.1) and (5.2), we get the *heuristic objective function*, denoted as f_H

$$\max f_H = -\sum_{e \in \mathcal{E}} \sum_{r \in |\gamma^E_e|} C^R T^H_r + \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \left(\sum_{d \in \mathcal{D}^{SR}(\xi_s)} R_{d,\gamma^P_{i(d),j(d)}} \right) \tag{5.3}$$

5.5 Constructing the Initial Solution

The construction heuristic starts with an empty solution, described in Section 5.2, and modifies it to make it feasible. The constructed, feasible solution is provided as the initial solution to the main heuristic framework described in Section 5.1, and thus the initial value of the *CurrentSolution* in Algorithm 1. A pseudocode of the construction heuristic is presented in Algorithm 3.

Algorithm 3 Generation of initial pricing solution

```
function ConstructionHeuristic(γ_∅)
    γ ← γ_∅
    while γ not feasible do
        Choose a carmove cm and an employee e randomly
        if Adding cm to e's task-list is feasible then
            Update γ (or more precisely γ^E) by adding cm to e's task-list
        end if
    end while
    return γ
end function
```

In the remaining part of this section, the construction heuristic is explained with an example solution that will be constructed from an empty solution, $\gamma_\emptyset$. As explained above, the construction heuristic aims to modify the empty solution $\gamma_\emptyset$ to be feasible. As discussed in Section 5.3, the charging constraints, Constraints (4.10), require a certain number of cars to be set to charging within each planning horizon. Constraints (4.10) are the only constraints presented in Section 4.4 which might make an empty solution infeasible. To satisfy these constraints, either employees need to relocate the cars to charging stations or customers need to move the cars to charging stations.

An empty solution has the fee levels of all zone pairs set to the lowest fee. This allows the customers to perform the maximum number of customer-based charging moves, as this pricing solution incentivizes the customers as much as possible to choose the car-sharing option.

The remaining cars in need of charging have to be relocated by the employees. To find a feasible solution, we separate a feasible relocation solution and an overall feasible solution. A feasible relocation solution

represents a solution constrained by all constraints except the charging constraints given by Constraints (4.10). An overall feasible solution, on the other hand, is constrained by all constraints. To find an overall feasible solution, the employees are to perform car-moves with cars in need of charging to charging zones, referred to as charging-moves. A charging-move is considered for each of the employees, and if it can be added to the task-list of an employee and achieve a feasible relocation solution, it is added to the solution γ^E. This procedure of adding charging-moves is repeated until the solution found is an overall feasible solution.

To illustrate this, consider a case with four zones, two employees, and eight cars. One of the zones is a charging zone, and six of the cars are in need of charging. Assume Constraints (4.10) demand that 50% of the cars in need of charging are to be put to charging. Further, assume that in every scenario, only one customer wants to go to a charging zone with a car in need of charging. In this case, two cars still need to be moved to charging stations. To meet this requirement, these cars must be relocated by employees. The construction heuristic then iterates over all car-moves moving a car in need of charging to a charging zone, unless the car is the one moved by a customer. First, employee 1 is assigned car-move 1. This holds for a feasible relocation solution but not an overall feasible solution. Then, employee 1 is assigned car-move 2. This combination is too time-consuming for employee 1, breaking the temporal Constraints (4.7). Car-move 2 is thus assigned employee 2 instead, making it both a feasible relocation solution and an overall feasible solution.

5.6 Adaptive Large Neighborhood Search

Adaptive Large Neighborhood Search (ALNS) is a search algorithm that destroys a solution before rebuilding it to a new solution. An outline of the ALNS can be found in Algorithm 4. The ALNS algorithm, which explores different pricing solutions, is integrated with two different Local Search (LS) algorithms, one for pricing solutions and one for relocation solutions. These LS algorithms are used to improve the solution found by the ALNS. In this thesis, the ALNS heuristic mainly focuses on exploring new pricing solutions, before trying to improve the solution, both for pricing and relocations, by using the integrated LS algorithms.

A set of operators are used to explore the neighborhood of a pricing solution. These are divided into two categories, namely destroy operators and repair operators. The destroy operator breaks down a pricing solution, bringing the solution closer to an empty solution. The repair operator takes the destroyed pricing solution and rebuilds it to a complete solution. The aim of the destroy-repair operation is to explore new pricing solutions in the neighborhood of the current solution. How the different destroy and repair operators work is described in detail in Sections 5.6.1 and 5.6.2, respectively. How the destroy and repair operators are chosen is elaborated on in Section 5.6.3.

5.6.1 Destroy Operators

The destroy operators used in the ALNS are described in this section. The destroy operators take in a complete pricing solution, i.e., pricing solutions where all pairs of zones have an assigned fee. The operators then break down the pricing solution by removing the assigned fees for a predefined number of zone pairs, η. η decides how large a fraction of the solution is to be destroyed in each iteration of the ALNS. A larger value of η will increase the chance of exploring solutions that are very different from the input solution. In this version of the ALNS, there are three different destroy operators to choose from.

Random Removal

Algorithm 4 ALNS of the current solution

```
function ALNS(γ)
    currentSolution = γ
    bestSolution = γ
    visitedSolutions = [currentSolution]          ▷ list of all solutions visited to this point
    temp = calculate initial temperature
    coolingRate = calculate cooling rate
    while Stopping criterion not met do
        newSolution = choose destroy operator and destroy currentSolution
        newSolution = choose repair operator and repair newSolution
        if (newSolution not in visitedSolutions) AND (newSolution accepted by simulated annealing) then
            add newSolution to visitedSolutions
            if newSolution better than currentSolution then
                perform besst improving LS on prices in newSolution
            else
                perform first improving LS on prices in newSolution
            end if
            perform LS on relocations in newSolution
            if newSolution better than bestSolution then
                bestSolution = newSolution
            end if
            add newSolution to visitedSolutions          ▷ newSolution has been updated
            currentSolution = newSolution
        end if
        update temperature based on cooling rate
        if Time for new segment then
            Start new segment
        end if
    end while
    return bestSolution
end function
```

The random removal operator randomly chooses η pairs of zones and removes their assigned fees. Random removal helps diversify the search space and therefore decreases the chance of getting stuck in a local optimum.

Worst Removal

The worst removal operator removes the η zone pairs that contribute the least to the objective value. In combination with a repair operator, this operator will help replace the worst fees with potentially better fees.

To help diversify the search, a determinism parameter, $p_{worst} \geq 1$, is included. The list, L, of zone pairs is sorted in ascending order by the zone pairs' contribution to the objective value. The zone pair at the first index is the zone pair with the fee that contributes the least to the objective value. The zone pair to be removed from the solution and L is found in L at index $i = y^{(p_{worst})} * |L|$, where $y \in [0, 1)$ is a uniformly random number. Consequently, low values of p_{worst} increase the degree of randomness, and a higher value for p_{worst} reduce the degree of randomness in the selection. This process is repeated η times.

Related Removal

The related removal operator removes η zone pair fees that are similar based on certain criteria. Shaw

(1998) first introduced related removal, with the intention of removing similar parts of a solution so that the reconstruction and maintenance of feasibility would be easier. Related removal was also used in the original ALNS implementation of Ropke and Pisinger (2006). In our thesis, the operator is modified to fit the pricing solution of the SE-VReP-DP.

In order to compare different tuples of zone pairs and fees, $t = ((i, j), l)$, a relatedness measure $R(t_1, t_2)$ is used. The relatedness measure examines the travel distance between both the origin, $d^o(t_1, t_2)$ and destination $d^d(t_1, t_2)$ of t_1 and t_2. In addition, the relatedness considers the difference in fees for t_1 and t_2.

Let l_1 and l_2 be the fees of zone pair of fee tuples t_1 and t_2, respectively. The complete relatedness measure between t_1 and t_2 is then given by

$$R(t_1, t_2) = q_o d^o(t_1, t_2) + q_d d^d(t_1, t_2) + q_f |l_1 - l_2|, \tag{5.4}$$

where q_o, q_d and q_f are weights for the origin zone distance, destination zone distance and difference in fees, respectively.

First, a zone pair (t_1) is randomly selected and added to a list, $L_{removed}$, of removed zone pairs. Secondly, a random zone pair from $L_{removed}$ is used to compare to the other zone pairs that are not yet included in $L_{removed}$. The relatedness is calculated and sorted in ascending order in a new list $L_{relatedness}$. The lower the relatedness $R(t_1, t_2)$ is, where t_2 is a zone pair not in $L_{removed}$, the more similar the two pairs are. To allow for some randomness when selecting zone pair from $L_{relatedness}$ to remove and add to $L_{removed}$, a determinism parameter $p_{related} \geq 1$ is included. This works in the same manner as for the worst removal operator. The process of choosing zone pairs from $L_{removed}$, calculating relatedness with other zone pairs, sorting, and finally selecting the zone pair to remove is repeated η times.

5.6.2 Repair Operators

This section describes the repair operators used in the ALNS. The repair operators take an incomplete solution as input and return a complete solution, where all zone pairs have an assigned fee. The repair operators go through the η zone pairs with missing fees and assign these a new fee. The assignment of fees is done according to one out of three different repair operators described in this section.

Random Insertion

The random insertion operator goes through the set of η zone pairs with missing fees, and randomly assigns it a new fee l from the set of fees $\mathcal{L}$. This operator helps diversify the search and can help mitigate the risk of getting stuck in a local optimum.

Greedy Insertion

The greedy insertion operator greedily chooses the fee that increases the heuristic objective value the most from the previous fee for each of the η destroyed zone pairs. The order of reassigning new fees to these zone pairs is random. For each zone pair that have been destroyed, this operator calculates the increase in objective value for each possible fee for the zone pair and chooses the best fee to assign to the zone pair. This operator is an efficient way of finding potentially improving fees for the destroyed zone pairs.

Random Greedy Insertion

The random greedy insertion operator includes some degree of randomness to the greedy process described in the greedy insertion operator. Similar to worst and related removal, there is a determinism parameter $p_{greedy} \geq 1$. For each destroyed zone pair, the random greedy operator evaluates the possible fees, and inserts the fee and the heuristic objective value for using that fee into a list $L_{obj.val}$. The list is sorted in descending order, so the fee with the highest objective value is at the first index. The fee to be assigned to the zone pair is then chosen in the same manner as in worst and related removal, where a value of p_{greedy} close to one increases the randomness, and a high value of p_{greedy} increases the chance of choosing the fee with the highest objective value. This process is repeated for each of the η destroyed zone pairs.

This operator combines the increased diversification from random insertion with the increased possibilities of choosing fees that positively affect the objective value from the greedy insertion operator.

5.6.3 Choosing Destroy and Repair Operators

Let Ω^D and Ω^R be the set of all destroy operators and repair operators, respectively, and let $\Omega^O = \Omega^D \cup \Omega^R$. A pair containing one destroy operator, $d \in \Omega^D$, and one repair operator, $r \in \Omega^R$, is chosen for each iteration of the ALNS. The selection is based on the *roulette wheel selection principle* where both destroy and repair operators are drawn from a weighted probability distribution of all possible destroy and repair operators. Each of the destroy and repair operators is associated with a weight, w_o, where $o \in \Omega^O$. The weights assigned to each operator form the distribution. The probability of the destroy operator $d \in \Omega^D$ being selected is

$$P(d) = \frac{w_d}{\sum_{i \in \Omega^D} w_i}, \tag{5.5}$$

and the probability of repair operator $r \in \Omega^R$ being selected is

$$P(r) = \frac{w_r}{\sum_{i \in \Omega^R} w_i}. \tag{5.6}$$

5.6.4 Acceptance Criterion

When a new solution is constructed from the current solution by the destroy and the repair operator, this solution is evaluated to see whether or not it is accepted. In this version of the ALNS, we are using Simulated Annealing (SA) as the acceptance criterion. This is the most commonly used acceptance criterion within the ALNS framework (Santini et al., 2018). One key element in SA is the *temperature*, τ. The idea behind SA is to explore when the temperature is high, and to exploit when the temperature is low. Exploration means that more solutions will be accepted to widen the neighborhood search. In contrast, exploitation means that we take advantage of the already found solutions and make fewer visits to faraway and worsening neighbor solutions.

When a new solution, γ_{new}, is found after using one destroy operator and one repair operator, there is a probability that γ_{new} will be accepted. When using SA, this probability is given by

$$P(accept(\gamma_{new}|\gamma_{prev})) = \begin{cases} exp(-\frac{f(\gamma_{new}) - f(\gamma_{prev})}{\tau}) & , f(\gamma_{new}) - f(\gamma_{prev}) =< 0 \\ 1 & , else \end{cases} \tag{5.7}$$

where γ_{prev} is the previously accepted solution by the ALNS and τ is the temperature. Equation (5.7) gives a probability of one if γ_{new} is better than γ_{prev}, and a probability in the range [0,1) if not.

The initial temperature depends on the problem, and can therefore not be set in advance. Ropke and Pisinger (2006) address the problem of setting an initial temperature, and draws the conclusion that accepting a solution which is $\delta \cdot 100\%$ worse in $p\%$ of the cases is a good initial temperature. With these values inserted into Equation 5.7 we get

$$p = e^{-\frac{(1-\delta) \cdot f(\gamma) - f(\gamma)}{\tau}}. \tag{5.8}$$

Setting $\tau = \tau_{init}$ to find the initial temperature, and using cost value $\delta = \delta_{init}$ and probability $p = p_{init}$ into Equation (5.8), we express the initial temperature as $\tau_{init} = \frac{\delta \cdot f(\gamma_{init})}{\ln y\%}$.

The temperature will be strictly sinking, at a constant thermodynamic speed, as the temperature is multiplied by a cooling rate, ζ, at every iteration of the ALNS. The temperature evolves in the following manner

$$\tau_{n+1} = \tau_n * \zeta, \tag{5.9}$$

where $\zeta \in (0, 1)$ is the cooling rate and n is the iteration number, making $\tau_0 = \tau_{init}$. It was shown by Nourani and Andresen (1998) that the constant thermodynamic speed cooling schedule was superior to the compared cooling schedules. The purpose of the cooling rate is to keep the temperature sinking, i.e., τ is decreasing with the iterations.

Folkestad and Klev (2021) set the cooling rate such that the temperature would reach a predetermined value, a final temperature denoted τ_{final}. The final temperature is decided upon the same principles as the initial temperature, based on Equation 5.8. The value of τ_{final} is the value that is accepting a δ_{final} cost of the best solution found with a probability of p_{final} in the final iteration, i.e., $\tau_{final} = \frac{\delta_{final} \cdot f(\gamma_{best})}{\ln(p_{final})}$. This gives rise to a cooling rate with the expression of $\zeta = 1 - (\tau_{final}/\tau_{init})^{(1/n)}$.

The temperature is steadily decreasing due to the cooling rate. The temperature can regain its value by using the technique called reheating. The purpose of reheating is to frequently alternate between exploration and exploitation during the entire heuristic search. In this implementation of the ALNS, we define a reheat strategy, $\tau_{strategy}$. If reheating is applied, $\tau_{strategy} = reheat$, the temperature will follow the cooling rate with n set to the number of iterations within a run of the ALNS, i.e., $n = I^{ALNS}$. The initial temperature, τ_{init}, is then recalculated after $n = I^{ALNS}$ iterations. If reheating is not applied, $\tau_{strategy} = no-reheat$, the temperature will follow the cooling rate with n set to the number of iterations ALNS will run in total when combining every run called by the main algorithm, i.e., $iterations = I^{ALNS} \cdot I^{main}$.

The acceptance criterion is extended beyond the Simulated Annealing with the addition of a *tabu list*, named *visitedSolutions* in Algorithm 4. The tabu list keeps track of already explored and evaluated solutions, making

sure not to accept any previously discovered and accepted solutions.

5.6.5 Weights and Segments

Running the ALNS is an iterative process. This process is divided into segments, defined as a batch of iterations within the ALNS. During a segment, data regarding performance for each operator is collected. When a segment is over, we update the operator weights based on these data, as further described in Section 5.6.7. The purpose of a segment is to update the weights at regular intervals, after the performance of the destroy and repair operators have been evaluated. This process defines the adaptive part of the *Adaptive* Large Neighborhood Search.

5.6.6 Stopping Criterion

The single stopping criterion of the ALNS is based on a maximum number of iterations. After preliminary testing using different criteria, we found that the ALNS performs best with a maximum number of iterations rather than a time limit or an improvement threshold. The ALNS is run multiple times with different initial solutions, as presented in Section 5.1 and Algorithm 1.

5.6.7 Adaptive Weight Adjustment

The weighted probability distribution described in Section 5.6.3 is a dynamic distribution, making the ALNS an adaptive heuristic. The weights for each destroy and repair operator are adjusted after a full segment of the ALNS, as described in Section 5.6.5. The adjustment is based on the performance of each operator during each iteration in the segment. At initialization, the first time the first ALNS is run, all weights are assigned a value of one, making the probability of choosing an operator equal for all operators. During one iteration of a segment, the chosen operators are assigned a reward, ρ, based on their performances. The rewards are accumulated over the iterations in a segment, resulting in an overall segment evaluation denoted as π. An operator can receive different rewards based on different criteria given by Table 5.3. These criteria are whether an operator contributes to a new global best solution, a new local best solution, or accepts a non-improving local solution. The latter is included in the list of rewards, as the operator contributes to diversifying the search when accepting a non-improving solution.

Table 5.3: Rewards for destroy and repair operators

Parameter	Reward criterion	Description
ρ_G	$f_H(\gamma') > f_H(\gamma_{best})$	The new solution is a new global best.
ρ_L	$f_H(\gamma') > f_H(\gamma_{current})$	The new solution is a new local best, but not a new global best.
ρ_N	$f_H(\gamma') < f_H(\gamma_{current})$, Accepted	The new solution is non-improving, but is accepted.

At the end of a segment, the weights are updated for each destroy and repair operator by Equation (5.10)

$$w_o = w_o(1 - \alpha) + \alpha \frac{\pi_o}{\sigma_o} \tag{5.10}$$

Here, w_o is the weight for destroy or repair operator $o \in \Omega^O$, π_o is the accumulated reward for operator o, and σ_o is the number of times operator o is used in the most recent segment. α is a parameter between zero and one, used to control how large an impact the evaluation of an operator in the most recent segment has on the operator's weight. If α is closer to one, the weights are highly determined by the associated operators' success in the most recent segment. However, if α is closer to zero, the weights remain rather stable, only adapting slightly based on the evaluation from the most recent segment.

5.7 Local Search

Integrating the exploratory principles of ALNS with the exploitative principles of LS gives rise to an algorithm discovering large parts of the solution space. When performing a local search, the current solution is improved by finding better solutions in the close neighborhood of the current solution. A neighbor of a solution is a new solution derived from the original solution by making only minor modifications. In the ALNS introduced in Section 5.6 there are two different LS components, one to improve the relocation solution, γ^E, and one to improve the pricing solution, γ^P. The two LS components use their proper Local Search Operators (LSOs), but the main structure of the LS algorithm is the same for both components. This section provides a brief description of the LS algorithm, before Sections 5.7.1 and 5.7.2 describe the LSOs used in the pricing LS (LSP) and the relocation LS (LSR), respectively

The principles and structure of an LS algorithm is presented in Algorithm 5. The LS algorithm's most important functionality is its LSOs. These perform the search of the neighborhood, and elect the neighbor that will be further explored. The structure around these LSOs consists of simple elements. The LS will use each of the relevant LSOs and perform a search using the current LSO until a stopping criterion is met. The stopping criterion of an LSO in a time limit of T^{LSO} and whether the LSO fails to find a better solution in the current neighborhood. If either of the criteria is met, the algorithm moves on to the next LSO. As long as at least one of the LSOs is able to find a better neighbor, the or the time limit of T^{LS} is not reached, the *stopping criterion of the LS is not met*. This means the search restarts exploring the neighborhood of the recently found (improved) solution with all the LSOs. When non of the LSOs are able to find better neighbors, the LS has come to an end, and it returns the best found solution.

The inputs to the LS are the first solution to be explored and a strategy. The strategy can be either *best improving* or *first improving*, and it applies to all LSOs. With a *best improving* search, the entire neighborhood of the current solution γ is evaluated, and the best out of the neighbors is chosen as the new value of γ. With a *first improving* search, the neighborhood of a solution γ is evaluated until a better neighbor is found, after which γ is updated. Indifferent of the strategy, this process is repeated until a *stopping criterion of the LSO is met*

5.7.1 Local Search in Pricing Solutions

Eilertsen et al. (2021) concluded that the pricing was a dominating factor in finding the optimal solution, as opposed to relocations. Therefore we incorporate an LS on pricing the solution (LSP) to further improve the ALNS heuristic. This section describes the LSO used in the LSP, namely the *price move operator*.

Price Move Operator

To create the neighborhood of a solution, we first need to set the conditions of what it means to be a neighbor.

Algorithm 5 General local search algorithm

```
function localSearch(γ, strategy)
    LSOs = set of LSOs in LS
    while stopping criteria of the LS is not met do
        for LSO in LSOs do
            while stopping criterion of the LSO is not met do
                γ = LSO.search(γ, strategy)
            end while
        end for
    end while
    return γ
end function
```

Recall that the pricing solution consists of a fee between all zone pairs. A solution is said to be a neighboring solution if all fees but one remain the exact same. Thereby, for a certain solution, one finds the neighbors by altering exactly one of the fees. The *price move operator* represents the neighborhood as a list of zone pairs. The list is constructed in a random order to broaden the search. The operator iterates over the list, testing different fees for each zone pair. With a *first improving* search strategy, the iteration stops when an improving solution is found. With a *best improving* search, the search would span $|\mathcal{I}|^2 \cdot |\mathcal{L}|$ zone pair fees, registering the best found solution.

5.7.2 Local Search in Vehicle Relocation

After performing both an ALNS and an LS on the pricing solutions, the heuristic attempts to further improve the solution with an LS on the relocation solution (LSR). Such a search could increase the profit in existing solutions, as well as guide the pricing search into new neighborhoods. This section describes the LSOs used by the LSR, namely *add move*, *remove move* and *inter swap*.

Add Move and Remove Move

The neighborhood for a *move* operator is found by adding or removing a car-move from the current relocation solution γ^E. Whether the operator will add or remove a car-move from the current solution is specified through a variable. The *add move* and the *remove move* operators investigate whether it is profitable to add or remove a car-move from either one of the employees' task lists, respectively. The evaluation is performed by calculating the objective values of the new solutions.

Inter Swap

The *inter swap* operator investigates the effect of swapping car-moves present in the solution between employees. This will not change the objective value immediately, but may reduce the relocation time for each employee. Imagine the problem is solved by having two employees, and the first-stage planning horizon is set to 60 minutes. The current relocation solution consists of four tasks, two tasks performed by each employee. Both employees use 60 minutes to complete the tasks, and can therefore not perform additional car-moves. If they were to swap one of their car-moves with the other, they would both use 40 minutes, as the relocation routes now are more time efficient. This way, both employees can perform additional car-moves, which may

lead to improved solutions.

Notice that we could include a similar *intra swap* procedure to optimize the route for each employee. However, as each employee has a maximum number of tasks of $|\mathcal{M}|$, we automatically order the task lists when a car-move is added or removed. This process is a simple permutation problem of minor scale when $|\mathcal{M}|$ has a low value, as we see it will have in Section 6.2, making an *intra swap* operator excessive.

5.8 Complexity Reduction

The SE-VReP-DP is a computationally complex problem with a large solution space. In order to increase the efficiency of the heuristic search, size reduction techniques and computational simplifications have been utilized.

5.8.1 Customers

As discussed in Section 4.1, the utility a transportation mode yields for a potential customer is composed of a deterministic and a stochastic part. Therefore a potential customer might have different behavior and preferences in each scenario. This may result in a potential customer preferring car-sharing with different price levels in different scenarios, and in other scenarios preferring a different mean of transportation. To reduce the search space, we attempt to separate the customers into two categories: those considered relevant customers, and those considered irrelevant customers. The different use cases for this division is clarified in Section 5.8.2-5.8.3. A potential customer is considered relevant if she prefers the car-sharing option in at least $\phi_{customer} \cdot 100\%$ of the scenarios, and irrelevant if not. To clarify, the car-sharing option yields the highest utility for a relevant customer for some price level available to the CSO in $\phi_{customer} \cdot 100\%$ of the scenarios.

5.8.2 ALNS Search Space Reduction

To improve the performance of the ALNS heuristic, the destroy and repair operators are only changing the prices of zone pairs that are found relevant. A zone pair is considered relevant if it fulfills three conditions. Recall the definition of relevant customers provided in Section 5.8.1. The first condition is that there is at least one car present in the zone pair's origin zone. This might be due to the car being there in the first place, or that an employee has relocated a car to this zone. The second condition is that there is at least one relevant customer in the zone pair's origin zone. This has the effect of finding a zone where there is demand for cars. The final condition for a zone pair to be relevant is that among the relevant customers, at least one has the zone pair's destination zone as its desired destination. If all these conditions are fulfilled, we consider the zone pair relevant. By only evaluating the relevant zone pairs, we avoid evaluating changes in zone pairs that do not have an impact on the objective value. This reduces the runtime and guides the search in a preferred direction.

5.8.3 LS Search Space Reduction

The LS for pricing solutions, or more precisely the *price move* LSO, uses the same search space as defined for the ALNS, only investigating what we define as relevant zone pairs. The LS for relocations has extended this definition to include which car-moves are considered relevant, when using the *add move* LSO. For each zone, we calculate a cars-to-customers ratio. This ratio is calculated by including only relevant customers in the zone, and all cars originating or currently being in the zone as a consequence of relocation. Recall relevant customers being defined in Section 5.8.1. If a given zone has a cars-to-customers ratio exceeding $\phi_{cars-to-customers}$, all car-moves with the given zone as destination zone are excluded from the relevant car-moves. This reduction in search space makes the algorithm only consider moving cars to areas which do not already have a high density of cars compared to customers.

To restrict the search space even further, the car-moves are filtered based on the time it takes to perform them. If the relocation time of a car-move exceeds $\phi_{relocation-time} \cdot \overline{T}$, where $\overline{T}$ is the planning horizon, the car-move is considered irrelevant. The only exception is if the car to be relocated is in need of charging. This is done to eliminate car-moves assumed to be too time-consuming, and thus unlikely to be part of the optimal solution.

A final effort to reduce runtime is defined through the LSOs themselves. Each LSO has a time limit of T_{LSO} seconds, avoiding a *best improving* search from being a bottleneck in larger instances of the problem. Further, the LS has an overall time limit of T_{LS} seconds. To avoid repeatedly investigating the same part of the search space for T_{LSO} seconds, the search space is randomly ordered at the beginning of each LS.

Chapter 6

Test Instances

To evaluate the performance of a commercial MIP-solver and an ALNS heuristic, we use test instances representing different dimensions and states of the car-sharing system. The test instances vary in the number of zones, vehicles, employees, and customers. Further, the different instances represent different geographical distributions of vehicles, employees, and zones. A Python script generates each test instance and distributes the mentioned elements. The script further generates model-specific information, such as car-moves, customer utility and requests through scenarios, and fee levels. The test instances are generated in a somewhat similar manner as done by Eilertsen et al. (2021), using data from Oslo situated CSO Vybil where applicable. Further, as Folkestad and Klev (2021) had direct access to data provided by Vybil, we rely on certain of their reflections when making similar decisions.

Section 6.1 provides an overview of the test instances used in this thesis. Further, Section 6.2 presents parameters used in the generation of the different test instances and their values. A description of each test instance component is provided in Section 6.3, before Section 6.4 discusses the rationale behind the choice of different test instances.

6.1 Overview

A test instance constitutes a number of zones, representing fractions of the operating area. Each zone contains a certain number of vehicles, employees, and customers and may be defined as a charging zone if it contains a charging station. A test instance further contains a number of vehicles, each with an origin zone and a start time indicating when it is available for relocations. A vehicle may be in need of charging and, if so, restricted to be relocated to charging zones only. A test instance also comprises a number of employees, each with an origin zone and a start time for when it first can perform a relocation. Finally, a test instance contains a number of customers, each with an origin zone and a desired destination zone. The customers further have a scenario-dependent utility, defining the highest fee accepted for it to be willing to rent a vehicle.

The different types of test instances used throughout the computational study are shown in Table 6.1. A test instance type is noted as X-Y-Z, where X is the number of zones, Y is the number of vehicles, and Z is the number of employees. Each test instance type contains, on average, ten potential customers per zone

and is solved using 25 scenarios. The reasoning for the setting of these values, as well as the values for other parameters, is further described in Section 6.2. The test instance types are divided into categories based on their sizes, being either small, medium, or large. For each type of test instance, three different test instance versions are generated. A separate set of small, medium, and large test instances are used for the parameter tuning in Section 7.2. The remainder of the analyses conducted in the computational study in Chapter 7 are based on separate sets of test instances for each analysis, unless otherwise stated.

Table 6.1: The test instance types used throughout the computational study. For each type of test instance, three different test instance versions are generated. The test instance types are categorized by size.

Test Instance Type	Zones	Vehicles	Employees	Customers per Zone	Scenarios	Size
5-4-1	5	4	1	10	25	
6-6-1	6	6	1	10	25	Small
8-8-1	8	8	1	10	25	
10-9-2	10	9	2	10	25	
15-12-2	15	12	2	10	25	Medium
20-15-2	20	15	2	10	25	
30-20-2	30	20	2	10	25	
40-25-2	40	25	2	10	25	Large
50-30-2	50	30	2	10	25	

6.2 Predefined Parameters

Certain parameters are set prior to generating the different test instances. These parameters are referred to as predefined parameters and include common parameters equal for all instances and parameters specifically set for each unique test instance.

The common parameters are displayed in Table 6.2, and include set sizes and problem defining parameters. The number of scenarios is set to 25, based on an extensive stability analysis by the use of a Sample Average Approximation conducted by Eilertsen et al. (2021). The probability related to the occurrence of each scenario is equal. Eilertsen et al. further argue that five different fee levels ensure satisfactory flexibility to a CSO and that the set of fees $\{-40, -20, 0, 20, 40\}$ of currency NOK has a significant spread to ensure impact on the individual customer utilities. $|\mathcal{L}|$ and $\mathcal{L}$ are therefore set accordingly. A planning horizon of $\overline{T} = 120$ minutes is set to fit a realistic situation, as the problem is intended to be solved every hour. This way, a CSO can provide employee routes and schedules for the coming hour and set prices for the subsequent hour. Notice that this way, the lengths of the first and second stages presented in Section 4.4 are 60 minutes each. The initial capacities at each charging station are set to three, and the cost of relocation per minute is set to NOK 0.43 per minute, as calculated by Eilertsen et al. (2021). The customer sensitivity parameters are set as done by Pantuso (2020) and Eilertsen et al. (2021). The different transportation mode prices are set to NOK 6 per minute for car-sharing, and NOK 38 for a public transportation ticket, as done by Eilertsen et al. (2021). Bicycles are assumed to be free of charge.

Table 6.2: Predefined parameters and set sizes common to all test instances. The notation is similar to the one presented in Section 4.3.

	Notation	**Value**
Number of scenarios	$\|\mathcal{S}\|$	25
Probability of scenarios	$P_s, s \in \mathcal{S}$	$1/\|\mathcal{S}\|$
Number of fees	$\|\mathcal{L}\|$	5
Fees [NOK]	$L_l, l \in \mathcal{L}$	{-40, 20, 0, 20, 40}
Planning horizon [minutes]	$\overline{T}$	120
Initial capacity at charging station	$N_i^{CS}, i \in \mathcal{I}^{CS}$	3
Cost of relocation per minute [NOK]	C^R	0.43
Customer sensitivity to:		
Price (P)	β_k^P	{-7, -19}
Car-sharing travel time (CS)	β_k^{CS}	-1
Pubic transportation travel time (PT)	β_k^{PT}	-2
Bicycle travel time	β_k^B	-2.5
Walking distance	β_k^{Walk}	-3
Waiting time	β_k^{Wait}	-6
Prices per trip [NOK] using:		
Car-sharing (CS)	p_{ijCS}	$6 \cdot T_{ijCS}$ + fee
Public transportation (PT)	p_{ijPT}	38
Bicycle	p_{ijB}	0

Table 6.3: Predefined parameters and set sizes varying for each test instance. The notation is similar to the one presented in Section 4.3.

	Notation	**Value**
Number of charging zones	$\|\mathcal{I}^{CS}\|$	$25\% \cdot \|\mathcal{I}\|$
Number of cars that needs charging	$\|\mathcal{V}^B\|$	$15\% \cdot \|\mathcal{V}\|$
Number of customers	$\|\mathcal{K}\|$	$10 \cdot \|\mathcal{I}\|$
Maximum number of tasks per employee	$\|\mathcal{M}\|$	5

The parameters varying for each test instance are displayed in Table 6.3. The number of charging zones is dependent on the number of zones in the test instance. As Vybil offers 35 parking lots, out of which nine are equipped with charging stations[1], we apply a ratio of charging zones per zone of 25%. Further, Folkestad and Klev (2021) claim the number of cars used by Vybil in need of charging is between 10% and 15% of the

[1]https://www.vy.no/alt-om-reisen/andre-transportmidler/vybil/vybil-parkeringsplass, accessed 2022-26-5

fleet. We fix this number to be 15% to handle the worst case. Eilertsen et al. (2021) conducted a complexity analysis concerning how the runtime and result quality depended on the test instance types. Parts of their findings concluded that having, on average, ten potential customers per zone yielded an acceptable trade-off between runtime and solution quality. The size of the set of customers, $|\mathcal{K}|$ is therefore set to $10 \cdot |\mathcal{I}|$. Finally, it is worth noticing that the maximum number of tasks per employee, $|\mathcal{M}|$, only has an effect when implementing the MIP to be solved by a commercial solver. A lower value of $|\mathcal{M}|$ reduces the complexity of the MIP in terms of numbers of variables and constraints. Folkestad and Klev (2021) claim that reducing the value of $|\mathcal{M}|$ from 6 to 4 reduced the computation time with 64.2% without affecting the objective value. Preliminary testing with the ALNS heuristic has yielded relocation solutions with car-moves rarely being of a length less than 15 minutes per relocation. Therefore the value of $|\mathcal{M}|$ is adjusted to follow the length of the first stage of the planning horizon, which is 60 minutes. This yields an optimistic upper bound of $\frac{60}{15} =$ four relocations per employee. To handle special cases, we adjust the value up to five.

6.3 Test Instance Components

The test instances are constituted by different components aiming to represent realistic states and planning situations. These are an operating areas in terms of zones, charging stations, employees, cars, and customers. To approximate the case of Vybil as much as possible, we use relevant data either directly from their published sites or indirectly through reflections made by Folkestad and Klev (2021), as they had access to data provided by Vybil. Where no applicable data is available, we make qualified estimates. Subsections 6.3.1, 6.3.2, 6.3.3 and 6.3.4 describe the operating area, employees, cars and customers, respectively.

6.3.1 Operating Area

The operating area is set to the city of Oslo, Norway. This area is divided into squares, called zones, with dimensions of either 800x800 meters or 400x400 meters, depending on the zone's location and its surroundings. Figure 6.1 displays a map of Oslo divided into a grid representing the zones used in the thesis. The choice of size for the zones follows the reasoning of Folkestad and Klev (2021) that this is sufficiently large to support the assumption that there is unlimited parking space within a zone. When generating test instances of different sizes, the zones used in an instance are uniformly drawn from the grid.

When generating test instance components as cars, customers and employees, we use the same technique as Pantuso (2020) to distribute the instance components within the operating area. The probability of an instance being located in zone i, π_i, is calculated based on the zone's distance to the city center of Oslo. Let this distance be denoted as d_i. The city center is set to Oslo Central Station, and the probability is described by Equation 6.1

$$\pi_i = \frac{\gamma_i \cdot d_i}{\sum_{j \in \mathcal{I}} \gamma_j \cdot d_j}, \tag{6.1}$$

where $\gamma_i = e^{-\alpha \cdot \Delta_i}$ with $\alpha \in [0, 1]$ and $\Delta_i = d_i - \frac{1}{|\mathcal{I}|} \sum_{j \in I} d_j$. When α increases, Equation 6.1 makes it more probable to have instances located in zones closer to the city center. This probability is used when assigning the origin location of cars and employees, and also the origin and destination of a customer request.

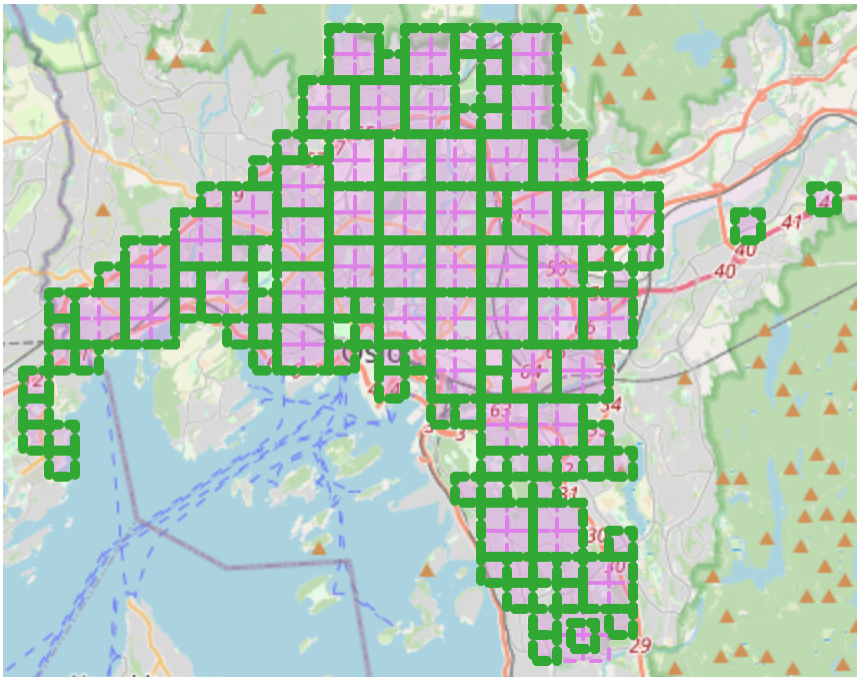


Figure 6.1: The operating area set to Oslo, divided into a grid of 113 zones of sizes 800x800 meters or 400x400 meters, defined by the green lines. The figure is generated using data provided by Folkestad and Klev (2021).

6.3.2 Employees

Employees are defined by their origin location, the maximum number of tasks they can perform, and their earliest start time for performing relocations. The origin locations of the employees are selected by Equation (6.1). The maximum number of tasks for an employee is predefined through $|\mathcal{M}|= 5$, as seen in Table 6.3.

The earliest start time for an employee, T_e^{SO}, is a particular case. The rationale for having a start time is to capture the situation where an employee is in the middle of a relocation at the beginning of the planning horizon. The start time is drawn from a weighted distribution ranging from 0 to 15, where values closer to 0 are more likely to be drawn. This distribution is based on the lower relocation time-bound of 15 minutes found in the preliminary testing of the ALNS heuristic.

6.3.3 Cars and Car-Moves

A car is defined by its origin location, whether or not it is in need of charging, its association to a list of car-moves, and its earliest start time available for relocation. A car is initially located in a zone according to Equation 6.1. Further, a uniformly drawn subset of all cars is defined as cars in need of charging. The size of the subset is given as a predefined parameter in Table 6.3. However, if a car is located in a charging zone, it will never be defined as in need of charging. This is to ensure only cars available for relocation are generated in the test instances to address the problem more precisely.

The set of possible car-moves for a car v, $\mathcal{R}_v$, include car-moves to all zones to which car v can be relocated. If a car v requires charging, $v \in \mathcal{V}^B$, only car-moves to charging zones are allowed. For sufficiently charged

cars, $\mathcal{R}_v$ includes all zones other than the car's origin zone. The earliest start time for relocation of a car associated with car-move r, T_r^{SC}, is drawn from the same distribution as is the earliest start time for an employee.

6.3.4 Customers

Each customer wants to travel from its origin zone i to a given destination zone j. The origins and destinations are given by Equation (6.1) and are selected similarly to the location of cars and employees. To model customer behavior, we follow the approach of Pantuso (2020) and use the Logit choice model. This is due to his problem being quite similar to the pricing aspect of our problem and that the alternative approach of Esmaeilpour et al. (2016) using reinforcement learning requires customer-specific data, which we do not possess. When a customer is exposed to different transportation modes, i.e., car-sharing, public transport, and bicycle, we let these modes yield different utilities for the customer, as described in Section 4.1. The utilities are used to select the preferred transportation mode for a customer, allowing us to model whether a customer wishes to use car-sharing, given the relocation and pricing strategies chosen.

As discussed in Section 4.1, the CSO is operating with imperfect information when considering the customers' behavior. To handle this imperfect information, the utility a customer k achieves by using transportation alternative x is described as $U_{kx} = V_{kx} + \tilde{\xi}_{kx}$, where V_{kx} is a deterministic part and $\tilde{\xi}_{kx}$ is a stochastic part.

The deterministic part is composed of measurable factors connected to the fare with a certain transportation mode and the customer sensitivity to these factors. Let x be a transportation mode offered to a customer, being either car-sharing (CS), public transportation (PT), or bicycle (B). The deterministic utility for a specific instance $x \in X = \{CS, PT, B\}$ for a trip between zones i and j is determined by different deterministic factors. These are the price of, the walking time to, the waiting time for, and the travel time for using the mode of transportation for the current fare. Let these parameters be defined as p_{ijx}, T_{ijx}^{Walk}, T_{ijx}^{Wait} and T_{ijx}^{X}, respectively. p_{ijx} is given in Table 6.2, while the remaining factors are described below. Further, each customer is assigned sensitivity parameters to determine to which degree its utility depends on the corresponding characteristics. These parameters are equal for all customers, except the one concerning price. To model different social levels of customers, the price sensitivity is set either higher or lower, each with equal probability. The sensitivity parameters for a customer k to the corresponding deterministic factor is denoted as β_k^X, listed in Table 6.2.

In order to handle the stochastic part of the utility, the Logit choice model is used. The Logit model is the most widely used model for decision making, when comparing to linear and Probit models (Bierlaire and Sharif Azadeh, 2016). When using the Logit model, we are trying to imitate the real stochastic value $\tilde{\xi}_{kx}$ by making multiple independent and identically Gumbel distributed draws. This gives the expression for a single draw $\xi_{kxs} = -\sigma \cdot \ln(-\ln(p))$, where σ is the standard deviation of all of the deterministic parts of the customers' utility for the different transportation modes, and $p \in [0, 1]$ is a uniformly drawn number. By generating these values for all customers in all scenarios for all transportation modes available, we are imitating the real value $\tilde{\xi}_{kx}$ for customers k's stochastic part of the utility of using an instance of a transport mode x.

$$\begin{aligned} F_{kxs}(p_{ijx}, T_{ijx}^{CS}, T_{ijx}^{PT}, T_{ijx}^{B}, T_{ijx}^{Walk}, T_{ijx}^{Wait}) = \beta_k^P \cdot p_{ijx} + \beta_k^{CS} \cdot T_{ijx}^{CS} + \beta_k^{PT} \cdot T_{ijx}^{PT} \\ + \beta_k^B \cdot T_{ijx}^{B} + \beta_k^{Walk} \cdot T_{ijx}^{Walk} + \beta_k^{Wait} \cdot T_{ijx}^{Wait} + \xi_{kxs}. \end{aligned} \tag{6.2}$$

Using the notation described above, we model the utilities through the utility function in Equation 6.2. When

a potential customer k's requests might be realized at the beginning of the second stage, Equation 6.2 is used to calculate the customer's utility for all instances x of transportation mode X available to fulfill its trip from zones i to j. Here, $T^X_{ijx} = 0$ if x is not corresponding to the respective transportation mode X. The value of the βs weighs the transportation characteristics, determining the significance of each characteristic to the customer. This calculation is done for all realizations of ξ_s, i.e., for all scenarios. The potential customer prefers the transportation mode yielding the highest utility. If its utility is highest for car-sharing, the potential customer becomes a customer, and a request is generated for the specific scenario.

The calculation of a customer's utility for car-sharing depends on the car's pick-up and drop-off fees. Suppose the car-sharing utility is the highest among the transportation modes for at least one fee level $L_l, l \in \mathcal{L}$ for a realization of ξ_{ijxk}. In that case, a customer request $d \in \mathcal{D}(\xi_s)$ is generated. $l(d)$ is the highest fee level acceptable to the customer. If none of the available fee levels make the customer choose the car-sharing service, the customer will choose another mode and, consequently, not generate a request. This is observed in Algorithm 6.

Based on the set of requests $\mathcal{D}(\xi_s)$, the subsets $\mathcal{D}^{CD}(\xi_s)$ (requests with charging zone destinations), $\mathcal{D}_d(\xi_s)$ (requests which have precedence over request d) and $\mathcal{D}(\xi_s)_{ij}$ (requests from zone i to zone j) are created. The profit generated by satisfying a request is denoted as R_{dl} and represents the revenue and cost related to customers using the cars. The revenue is constituted by multiplying the per-minute fee of the car-sharing service with the travel time of request d and adding the combined pick-up and drop-off fee l associated with the origin and destination of request d. The cost is, as with relocations, C^R per minute. Notice that the actual, combined pick-up and drop-off fee l is variable-dependent, so the parameter R_{dl} is generated for all possible fees l.

6.3.5 Distances and Travel Times

Equation (6.2) relies on several distance and time parameters. We have used Google Maps API *Distance Matrix*[2] to generate T^X_{ijx}. The centers of each square-shaped zone i and j are used as coordinates, and the transportation modes X selected are public transportation, bicycle, car, and walking. The API generates a matrix containing travel times when selecting the shortest path. The walking distance to each transportation mode from the customers' origin, T^{Walk}_{ijx}, is randomly chosen from the interval $[0, d^c_i]$, where d^c_i is the walking distance from the center of zone i to the edge of zone i. The waiting time for using transportation mode instance x, T^{Wait}_{ijx}, is generated as follows. If x is a vehicle provided by the CSO, the waiting time is randomly chosen from the interval $[0, 4]$. This is based on the assumption that customers only wait for a car if another customer is about to park the car, which is assumed to take four minutes. The waiting customer is not aware that another customer is parking the car. However, we assume the waiting customer receives information in, i.e., an app, that the car is available first after the drawn waiting time. If x is an instance of public transportation service, the waiting time is randomly chosen from the interval $[0, 10]$. This is derived from Oslo's public transportation company Ruter and their promise of six bus departures per hour in Oslo[3], yielding a maximum of ten minutes waiting time. If x is an instance of bicycles, it is assumed to be zero waiting time.

T^X_{ijx} constitutes the main component in the relocation time parameter, T^H_r. The relocation time includes the employee's travel time from its location to the car, the time it takes to drive the car to the relocation destination, and the time used for parking or initiating charging. T^X_{ijx} gives the two former components, while the parking and initiation of charging are estimated to be five and four minutes, respectively. These estimations are based on Folkestad and Klev (2021).

[2] https://developers.google.com/maps/documentation/distance-matrix/overview, accessed 2021-11-10

[3] https://ruter.no/contentassets/4c04543f70f443ba84550d4d1c8384fb/ruters-veileder-for-planlegging-av-linjenettet.pdf, accessed 2021-11-10

Algorithm 6 Generation of Customer Requests $\mathcal{D}(\xi_s)$ in scenario s.

input : ξ_s
$\mathcal{D}(\xi_s) \leftarrow \emptyset$
for customer $k \in \mathcal{K}$ **do**
  $i \leftarrow i(k), j \leftarrow j(k)$ ▷ $i(k)$ and $j(k)$ are the origin and destination of customer k
  $l^{MAX} \leftarrow -\infty$ ▷ The highest pick-up and drop-off fee l at which customer k chooses car-sharing
  v instance of CS
  u instance of PT
  w instance of B
  for pick-up and drop-off fee level $l \in \mathcal{L}$ **do**
    $p_{ijv} \leftarrow 6 \cdot T_{ijv} + \mathcal{L}_l$ ▷ calculate the price of a car-sharing ride
    $U^{CS} \leftarrow F_{kv}(p_{ijv}, \pi^1_{ijv}, ..., \pi^N_{ijv}) + \xi_{kvs}$ ▷ calculate utility of car-sharing
    $U^{PT} \leftarrow F_{ku}(p_{iju}, \pi^1_{iju}, ..., \pi^N_{iju}) + \xi_{kus}$ ▷ calculate utility of public transportation
    $U^{B} \leftarrow F_{kw}(p_{ijw}, \pi^1_{ijw}, ..., \pi^N_{ijw}) + \xi_{kws}$ ▷ calculate utility of bicycle
    if $U^{CS} > \max(U^{PT}, U^B)$ **then**
      $l_{MAX} \leftarrow l$
    end if
  end for
  if $l_{MAX} > -\infty$ **then** ▷ There exists a pick-up and drop-off fee l at which k prefers car-sharing.
    $d \leftarrow |\mathcal{D}(\xi_s)|+1$ ▷ A request is generated.
    $\mathcal{D}(\xi_s) \leftarrow \mathcal{D}(\xi_s) \bigcup \{r\}$
    $i(d) \leftarrow i$
    $j(d) \leftarrow j$
    $k(d) \leftarrow k$
    $l(d) \leftarrow l^{MAX}$
  end if
end for
return $\mathcal{D}(\xi_s)$

6.4 Rationale Behind the Test Instances

The chosen test instance sizes span from small to large. Following is a brief rationale behind the number of zones, employees, and cars chosen for the different test instance sizes. Reflections around the number of customers and the number of scenarios are already provided in Section 6.2.

Folkestad and Klev (2021) refer to travel data acquired from Vybil when they reason why test instances of 50 zones are considered reasonable sizes for real-sized instances. This is, among others, based on filtering on average visits per zone during a day using historical travel data. Further, Eilertsen et al. (2021) analyze how the performance of a commercial solver depends on test instance types and conclude that instances with more than 15 zones are far too challenging to solve within reasonable time. Therefore the chosen test instances in this thesis span from small instances with five zones up to large instances with 50 zones, the latter representing real-sized instances. This enables comparison of the ALNS heuristic to a commercial solver on the small- and medium-sized instances and performance analyses of the ALNS heuristic on reasonably large instances.

The number of cars approximately follows the number of zones without charging stations in each test instance. This is based on the analysis of data acquired from Vybil made by Folkestad and Klev (2021). Recall that an instance of type $5-4-1$ has five zones, where Table 6.3 informs us that one of the zones is a charging zone. Therefore, the number of cars is four for this test instance.

Vybil uses two employees for relocating cars. This is the case also for the medium and large-sized instances used in this thesis. However, for the small instances with up to eight cars, we only use one employee. This is because the size of the problem makes an extra employee redundant.

Chapter 7

Computational Study

This chapter presents the computational study of the SE-VReP-DP and the ALNS heuristic proposed to solve it. The test instances described in Chapter 6 form the basis for all tests. In real-life, the problem is intended to be solved on a frequent basis. Using the predefined parameters discussed in Chapter 6, with a planning horizon of two hours, it would be reasonable to solve the problem every hour. A CSO would run the heuristic on an instance representing the current situation, preferably multiple times in parallel. This is to handle the randomness in the ALNS heuristic, allowing the CSO to select the best solution among the different runs. The solution would indicate which car-moves to be performed by which employee in the coming hour and what prices to set for the hour after. As we are solving the instances for the coming planning horizon of two hours, we need to have the results at the start of the planning horizon. Because of this, we are considering 600 seconds, i.e., ten minutes, as a reasonable amount of time for solving the problem in real life. The purpose of the computational study is to investigate the ALNS heuristic's potential for such an application in a real-life car-sharing system and to evaluate the overall performance of the ALNS heuristic.

First, Section 7.1 discusses the testing environment. In Section 7.2 the parameter tuning process of the ALNS heuristic is described. Sections 7.3 and 7.4 present the performance evaluation of the ALNS heuristic and the comparison of the heuristic to the commercial solver Gurobi, respectively. Lastly, Section 7.5 discusses managerial insights from a short-term perspective before 7.6 analyzes the long-term effects of dynamic pricing through simulation.

7.1 Testing Environment

Table 7.1 presents the hardware and software used for implementing and testing the model and the ALNS heuristic. We used *Python 3.9.6* for generating test instances and developing the model, which is solved by the ALNS heuristic or by the MIP solver *Gurobi 9.5.0*. The source code for these implementations can be found here[1].

The computer used for testing the ALNS heuristic and MIP-model is a Lenovo NextScale nx360 M5. It consists of two CPUs with 12 cores each. Each core can process two threads in parallel, allowing it to run 48

[1] https://github.com/jvhenriksen/Vybil˙master, accessed 2022-30-05

Table 7.1: Hardware and software used to implement and test the ALNS heuristic and the MIP model.

Processor	2x 2.3GHz Intel E5 – 12 cores, 24 threads
Memory	64 GB RAM
Operating System	CentOS 7.9.2009
Python Version	Python 3.9.6
Gurobi Version	Gurobi 9.5.0

processes simultaneously. This could reduce the runtime, as scenarios could be processed in parallel. However, all parts of the ALNS heuristic are run in sequence, as preliminary testing showed minimal differences in runtime by running in parallel.

7.2 Parameter Tuning

This section describes the methodology used to determine the parameter values for the ALNS heuristic. The values to be determined are of the parameters discussed in Chapter 5. These impact the overall performance of the heuristic in terms of how and in what solution neighborhood the algorithm finds and evaluates new solutions. The tuning process is based on a separate set of test instances, each with the same characteristics as the small, medium, and large-sized instances described in Chapter 6. Section 7.2.1 describes the tuning process for each parameter, before Section 7.2.2 provides an overview of the parameters to tune. As the tuning process is extensive, complete results are found in Appendix D.

7.2.1 Methodology for Parameter Tuning

Ropke and Pisinger (2006) and Folkestad and Klev (2021) performed an ad-hoc trial-and-error phase while developing the heuristic to produce a fair set of parameter values. This set was further improved by a more extensive sequential parameter tuning, tuning the parameters one by one. This approach is adapted in this thesis. First, an ad-hoc phase determined reasonable values for certain parameters. A fitting tuning interval for each parameter was also found during this phase. Second, the remaining parameters were tuned in order of importance. The order was determined via the mentioned ad-hoc phase. Each parameter was tested with values from their respective tuning interval, keeping the remaining parameter values fixed. The tuning intervals mainly span between four and eight different values. The performance of the heuristic was evaluated based on the gap to the best-known objective value and total runtime. The best value for the tuned parameter was then used for the remaining of the tuning process. The parameter tuning was conducted on nine different test instance types, namely the small, medium, and large ones, as described in Section 6.1. Due to the stochasticity in both the problem and the heuristic algorithm, the performance was evaluated based on averages over three different instances of each test instance type and further on five runs of the ALNS heuristic on each instance. As we aim at solving real-sized instances of the problem, the performance of the heuristic on the larger test instances was valued higher when determining the parameter values. The parameters to tune, their initial values, their tuning intervals, and their final values are displayed in Table 7.2, listed in descending order of importance. Notice that some of the parameters have tuned values set to a limit point of their tuning interval. Widening the tuning intervals would, however, be unlikely to improve the results, as values outside the tuning intervals have been discarded through the extensive ad-hoc phase.

7.2.2 Parameters to tune

Table 7.2: The parameters used in the ALNS heuristic. The initial values, the tuning intervals and the determined values are displayed. Untuned parameters are displayed only with their determined values.

Parameter	Initial Value	Tuning Interval	Determined Value	Description
$\phi_{customer}$	1.0	$[0.3, 0.7]$	0.7	Relevant request threshold.
$\phi_{cars-to-customer}$	1.0	$[0.3, 0.7]$	0.7	Car-to-customer ratio threshold.
$\phi_{relocation-time}$	1.0	$[0.3, 0.7]$	0.7	Relocation time threshold.
ρ_G	10	$[10, 40]$	20	Weight reward parameter for a new globally best solution.
ρ_L	5	$[10, 20]$	15	Weight reward parameter for a new locally best solution.
ρ_N	1	$[5, 10]$	10	Weight reward parameter for a new non-improving solution.
α	0.1	$[0.4, 0.7]$	0.7	Reward decay parameter.
η	$[0.05, 1.00]$	$[0.05, 0.70]$	$[0.15, 0.70]$	Neighborhood size distribution interval.
q_o	0.05	$[0.05, 0.2]$	0.05	Related Removal weight for origin zone distance.
q_d	0.05	$[0.05, 0.2]$	0.05	Related Removal weight for destination zone distance.
q_f	0.1	$[0.1, 0.4]$	0.1	Related Removal weight for fee difference.
p_{worst}	1	$[1, 6]$	2	Worst Removal determinism parameter.
$p_{related}$	1	$[1, 6]$	2	Related Removal determinism parameter.
p_{greedy}	1	$[1, 6]$	4	Random Greedy Insertion determinism parameter.
$\tau_{strategy}$	*reheat*	$reheat, no-reheat$	*reheat*	Reheat strategy for Simulated Annealing.
τ_{init}			$\frac{f(\gamma_{init}) \cdot 0.1}{\ln 0{,}5}$	Initial temperature for Simulated Annealing.
τ_{final}			$\frac{f(\gamma_{best}) \cdot 0.1}{\ln (0.01)}$	Final temperature for Simulated Annealing.
ζ			$1 - (\frac{\tau_{final}}{\tau_{init}})^{\frac{1}{I^{ALNS}}}$	Cooling rate for Simulated Annealing.
I^{ALNS}			100	Number of iterations within the ALNS.
I^{S}			10	Number of iterations within a segment.
I^{main}			50	Number of iterations in the main framework.
T^{limit}			600 s	Runtime limit for the heuristic.
T^{LSO}			0.5 s	Runtime limit for a local search operator.
T^{LS}			2.0 s	Runtime limit for local search.
κ			50%	Share of pricing solution set to initial values in perturbation process

An overview of the parameters used in the ALNS heuristic is provided in Table 7.2. First, we have parameters controlling the size of the solution search space, namely $\phi_{customer}$, $\phi_{cars-to-customers}$ and $\phi_{relocation-time}$. These parameters indicate whether or not a zone pair or a car-move is relevant to investigate, and thus if they are to be included in the pricing and relocation search spaces or not. $\phi_{customer}$ defines a relevant request threshold to determine whether or not a customer is considered relevant. A threshold closer to 1 indicates that a customer must prefer car-sharing in closer to all scenarios to be considered relevant. A zone is considered relevant if there is at least one relevant customer located in it. $\phi_{cars-to-customer}$ defines a desired ratio threshold of cars to relevant customers in each relevant zone. The lower the threshold, the wider the gap between the number of relevant customers and cars in the relevant zone must be for a car-move to this zone to be regarded as relevant. $\phi_{relocation-time}$ defines the relocation time threshold for a car-move. This threshold controls the maximum relocation time a car-move can have to be considered relevant. The threshold is related to the planning horizon, such that a value of 1 defines the maximum relocation time of a car-move equal to the length of the planning horizon. The tuning intervals for the search space parameters are based on insight from the ad-hoc phase.

Second, we have the parameters related to the adaptiveness of the ALNS heuristic. ρ_G, ρ_L, and ρ_N define the different reward values an operator is assigned every time it leads to finding a globally best, locally best, or non-improving accepted solution, respectively. Their tuning intervals are based on insight from the ad-hoc

phase. Further, the reward decay parameter, α, defines the impact of the recent rewards when updating the operator weights. An α of 1 indicates that recent rewards have no impact, while an α of 0 indicates that recent rewards alone determine the new weights. Its tuning interval is set to capture a sufficiently granular variation in the impact of the weight reward values, partially based on values found successful by Folkestad and Klev (2021) and on insight from the ad-hoc phase.

Third, we have parameters related to the destroy and repair operators. η defines the interval from which the neighborhood size factor is drawn. A value of one indicates that the neighborhood size for pricing searches is all zone pairs, while values between zero and one set the size to a corresponding fraction of all zone pairs. Its tuning interval is based on the values used by Folkestad and Klev (2021). Further, q_o, q_d and q_f are Related Removal weights corresponding to distance to the origin zone, distance to the destination zone, and difference in fee, respectively. Their tuning intervals are based on insight from the ad-hoc phase. Next, p_{worst}, $p_{related}$ and p_{greedy} determine the degree of randomness in the destroy and repair operators. Their tuning intervals are set to capture a relatively low level of randomness at first and a level of randomness, making the operator logic almost redundant at the end.

Finally, we have parameters related to temperature control in the Simulated Annealing. $\tau_{strategy}$ defines the reheat strategy, which is set to be either *reheat* or *no-reheat*. The strategy determines the exploration level of the ALNS heuristic. The *reheat* strategy encourages a high level of exploration when the iteration within a run of ALNS is farther from I^{ALNS}, and the level of exploration is sinking towards the end of the search at iteration I^{ALNS}. The *no-reheat* strategy encourages a high level of exploration when the iteration number of Algorithm 1 is farther from I^{main}, and the level of exploration is sinking as the iteration number approaches I^{main}. The ad-hoc phase indicates that this parameter is important when it comes to the ratio between runtime and objective value gap and is therefore included in the tuning process.

The remaining parameters are either determined by insight from the ad-hoc phase, or based on related research. The initial temperature, the final temperature and the cooling rate, τ_{init}, τ_{final} and ζ, respectively, are set as described in Section 5.6.4. Continuing the logic presented in Ropke and Pisinger (2006) and Folkestad and Klev (2021), we are setting $\delta_{init} = 10\%$ and $p_{init} = 50\%$ in order to get the value of τ_{init}, and $\delta_{final} = 5\%$ and $p_{final} = 1\%$ in order to get the value of τ_{final}. These values are based upon Ropke and Pisinger (2006) and Folkestad and Klev (2021), and performed well in the ad-hoc phase. As the tuning process sets the $\tau_{strategy} = reheat$, we achieve τ_{final} when starting with τ_{init}, by setting ζ based upon these values and the number of iterations in one run of ALNS, i.e., I^{ALNS}. The iteration parameters I^{ALNS}, I^S and I^{main} are set to 100, 10 and 50, respectively. That is, the maximum number of iterations is in total 5000, where a new segment is initiated after each 10th iteration. The heuristic is however bounded by a time limit, and is therefore not guaranteed to complete all 5000 iterations. As previously mentioned, the time limit is set to $T^{limit} = 600$ seconds, as the problem is intended to be solved on a frequent basis. The time limits related to the LSs and their operators, T_{LS} and T_{LSO}, are set to 2 and 0.5 seconds, respectively. A perturbtation share of $\kappa = 50\%$ has proven to be efficient.

7.2.3 Conclusions of the Parameter Tuning

The initial and resulting values from the tuning process can be seen in Table 7.2 as the *Initial Value* and the *Determined Value*. To evaluate the effect of the tuning process, we solved the problem using the heuristic with initial and determined parameter values on a separate set of small, medium, and large test instances. Three versions of each test instance type were generated, and the heuristic was run five times for each instance with a time limit of 600 seconds. The average objective values and runtimes for finding the best-known solutions for the relevant instances are reported in Table 7.3. Complete results are provided in Table E.1. As we see, the ALNS heuristic itself proves to be robust on small instances, finding equally good solutions independently of its parameter values. For medium and large instances, it appears that the ALNS heuristic is more sensitive

to the parameter values, with a tuned heuristic improving the objective values with $2\% - 5\%$. This does, however, seem to come at a cost in runtimes finding the best-known solutions for some of the medium and large instances. For small instances, the tuned ALNS heuristic has a markedly reduced runtime as opposed to the untuned.

Table 7.3: Evaluation of how the parameter tuning process affects the performance of the ALNS heuristic. Three versions of each test instance type are generated. Each test instance is run five times with a time limit of 600 seconds. The columns display average values over the five runs and the three versions of each instance type.

Test Instance Type	**Before Tuning Process**		**After Tuning Process**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1	245.95	98.66	246.42	73.00
6-6-1	373.97	236.40	370.00	123.67
8-8-1	513.42	297.47	515.44	277.59
10-9-2	519.93	362.02	534.61	349.86
15-12-2	715.35	328.19	749.86	299.91
20-15-2	859.31	245.59	885.49	356.73
30-20-2	1332.70	268.71	1352.71	309.02
40-25-2	1673.84	289.12	1698.69	200.69
50-30-2	1800.04	412.53	1876.66	489.73
Average Total	892.72	282.08	914.43	275.58

Green cells indicate best average objective values.

7.3 Performance Evaluation of the ALNS Heuristic

This section describes the evaluation process of different performance aspects of the ALNS heuristic. The performance evaluation mainly concerns the heuristic's different components and settings and how they affect the solution quality and runtime. A separate set of test instances are generated for each analysis, unless otherwise stated.

7.3.1 The Value of Adaptivity

The value of adding adaptiveness to a Large Neighborhood Search (LNS) is investigated in this section. Adaptive Large Neighborhood Search (ALNS) differs from an LNS in choosing the destroy and repair operators. As explained in Sections 5.6.3-5.6.7, the ALNS heuristic is updating the probability weights of choosing an operator based on the operator's performance of finding accepted solutions and the quality of the accepted solutions. An LNS, on the other hand, randomly selects an operator with equal probability. The goal of adding adaptiveness is to select operators based on performance and thereby faster achieve better solutions.

Table 7.4 shows the results of running the algorithm with and without adaptiveness for small, medium, and large instances. Complete results are found in Table E.2. The adaptive selection of destroy and repair operators aids in finding solutions yielding higher objective values for most instances. The exceptions are for the two smallest instance types and the instances of type $40-25-2$. The adaptiveness does, however, increase the average runtime. Here, the columns *Time (s)* display the average runtimes for finding the best-known solutions. The average increase in runtime can be explained through the smarter choice of operators. The ALNS heuristic chooses operators based on performance, so the likeliness of finding accepted solutions in each iteration of the ALNS is higher than when choosing operators at random. Recall that an accepted solution is investigated for further improvement by the two LS components of the heuristic. Frequently finding accepted solutions leads to frequent local searches, which again leads to an increase in runtime.

Table 7.4: Evaluation of the effects of having adaptive selection of destroy and repair operators in the ALNS heuristic as opposed to random selection. Three versions of each instance type are generated. Each version is solved ten times for 600 seconds using both adaptive and random selection of operators. The columns display average values over the ten runs and three versions of each instance type.

Test Instance Type	**LNS (random)**		**ALNS (adaptive)**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1	249.93	118.76	246.30	14.04
6-6-1	372.28	13.04	370.33	24.83
8-8-1	470.04	184.13	514.25	117.01
10-9-2	530.89	296.96	536.32	338.94
15-12-2	745.34	265.69	750.41	374.30
20-15-2	904.51	304.06	918.64	344.87
30-20-2	1385.21	222.76	1395.98	279.51
40-25-2	1747.27	195.22	1736.54	255.70
50-30-2	2048.61	225.10	2095.57	346.42
Average Total	939.34	202.86	951.59	232.85

Green cells indicate best average objective values.

7.3.2 The Value of the Local Search Components

This section investigates the effect of the two integrated LS components in the ALNS heuristic. The LS components are discussed in Section 5.7, and are, briefly explained, trying to improve solutions found by the ALNS destroy and repair operators by evaluating neighboring pricing solutions using the Local Search for Pricing solutions (LSP) and neighboring relocation solutions using the Local Search for Relocation solutions (LSR). A separate set of three versions of each test instance type are generated for this analysis. Each test instance is run ten times for a maximum of ten minutes (600 seconds), both with and without the relevant LS component. Notice that when evaluating the performance without LSP, the LSR is active, and vice versa. Tables 7.5 and 7.6 report the average results for each test instance type with and without the LSP and the LSR components, respectively. Complete results are provided in Tables E.3 and E.4. Notice that the runtimes stated indicate when the ALNS heuristic found the best-known solution and not when the search

terminated. This is to examine how the LS components affect how quickly the ALNS heuristic finds high-quality solutions. Figures 7.1 and 7.2 show plots reporting the change in objective value in terms of gap to the best known solution with the increase in runtime.

Table 7.5: Evaluation of how the Local Search component for Pricing solutions (LSP) affects the performance of the ALNS heuristic. Three versions of each test instance type are generated. Each test instance is run ten times with a time limit of 600 seconds. The columns display average values over the ten runs and three versions of each test instance type.

Test Instance Type	**ALNS without LSP**		**ALNS with LSP**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1	222.21	189.51	230.52	104.84
6-6-1	324.55	309.75	356.12	206.26
8-8-1	502.36	297.29	560.77	266.89
10-9-2	442.63	270.37	514.74	387.43
15-12-2	638.42	353.73	764.91	425.49
20-15-2	791.14	409.73	900.91	398.27
30-20-2	1247.70	371.76	1308.33	461.27
40-25-2	1538.76	379.16	1577.75	500.52
50-30-2	1818.67	492.01	1796.61	494.73
Average Total	836.27	341.48	890.07	360.63

Green cells indicate best average objective values.

Table 7.5 presents the objective values and runtimes for each test instance type, recorded as the average over ten runs, for the ALNS heuristic both with and without LSP. For nearly all instances tested, Table 7.5 shows that the ALNS heuristic with LSP, on average, finds significantly better solutions than without the LSP. The runtimes for finding the best-known solutions differ among the varying test instances. For small sizes, the ALNS heuristic with LSP finds the best-known solution more rapidly than the ALNS heuristic without LSP. However, for instances of medium and large sizes the LSP component seems to slow down the heuristic somewhat. This is illustrated in Figure 7.1, where we can see that the ALNS heuristic without LSP explores more of the solution space more rapidly at the start and uses more time to find better solutions in the end. When including LSP, on the other hand, the heuristic uses more time at each iteration of the ALNS. Consequently, the ALNS heuristic with LSP improves the solutions slowly at the beginning of the search, though performing an overall better search in the long run. The trade-off between the improvement in objective value and increase in runtime could be taken into consideration if a real CSO were to choose whether or not to include the LSP.

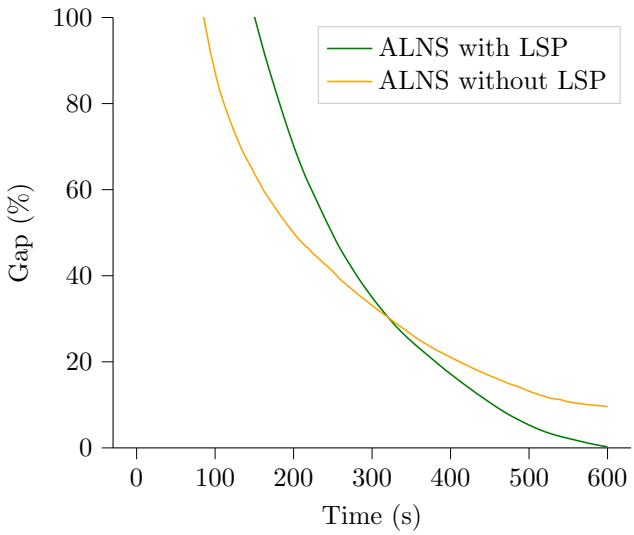


Figure 7.1: Runtime analysis for the ALNS heuristic with and without the Local Search component for Pricing solutions (LSP). The analysis is conducted on average for all test instances, using linear regression based on the times of discovery of improving solutions. The graph displays the average gap in percentage to the overall best solution found at any time of the run for all test instances. Similar runtime analyses for each test instance size is found in Figure E.1.

Table 7.6 presents the average objective values and runtimes for the ALNS heuristic both with and without the LSR component. The results prove that the LSR component helps find higher average objective values for all test instances. The runtime does, however, increase significantly when including the LSR component, as the ALNS without the LSR only investigates pricing solutions. This is further illustrated in Figure 7.2, showing that the ALNS without LSR obtains higher objective values in a short time for all test instances, while the ALNS with LSR finds better solutions when the runtime increases.

The results in Table 7.6 enforce the assumption made by Eilertsen et al. (2021) that pricing solutions are of more importance than relocation solutions when aiming at finding better solutions to the SE-VReP-DP. Recall that a solution constructed by the construction heuristic is provided as input to the ALNS heuristic. This solution may have a relocation solution with certain car-moves to ensure feasibility. The ALNS without the LSR then aims solely at finding improving pricing solutions, meaning the relocation solution provided as input remains the same throughout the search. The results indicate that searches within relocation solutions only improve the objective values with an average of 5% for all instances and an average of 2.9% for large instances. A real CSO should consider the trade-off between this improvement and the increase in runtime.

Table 7.6: Evaluation of how the Local Search component for Relocation solutions (LSR) affects the performance of the ALNS heuristic. Three versions of each test instance type are generated. Each test instance is run ten times with a time limit of 600 seconds. The columns display average values over the ten runs and three versions of each test instance type.

Test Instance Type	**ALNS without LSR**		**ALNS with LSR**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1	210.69	1.74	235.20	137.26
6-6-1	315.39	219.57	381.84	269.80
8-8-1	430.99	43.34	503.65	254.20
10-9-2	473.79	196.35	552.90	382.84
15-12-2	709.51	344.61	754.00	425.13
20-15-2	856.69	389.47	871.44	422.19
30-20-2	1224.36	372.82	1269.48	483.37
40-25-2	1526.05	334.59	1581.37	460.25
50-30-2	1736.42	394.29	1768.89	453.88
Average Total	831.54	255.20	879.86	365.43

Green cells indicate best average objective values.

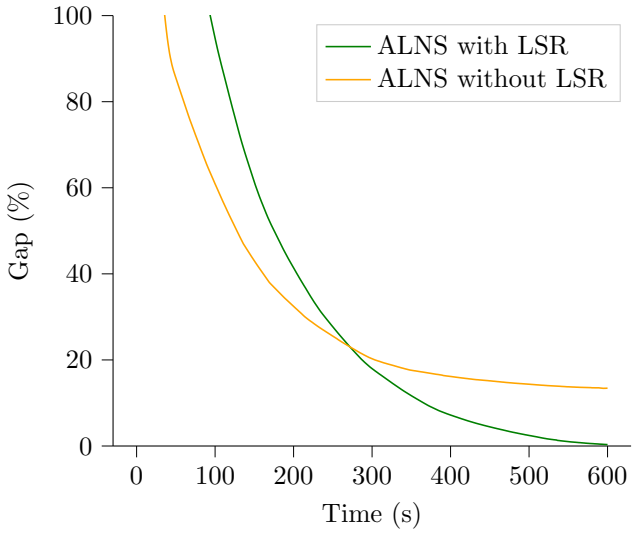


Figure 7.2: Runtime analysis for the ALNS heuristic with and without the Local Search component for Relocation solutions (LSR). The analysis is conducted on average for all test instances, using linear regression based on the times of discovery of improving solutions. The graph displays the average gap in percentage to the overall best solution found at any time of the run for all test instances. Similar runtime analyses for each test instance size is found in Figure E.2.

7.3.3 Time Limit Analysis

This section analyzes how the runtime affects the overall performance of the ALNS heuristic. The purpose of this analysis is to evaluate whether the ALNS heuristic proposed is applicable to solving the problem frequently. Recall that the heuristic's time limit is set to 600 seconds to adapt to a real-life situation where the problem is to be solved every hour. First, we investigate if the construction heuristic alone provides satisfactory solutions, making the remainder of the heuristic excessive. Second, we analyze whether the time limit of 600 seconds is reasonable or if it is too restraining.

In order to assure that the ALNS heuristic improves the initial solution provided by the construction heuristic, a comparison of the results from the construction heuristic and the ALNS heuristic is conducted. The comparison is based on a set of small, medium, and large test instances, where three versions of each instance type are generated. The average objective values and runtimes for finding the best-known solutions over ten runs on each instance are displayed in Table 7.7. Complete results are provided in Table E.5. Recall that the purpose of the construction heuristic is to deliver a feasible initial solution. Therefore, it does not necessarily make reasonable choices when considering the relocation solution or the pricing solution. This is observed in the results, as the average objective values of the construction heuristic are substantially improved by the remaining parts of the ALNS heuristic when performing a search for a maximum of ten minutes. Notice, however, that the runtimes for the construction heuristic are almost negligible for all test instances

Table 7.7: Comparison of the construction heuristic alone and the ALNS heuristic. Three versions of each test instance are generated. Each test instance is run ten times with a time limit of 600 seconds for the ALNS heuristic. Columns Objective Value display the average objective values for the test instance type. Columns Time (s) show the average runtimes for finding the best known solution for the relevant test instance type.

Test Instance Type	**Construction Heuristic**		**ALNS Heuristic**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1	21.75	0.01	251.37	256.40
6-6-1	70.92	0.01	369.65	238.95
8-8-1	127.86	0.01	484.25	289.87
10-9-2	84.95	0.02	526.62	374.28
15-12-2	132.19	0.02	719.34	418.53
20-15-2	162.54	0.03	875.04	389.03
30-20-2	308.34	0.04	1269.29	496.43
40-25-2	784.90	0.05	1671.44	456.20
50-30-2	759.57	0.11	1796.13	453.72
Average Total	272.56	0.03	884.79	374.82

Green cells indicate best average objective valuesl

As the ALNS heuristic is running, a greater part of the solutions in the search space is being discovered and evaluated. When the sizes of the instances increase, the solution spaces increase, leaving the heuristic with a greater task of finding the optimal solution. As discussed in Section 7.2.2, the time limit of the heuristic is set to ten minutes (600 seconds) when solving the test instances. To evaluate whether this time limit severely

restrains the heuristic's performance, we perform a comparison on a similar set of test instances as in the construction heuristic analysis, allowing the ALNS heuristic to run up to three hours. Again, we average over ten runs of the heuristic of each test instance to adjust for the randomness of the heuristic. Each run terminates after three hours or when the stopping criterion for maximum iterations is met. The best-found solutions after the stated time limits are reported. Further, the improvements in objective values compared to the best known solution after a maximum of 600 seconds are calculated. The results from this analysis are summarized in Table 7.8. Complete results are provided in Table E.6.

Table 7.8: Evaluation of how the time limit affects the performance of the ALNS heuristic. Three versions of each test instance type are generated. Each test instance is run ten times for 600 seconds (ten minutes), 3600 seconds (one hour) and 10800 seconds (three hours). The average objective values and runtimes over the ten runs and the three versions of each test instance type are displayed. Improvement is measured from the average best objective values found after a maximum of 600 seconds

Test Instance Type	**ALNS Heuristic (600 s)**		**ALNS Heuristic (3600 s)**			**ALNS Heuristic (10800 s)**		
	Objective Value	Time (s)	Objective Value	Time (s)	Improvement (%)	Objective Value	Time (s)	Improvement (%)
5-4-1	265.82	59.92	265.82	59.92	0.00	265.82	59.92	0.00
6-6-1	365.26	38.29	365.26	38.29	0.00	365.26	38.29	0.00
8-8-1	540.02	600.00	547.99	3008.19	1.47	552.16	3602.20	2.25
10-9-2	552.16	600.00	557.59	3600.00	0.98	562.26	6552.10	1.83
15-12-2	753.36	600.00	778.13	3541.66	3.29	779.67	5520.26	3.49
20-15-2	857.81	600.00	900.28	3600.00	4.95	913.67	6319.84	6.51
30-20-2	1252.92	600.00	1335.48	3600.00	6.59	1360.03	6093.16	8.55
40-25-2	1619.64	600.00	1717.22	3600.00	6.02	1750.41	5724.50	8.07
50-30-2	1853.28	600.00	2004.23	3600.00	8.15	2031.13	4390.07	9.60
Average Total	895.03	477.58	939.34	2738.67	3.49	953.38	4255.59	4.48

Green cells indicate best average objective values.

The results tell us that increasing the time limit on small test instances has nearly no effect, gaining only improvements of 1.47% and 2.25% on the largest of the small instances with time limits of 3600 and 10800 seconds, respectively. Notice that the improvements are measured based on the best-known objective values after a maximum of 600 seconds in both cases. For the medium test instances, the time limit of 600 seconds does not seem to significantly restrain the ALNS heuristic either, with improvements in the range of 0.98% to 6.51%. One could argue that these levels of improvement are somewhat significant. However, these minor improvements become negligible when considering the marked increases in runtime and how these would affect a real-life situation where the problem is to be solved frequently. However, the time limit value seems to be more critical when solving large test instances. The solutions found after ten minutes are improved in terms of objective value with up to 8.15% when increasing the time limit to one hour and up to 9.60% when increasing the time limit to three hours. This indicates that the ALNS heuristic could need some improvement regarding being applicable in real-life car-sharing systems. To evaluate the quality of the solutions found by the ALNS heuristic, they are compared to the solutions and upper bounds obtained by a commercial solver in the next section, Section 7.4.

7.4 Comparison of the ALNS heuristic to a Commercial Solver

The heuristic solution method is implemented due to the lack of performance of a commercial solver in the larger instances, as presented in Eilertsen et al. (2021). For small-sized and some medium-sized instances, a commercial solver can find the optimal solution efficiently. Because of this, we can evaluate the performance of the heuristic by comparing the solutions found by the commercial solver Gurobi and those found by the ALNS heuristic.

This analysis consists of two parts. First, the ALNS heuristic is compared to Gurobi with a time limit of 600 seconds. This is to investigate the ALNS heuristic's potential as opposed to Gurobi's for use in real-life car-sharing systems, solving the problem on a frequent basis with a shorter time limit. Second, the ALNS heuristic is compared to Gurobi with a time limit of 3600 seconds. This is to evaluate the overall performance of the ALNS heuristic in terms of finding solutions of good quality, but outside the scope of intended usage, as it is not supposed to run for more than 600 seconds. Two separate sets of small, medium, and large test instances are used for the two analyses, where three versions of each instance type are generated. Further, the problem is solved ten times for each instance with the ALNS heuristic but only once with Gurobi. This is to handle the randomness in the heuristic, which is not present in the commercial solver.

Table 7.9: Comparison of the ALNS heuristic and the commercial solver Gurobi. Three versions of each test instance type are generated. Each test instance is run ten times with the ALNS heuristic for a maximum of 600 seconds. The results from Gurobi after a maximum of 600 seconds are registered as a benchmark. Average objective values, runtimes and gaps are reported. The gap is based on the distance to the best upper bound found by Gurobi.

Test Instance Type	**Gurobi (600 s)**			**ALNS Heuristic (600 s)**		
	Objective Value	Gap (%)	Time (s)	Objective Value	Gap (%)	Time (s)
5-4-1	252.76	0.00	13.11	252.76	0.00	3.45
6-6-1	411.32	0.00	32.43	411.32	0.00	8.31
8-8-1	570.81	0.00	132.77	570.81	0.00	67.82
10-9-2	579.98	3.28	600.00	590.50	1.40	133.83
15-12-2	819.51	13.06	600.00	817.68	13.30	274.80
20-15-2	833.49	44.56	600.00	968.09	17.62	318.13
30-20-2	1454.11	27.65	600.00	1567.24	17.06	376.45
40-25-2	1264.69	79.10	600.00	1786.20	22.65	426.45
50-30-2	(-)	(-)	600.00	2037.64	(-)	370.77
Average Total	687.42	18.63	419.81	1001.47	11.91	393.71

Green cells indicate best average objective values.

(-) in *Objective Value* indicates no feasible solution found.

(-) in *Gap (%)* indicates no feasible upper bound found.

Average values are found replacing (-) with 0.

First, we examine how the ALNS heuristic performs compared to Gurobi when the time limit is set to 600

seconds, the time limit used in the majority of the analyses conducted. The average results over the three versions of each instance type are presented in Table 7.9. Complete results are provided in Table E.7. The objective values reported for the ALNS heuristic are averages over ten runs of each of the three versions of the instance types. The runtimes represent the time spent finding the best-known solutions. The gaps are measured as the distance in the objective values in percentage to the best upper bounds found by Gurobi. Notice that these bounds are pessimistic, meaning the gaps represent maximum distances to the optimal solutions. Therefore, the gaps to the optimal solutions lie somewhere in the interval $[0.00\%, Gap(\%)]$.

We observe from Table 7.9 that the two solution methods have achieved the same solutions for the small instances, with the ALNS heuristic being substantially more efficient. For the remaining instances, the ALNS heuristic out-competes Gurobi in all but the instances of type $15-12-2$, where there is a minor difference in the gaps of 0.24 percentage points. Further, the runtimes indicate that the ALNS heuristic finds the best-known solutions markedly more efficiently than Gurobi. This supports the applicability of the ALNS heuristic as opposed to Gurobi in a real-life situation where decisions must be made frequently. However, the gaps to the optimal solutions are somewhat large for the larger test instances. We observe that the gaps increase for the solutions obtained when the size of the instances increases for both Gurobi and the ALNS heuristic. The gaps in the solutions found by Gurobi are much higher than those in the solutions found by the ALNS heuristic for the large instances and not even available for the largest instance type. As Gurobi is unable to find an upper bound, for instance, $50-30-2$, the largest instance type with known gaps are the instances of size $40-25-2$. Here the gap is 22.65%, which means that the optimistic upper bound of the objective value is 2190.77 in this planning horizon. The ALNS heuristic solutions are, as already discussed, closer to the upper bound than Gurobi solutions. However, a gap of 22.65% indicates that there is room for improvement.

Table 7.10: Comparison of the ALNS heuristic and the commercial solver Gurobi. Three versions of each test instance type are generated. Each test instance is run ten times with the ALNS heuristic for a maximum of 3600 seconds. The results from Gurobi after a maximum of 3600 seconds are registered as a benchmark. Average objective values, runtimes and gaps are reported. The gap is based on the distance to the best upper bound found by Gurobi.

Test Instance Type	**Gurobi (3600 s)**			**ALNS Heuristic (3600 s)**		
	Objective Value	Time (s)	Gap (%)	Objective Value	Time (s)	Gap (%)
10-9-2	597.65	2844.57	0.00	588.71	402.91	1.53
15-12-2	861.34	3600.96	4.82	821.18	494.93	9.93
20-15-2	1099.88	3600.29	3.24	993.66	584.22	14.28
30-20-2	1746.79	3601.52	6.61	1607.98	1801.81	15.78
40-20-2	2155.20	3600.31	10.02	1993.30	2973.28	18.89
50-30-2	(-)	3600.57	(-)	2222.51	1875.70	(-)
Average Total	1076.81	3334.17	4.11	1371.22	1355.48	10.07

Green cells indicate best average objective values.

(-) in *Objective Value* indicates no feasible solution found.

(-) in *Gap (%)* indicates no feasible upper bound found.

Averages are found replacing (-) with 0.

Second, we investigate how the ALNS heuristic performs compared to Gurobi when the time limit is set to

3600 seconds. To do so, we generate a separate set of medium and large instance types and solve them in the same fashion as described above, this time with a time limit of 3600 seconds. Table 7.10 reports similar values as Table 7.9. Complete results are provided in Table E.8. We do not include the smallest instances, where Gurobi and the ALNS heuristic both solved to optimality within the time limit of 600 seconds, as presented in Table 7.9.

We observe from Table 7.10 that Gurobi outperforms the ALNS heuristic in all instances except for $50-30-2$, where Gurobi is unable to find any feasible solutions within the time limit. As the instances increase in size, the gaps for the solutions found by both Gurobi and the ALNS heuristic increase, as we would expect when the complexity is increased. However, when comparing the gaps of the solutions found by Gurobi and by the ALNS heuristic, we observe that the relative differences in gaps are not increasing but rather remain around nine percentage points for instances of size $20-15-2$ and larger. We also observe that Gurobi continues to find better solutions until the time limit is reached, while the ALNS heuristic usually reports its best solutions relatively early. For the medium-sized instances, i.e., $10-9-2$, $15-12-2$, and $20-15-2$, the best solutions are found within 600 seconds, constituting a sixth of the time limit. The ALNS heuristic reports its best solutions for the remaining large instances after at least half of the time limit has passed. As discussed in Section 7.3.3, the ALNS heuristic constructs a feasible solution in almost negligible time for all test instance types, which indicates that more or less all the time is spent improving a feasible solution. Even if increased runtime noticeably increases the performance of the ALNS heuristic, as shown in Section 7.3.3, we notice that it struggles to efficiently find better solutions when compared to the commercial solver Gurobi. A possible reason for this is that the ALNS heuristic may not be able to explore good solution neighborhoods properly, as the acceptance criterion may lead to a too frequent acceptance of unfavorable solutions. Another possible explanation is that the decomposed integration of the ALNS with the LS operators is not the ideal way of efficiently exploring the problem's solution space.

A positive aspect of the proposed solution method is the efficiency of finding a feasible solution, as Gurobi is unable to find any feasible solutions within the time limit of 3600 seconds for the largest instance type, even when the time limit is unreasonably high for applicability. This is a clear advantage of the ALNS heuristic to Gurobi when solving the SE-VReP-DP for larger instances on a frequent basis.

7.5 Managerial Insight

In this section, a series of analyses are conducted in order to provide insight regarding car-sharing systems to real-life CSOs. The main contribution of this thesis is the inclusion of dynamic pricing in car-sharing systems. Therefore, this section focuses on analyzing how dynamic pricing affects different aspects of the problems presented in the thesis. Recall from Chapter 1 that we refer to a dynamic pricing strategy, although it is not considered dynamic when solved only for one planning horizon. In a real-life setting, the relocation and pricing problem would be solved frequently on a daily basis. The long-term effects of including dynamic pricing cannot be properly analyzed without a simulation model such as a rolling horizon framework. A long-term analysis is presented in 7.6. In this section, however, the problem is evaluated in a short-term perspective and includes analyses concerning how the choice of pricing strategy affects different aspects related to car-sharing for a planning horizon of two hours. This short-term perspective represents solving the problem presented in this thesis once. One set of the small, medium, and large instance types is generated for these analyses. Further, three versions of each instance type are generated, and each version is solved ten times to handle the randomness in the ALNS heuristic. All test instances are as described in Chapter 6, and the ALNS heuristic is run with the same parameter settings as described in Section 7.2.2. The effects of dynamic pricing on profit, employee behavior and customer satisfaction are analyzed in Sections 7.5.1, 7.5.2 and 7.5.3, respectively. Before presenting these analyses, a brief discussion about the differences between the dynamic and static pricing problems is provided below.

The only difference between the dynamic pricing problem and the static pricing problem is the solution space of the pricing solution. The size of the relocation solution space, $Sol_{relocations}$, remains the same for both pricing strategies. However, the size of the solution space of the pricing solution space differs greatly for the two problems. While the solution space size of the pricing solution when using dynamic pricing is $(|\mathcal{I}|^2)^{|\mathcal{L}|}$, it is only 1 when static pricing is used. This is because the fee levels between all zone pairs are set to zero. Let us consider an instance of $|\mathcal{I}|= 50$ and $|\mathcal{L}|= 5$. The total size of the solution space for static pricing is then given by $Sol_{relocations} \cdot 1 = Sol_{relocations}$. For dynamic pricing, the solution space is given by $Sol_{relocations} \cdot (50^2)^5 \approx Sol_{relocations} \cdot 9.76 \cdot 10^{16}$. In the worst case, these numbers represent the number of solutions to be evaluated before finding the optimal solution, when not considering the complexity reduction techniques presented in Section 5.8.

The ALNS heuristic used with both dynamic and static pricing strategies works as explained in Chapter 5.6. However, as the main effort and focus in the ALNS heuristic is to find new and better pricing solutions, a static pricing strategy makes the problem less complicated to solve for the heuristic. The heuristic's destroy and repair operators will not yield any new solutions, as these try to make changes only to the pricing solution. The same logic is applied to the Local Search on Pricing solutions (LSP) component. Thereby, the only part of the heuristic that will search for new solutions is the Local Search on Relocation solutions (LSR). By this, for every iteration of the ALNS, the new solutions for the static pricing problem are only discovered by the local search operators presented in Section 5.7.2.

7.5.1 Effects of Dynamic Pricing on Profit

The main objective of a CSO in this thesis is to maximize the profit, as stated in Section 4.2. To gain insight into the effects of dynamic pricing on the profit, an analysis is conducted comparing the profits with a dynamic pricing strategy and a static pricing strategy. Table 7.11 presents the objective values and runtimes of the ALNS heuristic for finding the best known solutions with both static pricing and dynamic pricing. The values are averages over ten runs and three versions of each instance type. Complete results are provided in Table E.9.

The results presented in Table 7.11 show that the ALNS heuristic yields higher values when using dynamic pricing for all test instance types. The average improvement in profit from static pricing is 20%, which implies that dynamic pricing is profitable for the CSO. Note that there is a descending tendency in the improvement for the larger instances types. The average improvement of dynamic pricing for the smallest instance type is 85%, whereas the average improvement of the largest instance types is only 1%. The dynamic pricing problem is much more complex than the static pricing problem, with the complexity of the dynamic pricing problem increasing polynomially compared to the complexity of the static pricing problem, as discussed in the introduction of Section 7.5. By this, the ALNS heuristic can explore a larger proportion of the solution space of the static pricing problem than it can for the dynamic pricing problem within the same amount of time. From the runtime analysis conducted in Section 7.3.3, and the performance evaluation discussed in Section 7.4, one can assume the ALNS heuristic would find solutions for the dynamic pricing problem yielding even higher profits for the larger instance types having a higher time limit.

For the static pricing problem, the ALNS heuristic does not search for good pricing solutions. As the time limit is the same for solving static and dynamic pricing problems, the ALNS heuristic will have more time to find optimal employee-based relocations for the static pricing problem. The results from Table 7.11 indicate that focusing on finding good pricing solutions yields a higher profit. This confirms the assumption drawn by Eilertsen et al. (2021) that prioritizing the pricing solution has a bigger impact on the CSO's profit.

Table 7.11: The results of running the ALNS heuristic on equal instance with both static and dynamic pricing strategies for different sizes of the instances. Each instance is run ten times with a time limit of 600 seconds. The average results of three generated instances of each instance size compose the profit, i.e., Objective Value, and runtime of finding the best known solution, i.e., Time (s).

Test Instance Type	**Static Pricing**		**Dynamic Pricing**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1	139.30	5.84	257.27	165.24
6-6-1	237.00	12.78	362.61	219.38
8-8-1	389.89	48.13	528.63	211.37
10-9-2	387.43	136.25	570.82	367.80
15-12-2	590.52	423.02	777.78	339.47
20-15-2	646.05	609.33	907.07	431.68
30-20-2	1080.42	609.87	1263.76	390.12
40-25-2	1430.53	613.20	1535.89	491.38
50-30-2	1849.55	635.96	1864.86	514.86
Average Total	750.08	343.82	896.52	347.92

Green cells indicate best average values.

7.5.2 Effects of Dynamic Pricing on Employee-Based Relocations

Another interesting aspect is to see whether the introduction of dynamic pricing affects the employees' work. Results concerning the number of tasks performed by the employees, the number of employees used to perform these tasks, and the overall direct costs associated with the car-relocations are given in Table 7.12. These results are based on the same solutions discussed in Section 7.5.1, and presented as averages over the same ten runs and three versions of each instance type. Complete results are provided in Table E.10.

The results presented in 7.12 reveal that introducing a dynamic pricing strategy incentivizes employee-based relocations. For instances larger than $10 - 9 - 2$, there are on average two active employees for the dynamic pricing problem, while on average only 1.2 employees are active for the static pricing problem. Here, we define an *active employee* as an employee who performs at least one relocation during the planning horizon. Further, the average number of relocations performed nearly triples when introducing dynamic pricing, going from 2.49 to 6.17 for instances larger than $10 - 9 - 2$. This can be explained through the following. The dynamic pricing strategy allows the exploitation of potential customers' willingness to pay. In Section 7.5.3 we will observe that the average profit per request satisfied is significantly higher with dynamic pricing than with static pricing, indicating the static pricing strategy undersells the service in certain cases. As a consequence, a higher number of zones become relevant destinations for employee-based relocations with dynamic prices. The employees will therefore perform more relocations, exploiting demand in the most profitable way possible. There are costs associated with these relocations, such as maintenance and electrical costs. However, these costs do not include employee salaries and are noticeably lower compared to the increased revenue resulting from performing the relocations.

The number of active employees increases when introducing a dynamic pricing strategy for one planning

Table 7.12: The relocations performed, active employees and the cost of performing the relocations using static and dynamic pricing strategies. The values are averages based on three instances of the same instance type, where there is a time limit of 600 seconds.

Test Instance Type	Static Pricing			Dynamic Pricing		
	Relocations Performed	Active Employees	Relocation Costs	Relocations Performed	Active Employees	Relocation Costs
5-4-1	1.00	1.00	4.13	2.10	0.97	6.66
6-6-1	1.00	1.00	4.71	2.77	1.00	10.92
8-8-1	1.00	1.00	5.60	1.83	1.00	7.55
10-9-2	1.00	1.00	4.02	2.33	1.57	10.39
15-12-2	1.00	1.00	5.15	4.63	2.00	17.76
20-15-2	1.93	1.13	8.40	7.27	2.00	29.34
30-20-2	2.87	1.27	12.78	7.13	2.00	32.83
40-25-2	2.63	1.23	13.18	5.70	2.00	27.14
50-30-2	4.00	1.37	19.36	6.13	2.00	30.15
Average Total	1.83	1.11	8.59	4.43	1.61	19.19

horizon. This does not affect the objective values, as the employee salaries and other related costs to having an employee are considered fixed. For a CSO to properly make decisions concerning the number of employees and the potential benefits of dynamic pricing, these fixed costs, in addition to the expected number of active employees, should be considered. Further, the model formulation lacks a perspective on potential demand in a future exceeding the planning horizon. Incorporating this through, for instance, a third stage in the model would allow better insight into whether dynamic pricing could be used to reduce the number of employees, setting prices to incentivize customer moves to zones with expected future demand. This hypothesis is partly evaluated through a rolling horizon analysis in Section 7.6.2, where we investigate whether a dynamic pricing strategy can incentivize customers to perform necessary charging of cars.

7.5.3 Effects of Dynamic Pricing on Satisfied Customer Requests

Another aspect of a dynamic pricing strategy to consider is how it affects the capturing of demand. Table 7.13 presents the numbers of satisfied requests and profits per request for both a static and a dynamic pricing strategy. These results are based on the same solutions discussed in Sections 7.5.1 and 7.5.2, and presented as averages over the same ten runs and three versions of each instance type. Complete results are provided in Table E.11.

Table 7.13 shows that there is little difference in the average numbers of satisfied requests when comparing static and dynamic pricing strategies. One reason for this is the limited number of cars available. We observe that the number of satisfied requests is close to the number of cars offered by the CSO in the different test instances. If some of the cars in need of charging are being moved to charging stations by employees, an even smaller number of cars are available to satisfy customer requests.

However, the profit per request with dynamic pricing is substantially higher than with static pricing. This means that a dynamic pricing strategy not necessarily captures more demand, i.e., convinces more customers to use car-sharing, but better exploits the customers' willingness to pay. As an example, consider a satisfied customer request in the static pricing problem. This customer has accepted using the car-sharing service without any associated fee, i.e., a fee level of zero, when in reality the customer would be willing to pay a fee of 40 NOK. A dynamic pricing strategy would capture this willingness to pay, which is reflected in the results as the increase in average profit per request. This deduction coincides well with the analysis conducted in

Table 7.13: The number of customers served (Satisfied Requests) and the average profit they generate (Profit per Request) using static and dynamic pricing strategies. The results are average values from running the ALNS heuristic ten times on three versions of each test instance type, with a time limit of 600 seconds per run.

Test Instance Type	**Static Pricing**		**Dynamic Pricing**	
	Satisfied Requests	Profit per Request	Satisfied Requests	Profit per Request
5-4-1	2.53	57.04	3.36	79.23
6-6-1	4.45	54.18	4.75	79.11
8-8-1	6.16	64.13	6.77	79.25
10-9-2	7.40	52.98	7.71	75.41
15-12-2	10.37	57.41	10.45	76.20
20-15-2	11.54	56.40	12.18	76.92
30-20-2	16.26	66.93	17.11	75.96
40-25-2	20.59	69.99	20.48	76.51
50-30-2	25.01	74.46	24.51	77.56
Average Total	11.59	61.50	11.93	77.35

Section 7.5.2, where we concluded the employees perform more relocations to capture the higher willingness to pay.

Keeping customer satisfaction at a high level is beneficial for the CSO in the long run. This increases the likelihood of both retaining existing customers and attracting new potential customers. Fees making car-sharing more expensive could decrease customer satisfaction, as the price is one of the major elements of a customer's utility. One might argue that the effect of an increase in price will have a negative impact on future demand. However, the introduction of an incentive, i.e., a negative fee, can potentially capture otherwise unapproachable demand. An analysis investigating the customers' perception of the car-sharing system for different prices could be of additional help to the CSO when considering the use of dynamic prices. How dynamic pricing affects the CSO's customer base is further analyzed in Section 7.6.2.

7.6 Analyzing the Long-Term Effects of Dynamic Pricing Through Simulation

Examining how dynamic pricing affects different aspects of the car-sharing service for one planning horizon provides a good indication of the pricing strategy's potential. However, we need a more thorough simulation approach to achieve deeper insight into how a dynamic pricing strategy would affect a typical day for the CSO. Therefore, the problem presented in this thesis has been adapted to fit a rolling horizon framework, simulating a full working day. We will refer to the full day as long-term for simplicity and to clearly distinguish it from the short-term analysis made in 7.5. Section 7.6.1 briefly describes the functionality of the framework and the assumptions made, before Section 7.6.2 provides similar analyses as presented in Section 7.5 concerning the effects of a dynamic pricing strategy, however with a wider perspective.

7.6.1 Rolling Horizon Simulation Framework

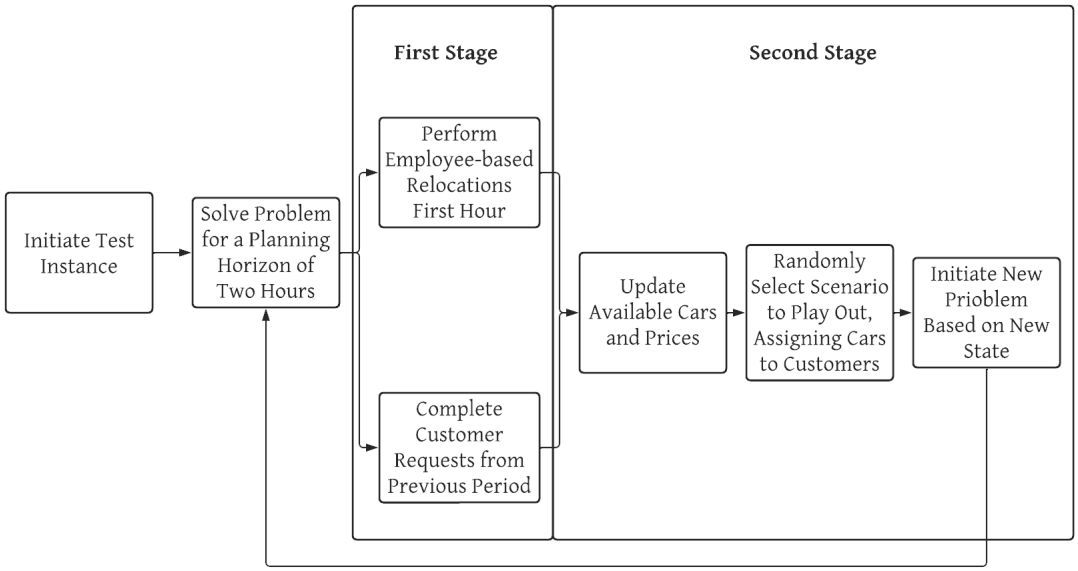


Figure 7.3: Overview of the rolling horizon simulation framework.

The rolling horizon simulation framework attempts to replicate a real-life situation, where the CSO makes frequent decisions concerning prices and employee-based relocations. Figure 7.3 provides an overview of the framework. First, a test instance is initiated as previously done in all precedent analyses conducted in this chapter. This provides the starting point for the simulation. The problem is solved with the ALNS heuristic, as usual with a planning horizon of two hours. Based on the solution provided, the two stages of the problems are simulated. First, all employee-based relocations are performed during the first hour of the planning horizon. Notice that cars put to charging by employees hereby are made unavailable. After the relocations are performed, we enter the second stage of the problem. A random scenario among all 25 possible scenarios is selected to be rolled out, distributing available cars to the requesting customers. Thus far, we have simulated two hours, after which we update the state of the world. The details of this process are described below. When the state is updated, a new problem is initiated based on the current state of the world.

From the initial starting point, we now find ourselves *one hour* ahead in time. This means we recently completed the *first stage* from the previous problem. When applying the solution found for the new problem, our new first stage enrolls simultaneously as the second stage of the previous problem. As illustrated in Figure 7.3 we therefore must complete customer requests from the second stage of the previous problem. The cars serving these customer requests are therefore unavailable for relocations in the first stage of the current problem, but may be available to serve new customers in the second stage of the current problem. Figure 7.4 attempts to illustrate how this overlap of problems finds place.

When updating the state after each second-stage period, we update values concerning the employees, the cars and future potential customers. Employees are reinitiated with origin zones corresponding to where their final relocations ended in the first stage of the previous problem. The cars' origin zones are updated in a similar manner, based on relevant relocations or customer requests moving the cars to different zones. Further, the cars' registered driving time is updated to keep track of their battery levels. If a car's registered

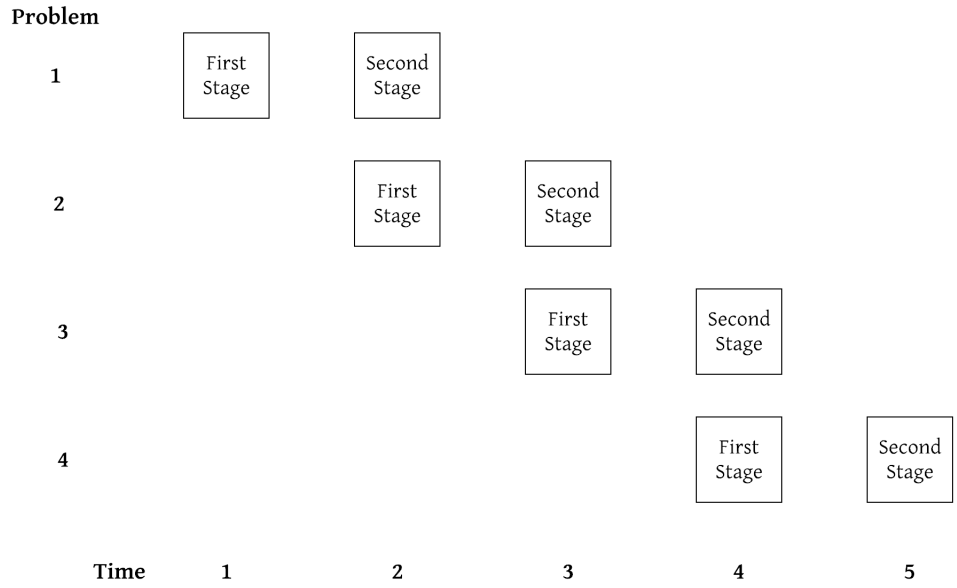


Figure 7.4: Overview of the time course in the rolling horizon framework. Here, the problem is solved every hour for five consecutive hours, yielding four different problems with overlapping first and second stages.

driving time since last charging reaches a certain threshold, the car is labeled as in need of charging for the next planning horizon. Similarly, if a car has been charging for a certain amount of time, it is labeled as fully charged and made available for the next planning horizon. Finally, a new set of customers are generated for the coming planning period. These customers are independent of the customers generated for the previous problem. All sets of customers used are equal for both the static and the dynamic problem.

Certain new parameters are necessary to enable the simulation of a day in the rolling horizon framework. These are the driving time threshold before a car needs to be charged and the time needed to sufficiently charge a car. Vybil operates with EVs of type Renault Zoe, with a range of 386 minutes driving time if maintaining an average speed of 40 kilometers per hour and having a battery charged up to 80%[2]. These numbers yield a driving time threshold of 386 minutes. 386 minutes is therefore used as the threshold in the simulation. This means that after having driven 386 minutes, a car has a battery level of 20% and is consequently labeled as in need of charging. When initiating the test instances at the beginning of the simulation, 15% of the cars are labeled as in need of charging. The remaining cars are assigned a registered driving time since last charging. This value is drawn from the interval $[0, 385]$. Notice that the interval does not include 386, meaning none of these cars are labeled as in need of charging. Using a rapid charger, a Renault Zoe needs one hour and five minutes to increase the battery level by 80%. Making a reasonable approximation to better match our planning horizon of two hours, we set the time to sufficiently charge a car to one hour. Notice that we make the simplification that all cars put to charging has a battery level of 20%.

The rolling horizon framework simulates ten hours, representing a work day from 7 a.m. to 5 p.m. Sets of three initial test instances of each instance type form the basis for the framework. The problem is solved by running the rolling horizon simulation framework for the ALNS heuristic ten times for each test instance using both static and dynamic pricing strategies. When the initial problem is solved for both strategies, the same scenario is selected to be played out for both solutions and a new set of customers is generated. This process is repeated nine additional times, meaning the last problem is solved at 4 p.m., all first stages

[2]https://www.renault.co.uk/electric-vehicles/zoe/battery.html, accessed 2022-07-06

completed within 5 p.m. and the final second stage lasts until 6 p.m. All sets of generated customers are equal for the two pricing strategies, and equal for the ten different runs of the ALNS heuristic. Results presented in Section 7.6.2 are provided as aggregates of results from each hour, representing total values after a complete working day.

7.6.2 Analyses of Long-Term Effects

The heuristic solution method has been tested in the rolling horizon framework described above to analyze the long-term effects of incorporating a dynamic pricing strategy. Some of the analyses presented in this section are similar to those conducted in Section 7.5, where the effects of dynamic pricing on profit, employee-based relocations, and customer requests were discussed. The findings concerning these elements are only briefly discussed in this section, as they simply provide confirmation of the previous findings in a larger scale. This section focuses on the long-term effects and will highlight aspects that were not evident from the short-term analyses. This includes how dynamic pricing affects the need for employees to perform charging-moves, i.e., relocations of cars in need of charging to charging stations. Additionally, reflections concerning how dynamic pricing affects the CSO's customer base are provided. Notice that all values presented throughout this section are accumulated averages over ten runs of the ALNS heuristic on three versions of each test instance type. Unless otherwise stated, the same set of test instances is used for the following analyses. Tables 7.14, 7.15 and 7.16 are similar to Tables 7.11, 7.12 and 7.13 presented in Section 7.5 in terms of content.

Table 7.14: Evaluation of how dynamic pricing affects the overall profit during a work day. The accumulated objective values over the ten hours in the work day are displayed in the columns Objective Value. The average runtimes the ALNS heuristic spent finding the best known solution for each hour are displayed in the columns Time (s).

Test Instance Type	**Static Pricing**		**Dynamic Pricing**	
	Objective Value	Runtime	Objective Value	Runtime
5-4-1	1731.09	3.17	2246.12	40.12
6-6-1	2842.57	6.85	3514.55	101.71
8-8-1	3838.62	18.05	4757.58	261.65
10-9-2	3674.10	52.82	5071.25	297.26
15-12-2	5190.65	101.00	6817.08	520.91
20-15-2	6951.42	185.79	8686.37	521.26
30-20-2	10101.74	276.44	11069.95	441.68
40-25-2	13746.34	429.30	15238.85	456.56
50-30-2	17335.61	442.62	18860.54	480.31
Average Total	7268.02	168.45	8473.59	346.83

Green values indicate best average objective values.

First, the accumulated profit for both the static and dynamic pricing strategies resulting from using the heuristic solution method for a full work-day is presented in Table 7.14. As in Section 7.5.1, we observe a significant improvement in objective values for all test instances with a dynamic pricing strategy compared to a static pricing strategy. It is reasonable to assume that for an even longer time horizon, this tendency will

become more evident and the relative increases in objective values will be more significant. Second, the results presented in Table 7.15 reveal that the average number of active employees and employee-based relocations are higher than with static pricing, and is likely due to the desire to exploit the customers' willingness to pay. Again, these results confirm the conclusions drawn in Section 7.5.2. Finally, the results displayed in Table 7.16 show the numbers of satisfied customer requests do not significantly differ for the two pricing strategies, possibly due to the number of requests being limited by the number of cars available in both cases. The average profits per request are, however, significantly higher with dynamic pricing. The reasoning behind this is provided in Section 7.5.3.

Table 7.15: Evaluation of how dynamic pricing affects the employee-based relocations per hour during a work day. The average number of active employees, number of relocations performed and relocation costs are displayed for both static and dynamic pricing.

Test Instance Type	**Static Pricing**			**Dynamic Pricing**		
	Active Employees	Relocations Performed	Relocation Costs	Active Employees	Relocations Performed	Relocation Costs
5-4-1	0.10	1.00	4.71	0.49	5.67	25.87
6-6-1	0.10	1.00	4.79	0.52	8.56	33.91
8-8-1	0.19	1.89	8.39	0.69	12.22	52.79
10-9-2	0.20	2.00	6.00	0.84	13.56	51.47
15-12-2	0.21	2.11	10.71	1.13	19.89	83.19
20-15-2	0.23	2.33	9.14	1.22	23.33	94.73
30-20-2	0.33	3.33	16.98	0.89	17.78	86.99
40-25-2	0.34	4.78	29.29	0.94	20.22	102.22
50-30-2	0.72	13.11	68.40	1.12	24.11	125.18
Average Total	0.27	3.51	17.60	0.87	16.15	-72.93

We observe from these results that solving the problem in a rolling horizon framework simply confirms the conclusions drawn in Section 7.5. The framework does, however, provide potentially useful insight in terms of employee and customer behavior. An interesting aspect to consider is how the inclusion of dynamic pricing affects the need for employees to perform necessary relocations of cars in need of charging to charging stations. We refer to these relocations as charging-moves henceforth. Figure 7.5 shows the average number of charging-moves performed by employees and customers for a full work day using both static and dynamic pricing strategies.

Table 7.16: Evaluation of how dynamic pricing affects the satisfied customers. The accumulated number of satisfied requests and average profit per request are displayed. The results are average values from running the simulation framework ten times on three versions of each test instance type.

Test Instance Type	**Static Pricing**		**Dynamic Pricing**	
	Satisfied Requests	Profit per Request	Satisfied Requests	Profit per Request
5-4-1	32.89	53.37	34.00	66.01
6-6-1	49.89	57.42	52.00	68.04
8-8-1	66.00	58.57	65.89	72.88
10-9-2	72.67	51.35	76.56	67.46
15-12-2	102.89	51.67	99.56	70.41
20-15-2	125.56	56.21	122.89	72.18
30-20-2	173.56	59.31	166.11	69.42
40-25-2	212.11	66.03	207.56	76.14
50-30-2	250.33	70.70	239.00	81.87
Average Total	120.65	58.29	118.17	71.60

The results presented in Figure 7.5 reveal that fewer charging-moves are performed by employees when a dynamic pricing strategy is applied. Nevertheless, Table 7.15 tells us that the average number of total employee-based relocations performed is higher with a dynamic pricing strategy. This indicates that introducing a dynamic pricing strategy to a higher degree incentivizes customers to perform important charging-moves otherwise performed by employees. This allows for two different reflections, depending on how the CSO regards its employees. First, we assume the CSO uses its employees mainly to assure the fleet of cars is sufficiently charged. The use of dynamic pricing is then likely to reduce the number of necessary employee-based relocations. This gives the CSO an opportunity of reducing the number of employees. Second, we assume the CSO additionally uses its employees to balance its fleet of cars to better meet customer demand. The use of dynamic pricing then allows the employees to concentrate on balancing the fleet, performing profitable relocations in terms of exploiting customers' willingness to pay. In either of the two, the advantage of having more employees to perform profitable relocations should be compared to the cost associated with employee wages and other related costs.

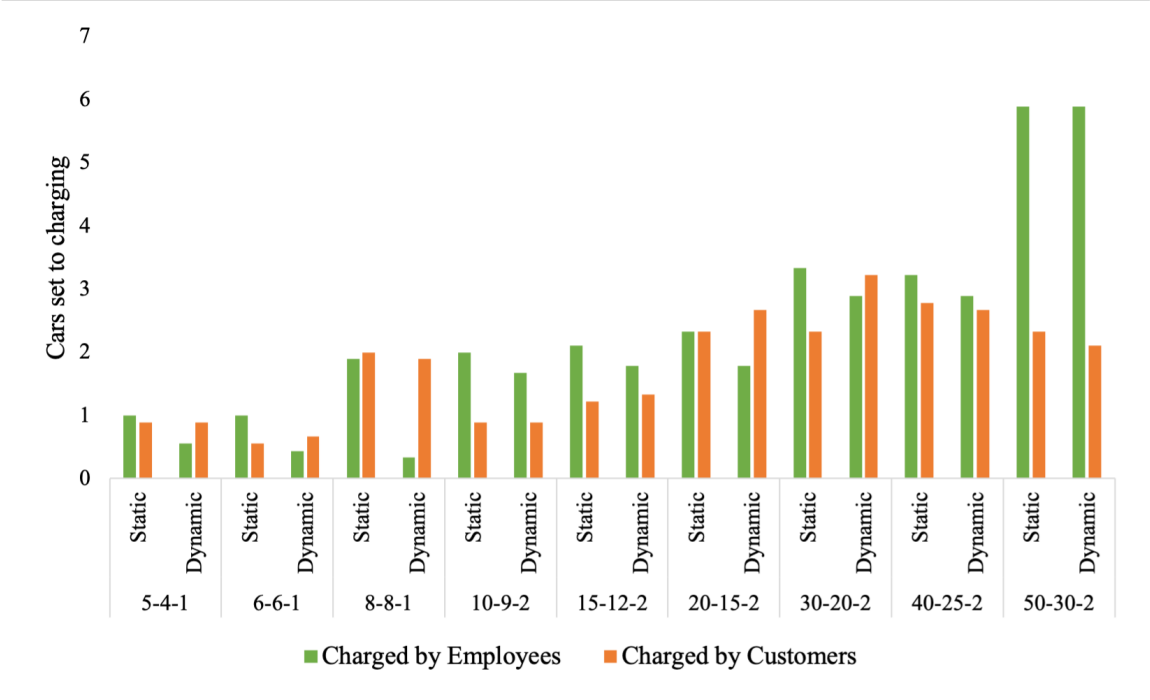


Figure 7.5: Illustration of how the charging of cars is affected by the chosen pricing strategy. The average numbers of cars set to charging by employees and customers during a day are measured by the green and orange columns, respectively. The results are average values from running the simulation framework ten times on three versions of each test instance type.

Another interesting aspect to investigate is how the use of dynamic pricing affects the CSO's customer base. To evaluate this, we define the customer base as follows. The main element of a CSO's customer base is the *satisfied customers*. These customers are the ones we previously have referred to as satisfied requests, being customers preferring car-sharing over other means of transport and being allocated a car. The next element of the customer base is the *willing customers*. These are customers preferring car-sharing over other means of transport, however not being able to rent a car due to a shortage of cars. The final element is the *potential customers*. These are customers who, in the case of dynamic pricing only, would prefer car-sharing if the fee was set sufficiently low. For a static pricing strategy, the willing and the potential customers are equal as there is no fee, i.e., only a fee of zero. Figure 7.6 presents how the CSO's customer base is affected by the two different pricing strategies. As previously introduced in Table 7.16, we know the numbers of satisfied customers are approximately the same with the two pricing strategies. On the other hand, the customer base is significantly larger when using a dynamic pricing strategy. This is mainly due to the marked increase in potential customers, should the CSO decide to lower the chosen fees. However, the results in Table 7.16 indicate that the CSO chooses higher fees than the minimum fee in most cases, as the average profits per request are markedly higher when using dynamic pricing as opposed to static pricing. Therefore we observe from Figure 7.6 that the numbers of willing customers are higher with static prices.

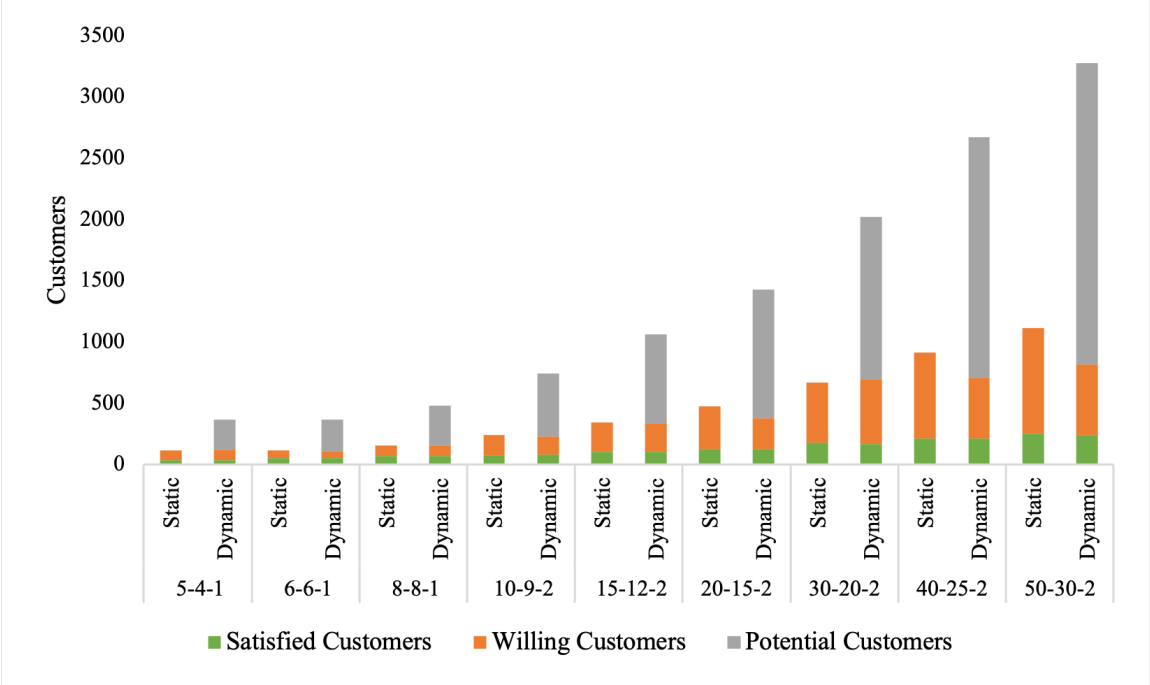


Figure 7.6: Illustration of how the selection of pricing strategy affects the customer base. Satisfied customers are measured by the green columns, while the willing customers using the chosen prices from the rolling horizon simulation are measured by the orange columns. The potential customers are measured by the gray columns. The results are average values from running the simulation framework ten times on three versions of each test instance type.

Based on the above results, the CSO should consider the following when evaluating how dynamic pricing affects its customer base. A dynamic pricing strategy may increase the number of potential customers. This is, however, only if the CSO lowers the fees to a more acceptable level. This would, on the other hand, negatively affect the average profits per request. Further, it comes without a guarantee of being able to satisfy the new willing customers, as there may be a shortage of cars. The insight provided from this analysis would thus be more awarding for the CSO if it simultaneously considers its customer base and the size of its car fleet.

Chapter 8

Concluding Remarks

This thesis considers solution methods for the Stochastic Electric Vehicle Relocation Problem with Dynamic Pricing (SE-VReP-DP). The problem concerns having employees rebalance the fleet of cars in addition to setting prices between all pairs of zones to better meet uncertain demand. The goal of the CSO is to maximize profit, which is done by keeping the cars charged and well-distributed to meet customer demand and having prices that utilize the customers' willingness to pay. This chapter provides a conclusion of the proposed solution methods and the findings from testing these for the SE-VReP-DP. The work presents novel research, as it differs from similar studies by the inclusion of dynamic pricing and the way of handling stochasticity.

A mathematical model is proposed for the SE-VReP-DP. This is a single objective two-stage stochastic mixed-integer programming (MIP) model. The input of the model is the initial vehicle, customer, and employee distributions in addition to battery levels of the EVs and possible fees for each zone pair. The output is a set of relocation tasks for each employee and fees for each zone pair. The relocation solution is inspired by Folkestad and Klev (2021), where each car-move is associated with an employee, task number, car, origin and destination zone, and whether it is a charging-move or not. A charging-move means relocating a car in need of charging to a charging station. The first stage of the model makes decisions regarding the employee-based relocations and sets the prices for each pair of zones. The second stage of the model uses the first-stage decisions to serve the possible requests of the customers in each scenario. The first-stage decisions are made by taking the information from all second-stage scenarios into account.

The objective of the mathematical model is to maximize approximated profit. This includes the costs associated with each car-move and the expected revenue yielded by satisfying customer requests over the set of scenarios. By using scenarios in the model formulation, the model better deals with the uncertainty associated with customer behaviour. The customers are exposed to different modes of transport to fulfill their desired origin-destination trip. To determine which option to choose, the customer selects the available option that yields the highest utility for her. A transportation mode yields a utility for a customer, and this can be divided into a deterministic part and a stochastic part. The deterministic part is considered measurable to the CSO, while the stochastic part is composed of the error in the approximation made by the deterministic part. The stochastic part is approximated using a Logit choice model, where the estimated values are averaged out using scenarios. In each scenario, a realization, i.e., an estimated value, of the stochastic part is drawn from the associated Gumbel distribution. The total revenue is calculated as an average over the revenue generated by satisfied customer requests from each scenario. The model takes the approximation of the generated revenue into account when finding the employee-based relocations to be performed and setting the prices for

each zone pair.

Results from Eilertsen et al. (2021) show that the model is too complex to be solved for real-sized instances by a commercial solver within a reasonable time. To solve real-sized instances of the SE-VReP-DP, an ALNS heuristic is proposed. The ALNS heuristic proposed is a decomposition-based heuristic with integrated Local Search (LS) components, where searches for employee-based relocations and fees for zone pairs are performed separately. The ALNS heuristic uses destroy and repair operators to explore pricing solutions and mitigate the issue of getting stuck in local optima. The heuristic combines this with local searches for both the relocation solutions and pricing solutions to better explore promising neighborhoods.

Several test instances were generated to evaluate the performance of the ALNS heuristic. The test instances were generated inspired by Folkestad and Klev (2021) and Eilertsen et al. (2021), and are based on data provided by the Norwegian Oslo-based car-sharing company Vybil and data from the Google Maps API. The test instances vary in the number of zones, cars, and employees. The results from the computational study show that the ALNS heuristic produces solutions of higher quality than the commercial solver within a reasonable amount of time for real-sized instances. A time limit of 600 seconds is considered reasonable for real-life application, as the problem is meant to be solved frequently, e.g., on an hourly basis. When the size of the instance types increases, the optimality gaps of the solutions found also increase. Moreover, the commercial solver is not able to find any feasible solutions for the largest instances. The ALNS heuristic, on the other hand, is always able to find feasible solutions that are equally good or better than those found by the commercial solver within a time limit of 600 seconds. The performance of the ALNS heuristic is further evaluated independently of the commercial solver. First, results show that including both the LS for relocation solutions and the LS for pricing solutions yields on average higher objective values for almost every instance type. Second, the effect of increasing the runtime shows that for the largest instances, there is a marked improvement in the objective values. This reveals that the larger problems are more complex, and that the ALNS heuristic requires more time to find good solutions. Improvements should be made to the heuristic in order for it to find these solutions in less time.

The ALNS heuristic was further used on several test instances to provide managerial insight to a CSO. Results show that including dynamic pricing yields on average higher profits for all instance types tested. Though the numbers of satisfied requests were similar with both pricing strategies, the average profit made per satisfied request was significantly higher with a dynamic pricing strategy. The long-term effects of introducing dynamic pricing were illustrated using the ALNS heuristic in a rolling horizon framework. These results show that the inclusion of dynamic pricing makes customers perform more charging-moves and that the profit is substantially increased. The size of the potential customer base using dynamic pricing was further found to be markedly higher than when using static pricing. These findings provide valuable information to a CSO when considering the size of both the employee staff and the car fleet.

The real value of the proposed solution methods lies in their ability to find good solutions within a reasonable time, while taking the uncertain demand into account. The proposed MIP model and ALNS heuristic both add value to a real-life CSO by efficiently dealing with uncertainty and revealing the benefit of taking a dynamic pricing strategy. We consider the proposed formulations and solution methods a contribution to the management of car-sharing systems.

Chapter 9

Future Research

There are possible changes that can be made to both the mathematically formulated model of the problem and the ALNS heuristic proposed in this thesis. This could improve efficiency or include new aspects that could be relevant for a CSO and therefore provide a deeper level of insight.

As mentioned in Eilertsen et al. (2021), there are simplifications that could be done to the model to reduce runtime when finding exact solutions. First, cars in need of charging could be made unavailable to the customers. This would reduce the second-stage problem to assigning cars to customer requests. Secondly, the problem could be considered decomposed. Hellem et al. (2021) discuss the possibility of determining employee routes in separate sub-problems in their preliminary research study. Pantuso (2020) propose solving the dynamic pricing problem using an L-shaped method. Both of these propositions would likely lead to a reduced runtime and complexity. Note, however, that these changes may also make the model less realistic.

There are also changes that could be made to make the model more realistic and capture cases that could yield insight and additional profits for the CSO. When employees are relocating cars and need to get to another location, they use folding bicycles in the proposed model. Instead, two or more employees could share a car being relocated in order to relocate the employees simultaneously. Sharing cars this way could save costs and possibly time, and thus the need of employees. Further, the model in this thesis only includes maintenance of cars as a small addition to cost and travel times. To make the model more realistic, maintenance could be modeled similarly to the way cars in need of charging are handled. This means that the car would have to be moved to a location to perform maintenance, making it unavailable to a customer.

Other pricing policies could also be considered. Only having either a pick-up fee or a drop-off fee could make the situation more flexible for customers, in case they are not sure where they would drop the car off, and could also reduce the complexity of the model. Additional flexibility could also be added to the model, as customers can only choose cars with origins in the same zone as the customers' origin. Allowing customers to walk to another zone to pick up a car could make the model more realistic. By working more closely with Vybil and retrieving customer data, the customer behavior could also be more accurately simulated and further increase the accuracy of the results.

Another possible adjustment is to change the cost of performing relocations. In this thesis, the cost of having employees relocate the cars is ignored - only the costs related to the cars themselves are included. Including a cost occurring if an employee performs at least one car-move could improve the applicability of the results

yielded by this thesis. Further, the costs related to a car-move as they are implemented now are negligibly low compared to the potential revenue gained by performing the car-move. Consequently, the choice of relocating a car is nearly always taken if it aids in exploiting the customers' willingness to pay. Increasing the cost of performing relocations could help mitigate this behavior.

The ALNS heuristic performs better when increasing the runtime. Changes could be made to make the heuristic find better solutions quicker. In order to improve the performance of the ALNS heuristic, different configurations and implementations should be considered. Currently, the ALNS heuristic is decomposed into two separate searches, namely the pricing solution search and the relocation solution search. These two operations can each be modified and tested to find better-performing versions. An approach that is not decomposition-based could also be taken, where pricing and relocation solution searches are combined, yielding a completely different search space for the different integrated algorithms. By incorporating machine learning techniques in the ALNS heuristic, specific characteristics of the instances could be considered when choosing destroy and repair operators. The adaptiveness in the ALNS heuristic has a positive impact on the quality of the solutions found. However, it is also associated with a longer runtime before the best known solution is found. Using reinforcement learning to better choose the pair of destroy and repair operators could improve the performance.

A third stage could be added to the model to simulate the future better and prevent the model from making myopic decisions. The third stage would then function as a simulation of what the future may look like, and the long-term effects would be better illustrated and taken into consideration when the first-stage decisions are made. Another possibility is to simulate a real-life situation in the rolling horizon simulation framework for a longer period, e.g., a week or a month, to better analyze the performance of the ALNS heuristic and provide possibly more accurate insight to a CSO.

Bibliography

G. Alfian, J. Rhee, and B. Yoon. A simulation tool for prioritizing product-service system (pss) models in a carsharing service. *Computers and Industrial Engineering*, 70(1):59–73, 2014.

D. J. . F. Z. Amine Ait-Ouahmed. Relocation optimization of electric cars in one-way car-sharing systems: modeling, exact solving and heuristics algorithms. *International Journal of Geographical Information Science*, 32:2:367–398, 2018.

A. Avenalia, Y. M. Chianeseb, G. Ciucciarellib, G. Grania, and L. Palagi. Profit optimization in one-way free float car sharing services: a user based relocation strategy relying on price differentiation and urban area values. Technical report, Department of Computer, Control and Management Engineering, Universita'degli Studi di Roma "La Sapienza", 2019.

I. Barnes, K. Frick, E. Deakin, and A. Skabardonis. Impact of peak and off-peak tolls on traffic in san francisco–oakland bay bridge corridor in california. *Transportation Research Record*, 2297:73–79, 2012.

M. J. Barth, M. Todd, and L. Xue. User-based vehicle relocation techniques for multiple-station shared-use vehicle systems. 2004.

M. Bierlaire and S. Sharif Azadeh. Demand-based discrete optimization. Technical report, 2016.

E. Biondi, C. Boldrini, and R. Bruno. Optimal deployment of stations for a car sharing system with stochastic demands: A queueing theoretical perspective. *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1089–1095, 2016.

A. Bogyrbayeva, S. Jang, A. Shah, Y. J. Jang, and C. Kwon. A reinforcement learning approach for rebalancing electric vehicle sharing systems. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2021. doi: 10.1109/TITS.2021.3085217.

C. Boldrini, R. Incaini, and R. Bruno. Relocation in car sharing systems with shared stackable vehicles: Modelling challenges and outlook. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2017.

B. Boyacı, K. G. Zografos, and N. Geroliminis. An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational Research*, 240(3):718–733, 2015.

B. Boyacı, K. G. Zografos, and N. Geroliminis. An integrated optimization-simulation framework for vehicle and personnel relocations of electric carsharing systems with reservations. *Transportation Research Part B: Methodological*, 95:214–237, 2017.

G. Brandstätter, C. Gambella, M. Leitner, E. Malaguti, F. Masini, J. Puchinger, M. Ruthmair, and D. Vigo. Overview of optimization problems in electric car-sharing system design and management. *Dynamic Perspectives on Managerial Decision Making*, 22:441–447, 2016.

G. Brandstätter, M. Kahr, and M. Leitner. Determining optimal locations for charging stations of electric car-sharing systems under stochastic demand. *Transportation Research Part B: Methodological*, 104:17–35, 2017.

M. Bruglieri, A. Colorni, and A. Luè. The vehicle relocation problem for the one-way electric vehicle sharing: An application to the milan case. *Procedia - Social and Behavioral Sciences*, 111:18–27, 2014.

M. Bruglieri, F. Pezzella, and O. Pisacane. Tan adaptive large neighborhood search for relocating vehicles in electric carsharing services. *Discrete Applied Mathematics*, 253:185–200, 2019.

L. Cai, X. Wang, Z. Luo, and Y. Liang. A hybrid adaptive large neighborhood search and tabu search algorithm for the electric vehicle relocation problem. *Computers Industrial Engineering*, 167:108005, 2022.

E. M. Cepolina and A. Farina. A new shared vehicle system for urban areas. *Transportation Research Part C: Emerging Technologies*, 21(1):230–243, 2012.

Y. Cheng, X. Deng, and M. Zhang. A novel business model for electric car sharing. *International Workshop on Frontiers in Algorithmics*, pages 76–87, 2019.

C. F. Choudhury, L. Yang, J. de Abreu e Silva, and M. Ben-Akiva. Modelling preferences for smart modes and services: A case study in lisbon. *Transportation Research Part A: Policy and Practice*, 115:15–31, 08 2018.

R. Christian and E. Morin. Mobility services for urban sustainability: environmental assessment. *Moses Report WP6, Trivector Traffic AB*, 2005.

G. H. d. A. Correia and A. P. Antunes. Optimization approach to depot location and trip selection in one-way carsharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):233–247, 2012.

K. Crane, S. Nataraj, and L. E. S. H. Energy services analysis: An alternative approach for identifying opportunities to reduce emissions of greenhouse gases. *Technical report, RAND Corporation*, 2012.

F. de Donnea. Consumer behaviour, transport mode choice and value of time: Some micro-economic models. *Regional and Urban Economics*, 1(4):355–382, 1972.

M. Duncan. The cost saving potential of carsharing in a us context. *Transportation*, 33:363–382, 2011.

U. Eilertsen, O. M. Falck-Pedersen, J. V. Henriksen, K. Fagerholt, and G. Pantuso. A stochastic modeling approach to the electric vehicle relocation problem with dynamic pricing. pages 1–64, 2021.

M. Esmaeilpour, H. Yasinian, K. Esmaeilpour, R. Ramazani, and M.-H. Jahangiri. Reinforcement learning approach for customer behavior modeling gift credit optimal allocation. 05 2016.

W. D. Fan. Management of dynamic vehicle allocation for carsharing systems: Stochastic programming approach. *Transportation Research Record*, 2359(1):51–58, 2013.

C. Folkestad, N. Hansen, K. Fagerholt, H. Andersson, and G. Pantuso. Optimal charging and repositioning of electric vehicles in a free-floating carsharing system. *Computers Operations Research*, 113:104771, 08 2019.

L. C. E. Folkestad and M. D. Klev. The stochastic electric car-sharing relocation problem. an adaptive large neighborhood search. Master's thesis, NTNU, 06 2021.

D. K. George and C. K. Xia. Fleet-sizing and service availability for a vehicle rental system via closed queueing networks. *European Journal of Operational Research*, 211(1):198–207, 2011.

S. Hellem, C. A. Julsvoll, M. Moan, H. Andersson, K. Fagerholt, and G. Pantuso. The dynamic electric carsharing relocation problem. *EURO Journal on Transportation and Logistics*, 10, 2021. 100055.

A. Hottung and K. Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. *2019 ECAI 24th European Conference on Artificial Intelligence*, 41:443–450, 2019.

L. Hu and Y. Liu. Joint design of parking capacities and fleet size for one-way station-based carsharing systems with road congestion constraints. *Transportation Research Part B: Methodological*, 93:268–299, 2016.

K. Huang, K. An, and G. H. d. A. Correia. Planning station capacity and fleet size of one-way electric carsharing systems with continuous state of charge functions. *European Journal of Operational Research*, 287(3):1075–1091, 2020.

X. Huo, X. Wu, M. Li, N. Zheng, and G. Yu. The allocation problem of electric car-sharing system: A data-driven approach. *Transportation Research Part D: Transport and Environment*, 78:102192, 2020.

S. Illgen and M. Höck. Literature review of the vehicle relocation problem in oneway car sharing networks. *Transportation Research Part B: Methodological*, 120:193–204, 2019.

D. Jorge, G. Molnar, and G. H. de Almeida Correia. Trip pricing of one-way station-based carsharing networks with zone and time of day price variations. *Transportation Research Part B: Methodological*, 81:461–482, 2015.

T. Kamatani, Y. Nakata, and S. Arai. Dynamic pricing method to maximize utilization of one-way car sharing service. In *2019 IEEE International Conference on Agents (ICA)*, pages 65–68. IEEE, 2019.

A. Kek, R. Cheu, Q. Meng, and C. Fung. A decision support system for vehicle relocation operations in carsharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 45:149–158, 01 2009.

R. Kikuchi and H. Miwa. Dynamic pricing method for one-way car sharing service to meet demand and to maximize profit under given utility function. *Lecture Notes on Data Engineering and Communications Technologies*, 65:324–332, 2021.

R. Klein, S. Koch, C. Steinhardt, and A. K. Strauss. A review of revenue management: Recent generalizations and advances in industry applications. *European Journal of Operational Research*, 284(2):397–412, 2020.

K. S. Kühne, T. A. Rickenberg, and M. H. Breitner. An optimization model and a decision support system to optimize car sharing stations with electric vehicles. *Operations Research Proceedings 2014*, pages 313–320, 2016.

H. Lei, G. Laporte, and B. Guo. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers Operations Research*, 38:1775–1783, 2011.

X. Li, C. Wang, and X. Huang. A two-stage stochastic programming model for car-sharing problem using kernel density estimation. *Concordia University: arXiv*, 09 2019.

X. Li, C. Wang, and X. Huang. Ddksp: A data-driven stochastic programming framework for car-sharing relocation problem. *Concordia University: arXiv*, 2020.

H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

Q. H. Z. D. Luo, Z. and A. Lim. Adaptive large neighborhood search heuristics for the vehicle routing problem with stochastic demands and weight-related cost. *Transportation Research Part E: Logistics and Transportation Review*, 85:69–89, 2016.

E. Martin and S. Shaheen. Impacts of car2go on vehicle ownership, modal shift, vehicle miles traveled, and greenhouse gas emissions: An analysis of five north american cities. *Transportation Sustainability Research Center, UC Berkeley*, 3, 2016.

J. I. McGill and G. J. Van Ryzin. Revenue management: Research overview and prospects. *Transportation Science*, 33(2):233–256, 1999.

P. Modesti and A. Sciomachen. A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, 111(3):495–508, 1998.

R. Nair and E. Miller-Hooks. Fleet management for vehicle sharing operations. *Transportation Science*, 45 (4):524–540, 2011.

Y. Nourani and B. Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31(41):8373, 1998.

M. Nourinejad and M. J. Roorda. A dynamic carsharing decision support system. *Transportation Research Part E: Logistics and Transportation Review*, 66:36–50, 2014.

M. Nourinejad, S. Zhu, S. Bahrami, and M. J. Roorda. Vehicle relocation and staff rebalancing in one-way carsharing systems. *Transportation Research Part E*, 81(Complete):98–113, 2015.

G. Pantuso. Formulations of a carsharing pricing and relocation problem. In *International Conference on Computational Logistics*, pages 295–310. Springer, 2020.

C. Prince. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers Operations Research*, 31:1985–2002, 2004.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

A. Santini, S. Ropke, and L. M. Hvattum. A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, 24(5):783–815, 2018.

A. G. Santos, P. G. Cândido, A. F. Balardino, and W. Herbawi. Vehicle relocation problem in free floating carsharing using multiple shuttles. *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2544–2551, 2017.

T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96–115, 2005.

T. Schiller, J. Scheidl, and T. Pottebaum. Car sharing in europe - business models, national variations and upcoming disruptions. 2017. URL `https://www2.deloitte.com/content/dam/Deloitte/de/Documents/consumer-industrial-products/CIP-Automotive-Car-Sharing-in-Europe.pdf`. Accessed: 28.11.2021.

A. M. Selcuk and Z. M. Avṣar. Dynamic pricing in airline revenue management. *Journal of mathematical analysis and applications*, 478(2):1191–1217, 2019.

S. A. Shaheen, N. D. Chan, and H. Micheaux. One-way carsharing's evolution and operator perspectives from the americas. *Transportation*, 42(3):519–536, 2015.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *1998 CP International conference on principles and practice of constraint programming*, pages 417–431, 1998.

K. Tan, C. Cheong, and C. Goh. Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. *European Journal of Operational Research*, 177(2):813–839, 2007.

D. Wang and F. Liao. Analysis of first-come-first-served mechanisms in one-way car-sharing services. *Transportation Research Part B: Methodological*, 147:22–41, 2021.

A. Waserhole and V. Jost. Pricing in vehicle sharing systems: optimization in queuing networks with product forms. *EURO Journal on Transportation and Logistics*, 5(3):293–320, 2016.

A. Waserhole, V. Jost, and N. Brauner. Pricing techniques for self regulation in vehicle sharing systems. *Electronic Notes in Discrete Mathematics*, 41:149–156, 2013.

S. Weikl and K. Bogenberger. Relocation strategies and algorithms for free-floating car sharing systems. *IEEE Intelligent Transportation Systems Magazine*, 5(4):100–111, 2013.

M. Xu and Q. Meng. Fleet sizing for one-way electric carsharing services considering dynamic vehicle relocation and nonlinear charging profile. *Transportation Research Part B: Methodological*, 128:23–49, 2019.

Appendix A

Model Formulation

Objective Function

$$\max z = -\sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}}\sum_{m\in\mathcal{M}} C^R T_r^H x_{erm} + \sum_{s\in\mathcal{S}} P_s \sum_{d\in\mathcal{D}(\xi_s)}\sum_{v\in\mathcal{V}}\sum_{l\in\mathcal{L}_d} R_{dl} y_{vdls} \tag{A.1}$$

Constraints

Pricing of EVs

$$\sum_{l\in\mathcal{L}} \lambda_{ijl} = 1, \qquad i,j \in \mathcal{I} \tag{A.2}$$

Availability of EVs

$$1 - \sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} = \sum_{i\in\mathcal{I}} z_{iv}, \qquad v \in \mathcal{V}^B \tag{A.3}$$

$$\sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_i^N\cap\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} = z_{iv}, \qquad i \in \mathcal{I}\backslash\{o(v)\}, v \in \mathcal{V}\backslash\{\mathcal{V}^B\} \tag{A.4}$$

$$1 - \sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} = z_{o(v),v}, \qquad v \in \mathcal{V} \tag{A.5}$$

$$\sum_{i\in\mathcal{I}} z_{iv} \leq 1, \qquad v \in \mathcal{V} \qquad \text{(A.6)}$$

Relocation of EVs

$$\sum_{r\in\mathcal{R}} x_{e,r,(m+1)} \leq \sum_{r\in\mathcal{R}} x_{erm}, \qquad e \in E, m \in \mathcal{M}\backslash\{|\mathcal{M}|\} \qquad \text{(A.7)}$$

$$\sum_{r\in\mathcal{R}} x_{erm} \leq 1, \qquad e \in \mathcal{E}, m \in \mathcal{M} \qquad \text{(A.8)}$$

$$\sum_{e\in\mathcal{E}} \sum_{r\in\mathcal{R}_v^V} \sum_{m\in\mathcal{M}} x_{erm} \leq 1, \qquad v \in \mathcal{V} \qquad \text{(A.9)}$$

$$\sum_{e\in\mathcal{E}} \sum_{i\in\mathcal{I}^{CS}} \sum_{r\in\mathcal{R}_i^{CD}} \sum_{m\in\mathcal{M}} x_{erm} + \sum_{s\in\mathcal{S}} P_s W(z,\lambda,\xi_s) \geq N^B \cdot |\mathcal{V}^B| \qquad \text{(A.10)}$$

Time Usage

$$t_{em} + T_r^H \cdot x_{erm} + \sum_{r_1\in\mathcal{R}\backslash\{r\}} T_{d(r),o(r_1)} x_{e,r_1,(m+1)} - M_r(1 - x_{erm}) \leq t_{e,(m+1)},$$

$$e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M}\backslash\{|\mathcal{M}|\} \qquad \text{(A.11)}$$

$$t_{em} \leq t_{e,(m+1)}, \qquad e \in \mathcal{E}, m \in \mathcal{M}\backslash\{|\mathcal{M}|\} \qquad \text{(A.12)}$$

$$(T_e^{SO} + T_{o(e),o(r)}) \cdot x_{er1} \leq t_{e1}, \qquad e \in \mathcal{E}, r \in \mathcal{R} \qquad \text{(A.13)}$$

$$T_r^{SC} x_{erm} \leq t_{em}, \qquad e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M} \qquad \text{(A.14)}$$

$$t_{e|\mathcal{M}|} + \sum_{r\in\mathcal{R}} T_r^H x_{er|\mathcal{M}|} \leq \overline{T}^1, \qquad e \in \mathcal{E} \qquad \text{(A.15)}$$

Request Handling

$$\sum_{e\in\mathcal{E}} \sum_{v\in\mathcal{V}^B} \sum_{r\in\mathcal{R}_i^{CD}\cap\mathcal{R}_v^V} \sum_{m\in\mathcal{M}} x_{erm} + \sum_{v\in\mathcal{V}^B} \sum_{d\in\mathcal{D}_i^{CD}(\xi_s)} \sum_{l\in\mathcal{L}} y_{vdls} \leq N_i^{CS}, \qquad i \in \mathcal{I}^{CS}, s \in \mathcal{S} \qquad \text{(A.16)}$$

$$W(z,\lambda,\xi_s) = \sum_{v\in\mathcal{V}^B}\sum_{d\in\mathcal{D}^{CD}(\xi_s)}\sum_{l\in\mathcal{L}} y_{vdls}, \qquad s\in\mathcal{S} \qquad \text{(A.17)}$$

$$\sum_{v\in\mathcal{V}}\sum_{l\in\mathcal{L}_d} y_{vdls} \le 1, \qquad d\in\mathcal{D}(\xi_s), s\in\mathcal{S} \qquad \text{(A.18)}$$

$$\sum_{d\in\mathcal{D}(\xi_s)}\sum_{l\in\mathcal{L}_d} y_{vdls} \le 1, \qquad v\in\mathcal{V}, s\in\mathcal{S} \qquad \text{(A.19)}$$

$$\sum_{l\in\mathcal{L}_{d_1}(\xi_s)} y_{vd_1ls} + \sum_{d_2\in\mathcal{D}_{d_1}(\xi_s)}\sum_{l\in\mathcal{L}_{d_2}} y_{vd_2ls} \le z_{i(d_1),v}, \qquad d_1\in\mathcal{D}(\xi_s), v\in\mathcal{V}, s\in\mathcal{S} \qquad \text{(A.20)}$$

$$y_{vd_1l_1s} + \sum_{d_2\in\mathcal{D}_{d_1}(\xi)}\sum_{l_2\in\mathcal{L}_{d_2}(\xi)} y_{vd_2l_2s} + \sum_{v_1\in\mathcal{V}:v_1\neq v} y_{v_1d_1l_1s} \ge \lambda_{i(d_1),j(d_1),l_1} + z_{i(d_1),v} - 1,$$

$$d_1\in\mathcal{D}(\xi_s), v\in\mathcal{V}, l_1\in\mathcal{L}_{d_1}, s\in\mathcal{S} \qquad \text{(A.21)}$$

$$\sum_{v\in\mathcal{V}} y_{vdls} \le \lambda_{i(d),j(d),l}, \qquad d\in\mathcal{D}(\xi_s), l\in\mathcal{L}_d, s\in\mathcal{S} \qquad \text{(A.22)}$$

Binary, Non-Negativity and Integer Definitions

$$\lambda_{ijl}\in\{0,1\}, \qquad i,j\in\mathcal{I}, l\in\mathcal{L} \qquad \text{(A.23)}$$

$$z_{iv}\in\{0,1\}, \qquad i\in\mathcal{I}, v\in\mathcal{V} \qquad \text{(A.24)}$$

$$x_{erm}\in\{0,1\}, \qquad e\in\mathcal{E}, r\in\mathcal{R}, m\in\mathcal{M} \qquad \text{(A.25)}$$

$$t_{em}\ge 0, \qquad e\in\mathcal{E}, m\in\mathcal{M} \qquad \text{(A.26)}$$

$$y_{vdls}\in\{0,1\}, \qquad v\in\mathcal{V}, d\in\mathcal{D}(\xi_s), l\in\mathcal{L}, s\in\mathcal{S} \qquad \text{(A.27)}$$

Notation List

Table A.1: Sets used in the model.

Notation	Explanation
$\mathcal{I}$	Set of zones
$\mathcal{I}^{CS}$	Set of charging zones
$\mathcal{E}$	Set of employees
$\mathcal{M}$	Set of abstract employee tasks
$\mathcal{V}$	Set of car-sharing vehicles
$\mathcal{V}^B$	Set of vehicles in need of charging
$\mathcal{R}$	Set of car-moves
$\mathcal{R}_v^V$	Set of car-moves for vehicle v
$\mathcal{R}_i^N$	Set of car-moves with destination zone i
$\mathcal{R}_i^{CD}$	Set of car-moves with destination zone $i \in \mathcal{I}^{CS}$
$\mathcal{K}$	Set of customers
$\mathcal{K}_i$	Set of customers travelling from zone i
$\mathcal{K}_{ij}$	Set of customers travelling from zone i to j
$\mathcal{S}$	Set of scenarios
$\mathcal{L}$	Set of possible combined pick-up and drop-off fee levels
$\mathcal{D}(\xi_s)$	Set of requests for a given realization of ξ_s in scenario s
$\mathcal{D}^{CD}(\xi_s)$	Set of requests with destination in a charging zone for a given realization of ξ_s in scenario s
$\mathcal{D}_i^{CD}(\xi_s)$	Set of requests with destination in charging zone i for a given realization of ξ_s in scenario s
$\mathcal{D}_d(\xi_s)$	Set of requests that has precedence over request d in scenario s
$\mathcal{D}_{ij}(\xi_s)$	Set of requests going from zone i to j in scenario s
$\mathcal{L}_d$	Set of combined pick-up and drop-off fee levels that are less than or equal to the highest fee accepted for request d

Table A.2: Parameters used in the model.

Notation	Explanation
C^R	Accumulated toll, maintenance and energy (electricity) cost per minute of driving a car
T_r^H	The time it takes to perform the car-move r
P_s	Probability of scenario s occurring
R_{dl}	Profit of satisfying request d with pick-up fee level l
$o(v)$	Origin of car v at start of planning horizon
$o(r)$, $d(r)$	Origin and destination of car-move r, respectively
T_e^{SO}	Earliest start time employee e is available to perform a task
$o(e)$	Origin of employee e at start of planning horizon
$\overline{T}^1$	Time length of the first stage
T_r^{SC}	First available time a car is available for car-move r
T_{ij}	The time it takes to travel between zone i and j by bicycle
L_l	Value of a fee at fee level l
N_i^{CS}	Number of available charging spots in zone $i \in \mathcal{I}^{CS}$
N^B	Share of cars in need of charging which needs to be charged in the planning horizon
$i(d)$, $j(d)$	Origin and destination of customer request d, respectively
$k(d)$	Customer of customer request d
$l(d)$	The highest fee level at which customer $k(d)$ would prefer car-sharing to other transportation modes
M_r	Big-M notation for each car-move r

Table A.3: Decision variables used in the model.

Notation	Explanation
z_{iv}	1 if vehicle v is available to customers in zone i at the beginning of the second stage, at time $\overline{T}^1$, 0 otherwise.
λ_{ijl}	1 if fee level l is used between zone i and j, 0 otherwise.
x_{erm}	1 if employee e makes relocation move r as her task number m, 0 otherwise.
y_{vdls}	1 if vehicle v satisfies request d with the fee level l in scenario s, 0 otherwise.

Appendix B

Big-M Calculation

The Big-M parameters in Constraints 4.11 are the smallest value making the constraints redundant when not applicable, i.e., every case where the decision variable $x_{erm} = 0$. Setting x_{erm} to zero gives us the following:

$$\sum_{r_1 \in \mathcal{R} \setminus \{r\}} T_{d(r),o(r_1)} x_{e,r_1,(m+1)} - (t_{e,(m+1)} - t_{em}) \leq M_r, \qquad e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M} \setminus \{|\mathcal{M}|\}$$

Constraints 4.8 and 4.9 assure that $x_{e,r_1,(m+1)}$ will be equal to one at most once for a given car-move r_1. We therefore simplify the expression above to the following:

$$T_{d(r),o(r_1)} - (t_{e,(m+1)} - t_{em}) \leq M_r, \qquad e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M} \setminus \{|\mathcal{M}|\}$$

To find the tightest expression, we seek to find the lowest possible values for M_r. This is done by maximizing the left term of the expression:

$$M_r = \max T_{d(r),o(r_1)} - (t_{e,(m+1)} - t_{em}), \qquad e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M} \setminus \{|\mathcal{M}|\}$$

The first term in the above expression is the travel time from the destination of car-move r to the origin of car-move r_1. The second term is the time between the beginning of employee e's next task, and the beginning of its current task. The combination of these two terms represent a trip ending in the origin of the next car-move, r_1, and we can therefore express the term as:

$$M_r = \max_{r_1 \in \mathcal{R} \setminus \{r\}, i \in \mathcal{I}} T_{d(r),i} - (T^H_{r_1} + T_{d(r_1),i}), \qquad r \in \mathcal{R}$$

Appendix C

Example Solution

This appendix presents a possible solution to a small example problem. The purpose is to give a deeper insight into how the model works, how decisions are made, and the timeline. Section C.1 presents the initial state of the model, while Sections C.2 and C.3 describe the first and second stages, respectively.

C.1 Initial State

Each run of the model starts with an initial state. This state is replicated in Figure C.1. The elements present are geographical zones, employees, and cars. In the example, we operate with nine zones. Two of the zones have charging stations, making them charging zones. Both of the charging zones have one available charging slot each. The employees are initiated with a start location, and a time they become available to perform their first relocation. Employee 1 is located in zone 9 and is available at time 0, while employee 2 is on its way to zone 7, arriving at time 7. The planning horizon is set to 60 minutes. The cars are initiated similarly to the employees, with a start location and a start time. There are in total 10 cars. The seven black cars are sufficiently charged, while the three red ones require charging. All cars are available for relocation at time 0.

Figure C.1 presents all the information known to the CSO at the beginning of the planning horizon. Information concerning customer requests is a stochastic element captured by different scenarios. In this example, the first stage lasts from time 0 to time 30, while the second stage lasts from time 30 to time 60. There are two second-stage scenarios. Decisions made in the first stage are illustrated in Section C.2, before Section C.3 elaborates the second stage and its two scenarios.

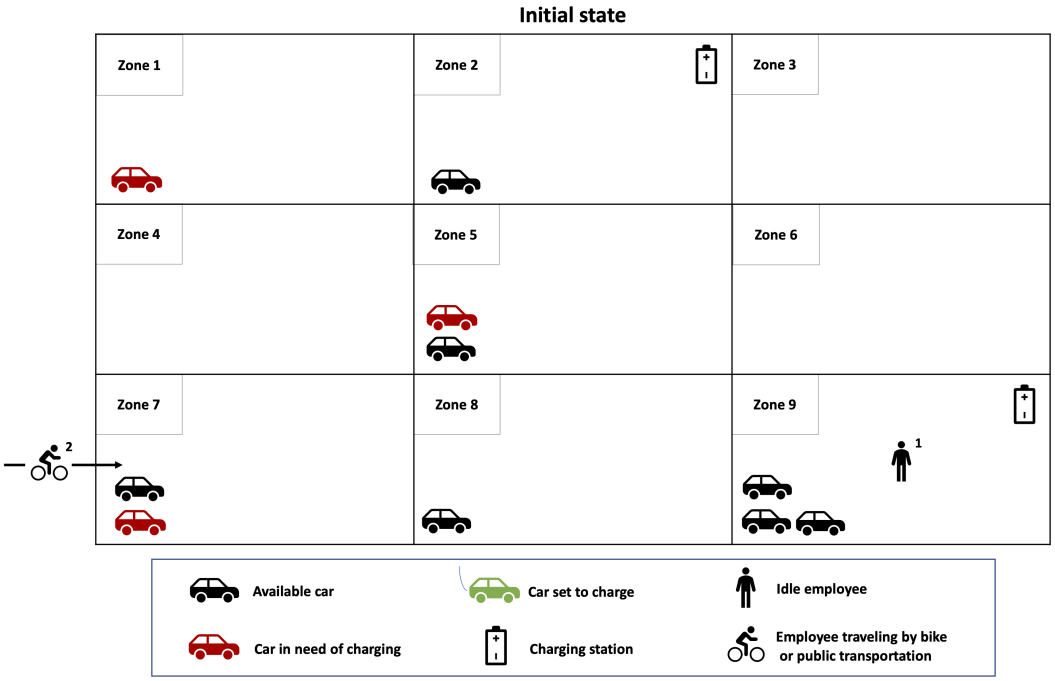


Figure C.1: Initial state.

C.2 First Stage (Time 0 to Time 30)

Table C.1: Pick-up and drop-off fees from zone to zone

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	20	-100	-100	-80	-80	-20	100	20
2	0	60	40	100	40	-60	-20	20	20
3	0	-20	-100	40	60	-20	-80	60	20
4	0	-80	20	-60	-80	-20	-100	40	0
5	0	80	-100	40	-80	80	-20	20	0
6	0	0	20	-100	60	100	-60	20	20
7	0	20	-100	-100	-0	-40	-20	-40	20
8	0	80	-100	20	-80	-40	-60	20	20
9	0	80	40	-100	60	-20	-80	20	60

During the first stage, decisions concerning relocation and pricing are made. The combined pick-up and drop-off fees for a car-sharing trip between each pair of zones are set and are given in Table C.1. These fees come in addition to the fixed per-minute fee, equal for all car-sharing trips. The routing and scheduling of each employee are decided and given as follows.

At time 0, employee 1 starts to relocate a car from zone 9 to zone 8. Employee 2 is traveling by bicycle, on its way towards zone 7. This is shown in Figure C.2.

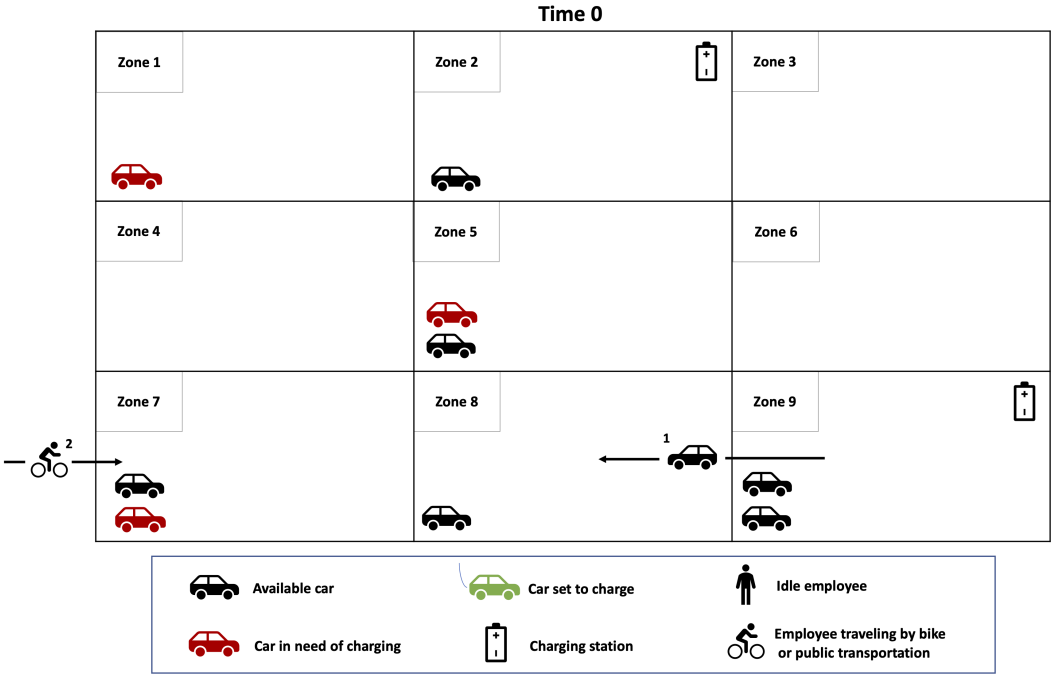


Figure C.2: Time 0. Employee 1 moves a car from zone 9 to zone 8, while employee 2 is on its way to zone 7.

At time 7, employee 2 arrives at zone 7 and immediately starts to relocate the car needing charging from zone 7 to zone 2. Employee 1 is still relocating a car between zone 9 and zone 8. This is shown in Figure C.3.

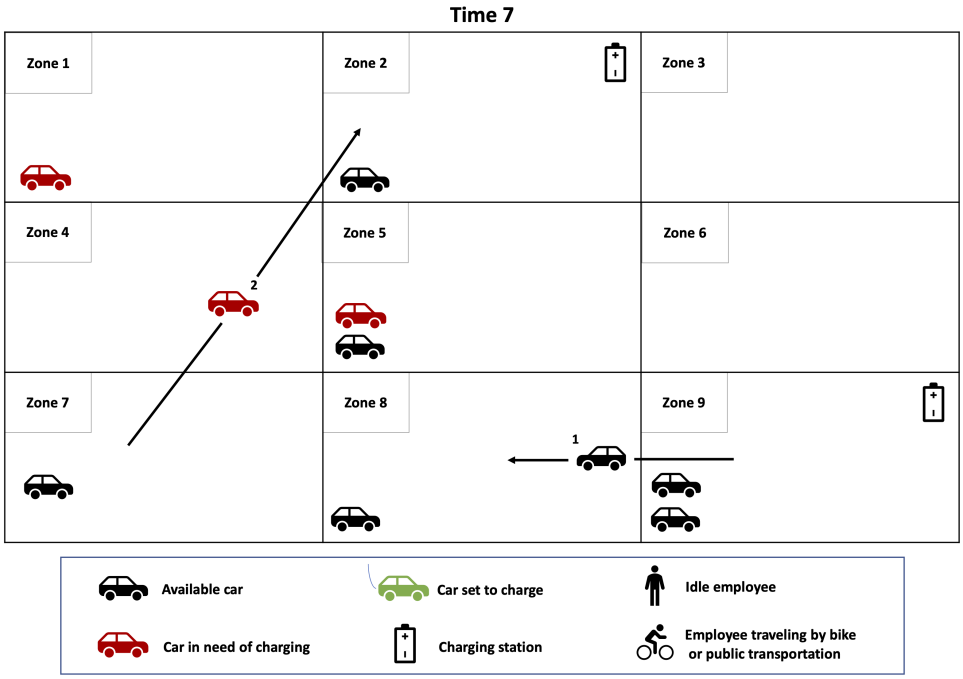


Figure C.3: Time 7. Employee 1 is still moving a car from zone 9 to 8, while employee 2 moves the car in need of charging from zone 7 to zone 2.

At time 12, employee 1 arrives at zone 8 and completes its first relocation, after which it immediately starts to travel by bicycle to zone 5. Employee 2 is still relocating a car to zone 2. This is shown in Figure C.4.

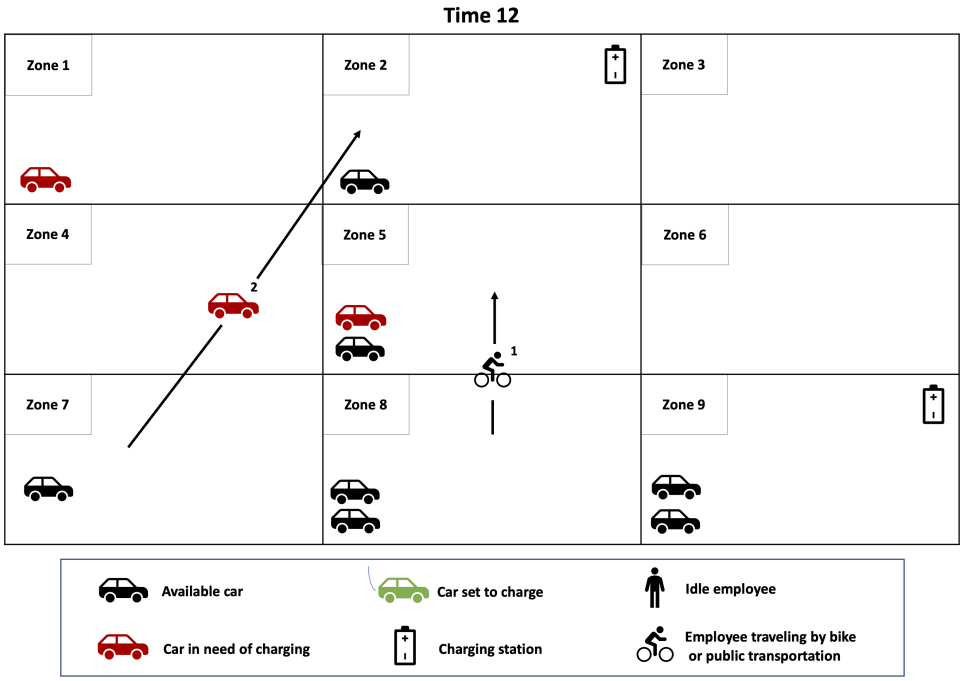


Figure C.4: Time 12. Employee 1 travels by bicylce to zone 5, while employee 2 is still moving the car in need of charging to zone 2.

At time 19, employee 1 arrives at zone 5, immediately initiating a car relocation from zone 5 to zone 6. Employee 2 is still relocating a car to zone 2, as seen in Figure C.5.

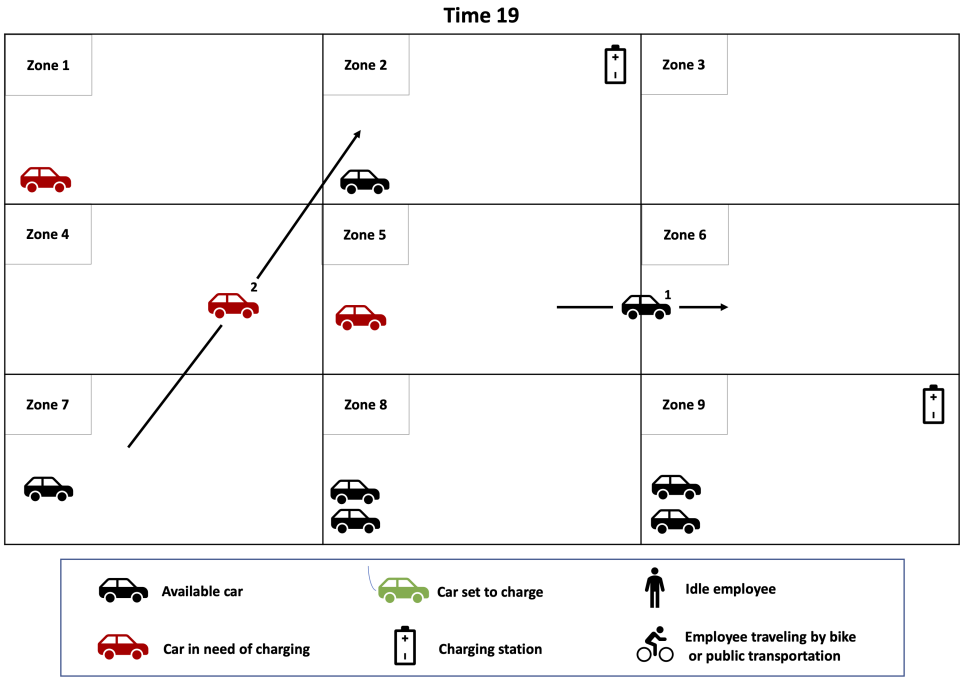


Figure C.5: Time 19. Employee 1 arrives at zone 5, and starts moving a car to zone 6. Employee 2 is still relocating the car in need of charging to zone 2.

At time 28, both employees finish their current relocation tasks. As there is insufficient time left to perform further relocations, both employees stay idle, and the states of time 28 and 30 are equal. This is shown in Figure C.6 and marks the end of the first stage.

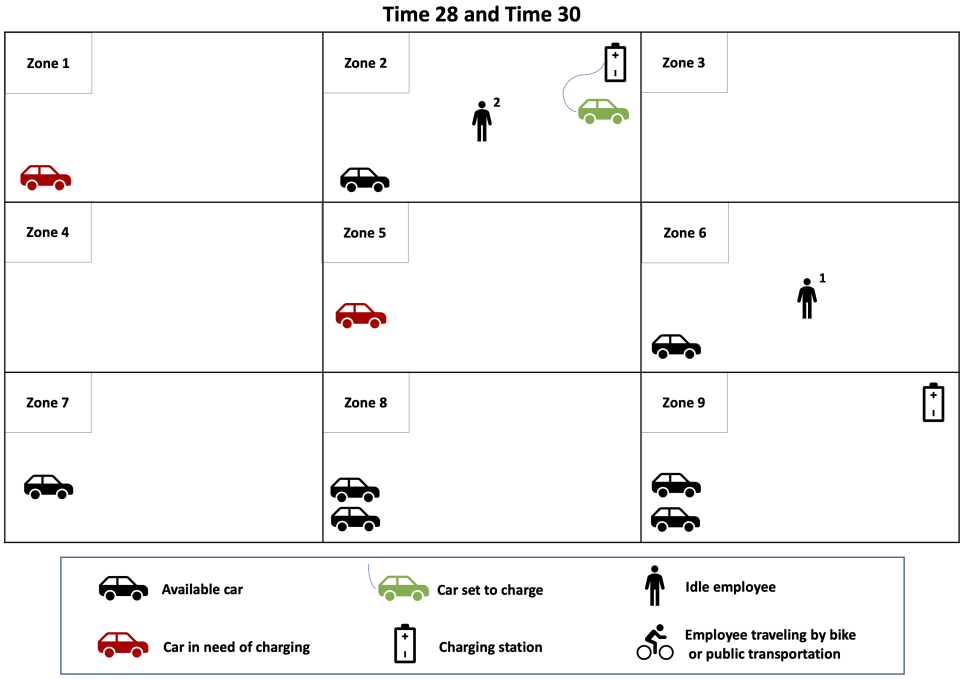


Figure C.6: Times 28 and 30. Both employees remain idle, and the first stage is over.

C.3 Second Stage (Time 30 to Time 60)

At time 30 in our example, we enter the second stage of the model. Relocations by employees are complete, and the pick-up and drop-off fees for car-sharing travels between each pair of zones are set according to Table C.1. For simplicity, employee information is removed from further visualizations, as it is no longer relevant.

At this point, potential customers are realized. Each potential customer has a start location and wishes to travel to another zone, as shown in Table C.2. Whether or not they want to use the car-sharing service depends on their preferences, measured through individual utilities. The customer utility is calculated for each transportation mode, i.e., car-sharing, public transportation, and bicycles. Depending on the price, waiting time, walking distance, and travel time with the different transportation modes, each customer assigns a utility value to use them. The customers have an additional utility factor based on elements only known to the customers themselves, making the total utility value unpredictable to the CSO. If the utility for car-sharing is the highest for a potential customer, it becomes a realized customer and generates a request to use a car. The customer IDs are assigned in ascending order according to when the customer requests appear in time. The customers are prioritized based on their IDs. If both customers 1 and 2 request the same car, customer 1 is prioritized. The potential customers and the state information at the beginning of the second stage are seen in Figure C.7. This information is equal in both scenarios.

The second stage consists of two scenarios in our example. It is the customers' requests for travel by car-sharing service that vary between the scenarios. This is because the individual element in the customers' preferences varies in each scenario, allowing a customer to request a car in one scenario and not request one in another under otherwise identical conditions. Whether or not the customer is satisfied will be given by different dependencies. This is investigated in the demonstration of scenario 1 and scenario 2 in Sections

C.3.1 and C.3.2, respectively.

Table C.2: Customer requests from zone to zone

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
From	1	2	2	3	4	4	5	6	7	7	7	8	8	8	9
To	9	5	6	4	8	9	2	3	1	5	6	3	7	1	3

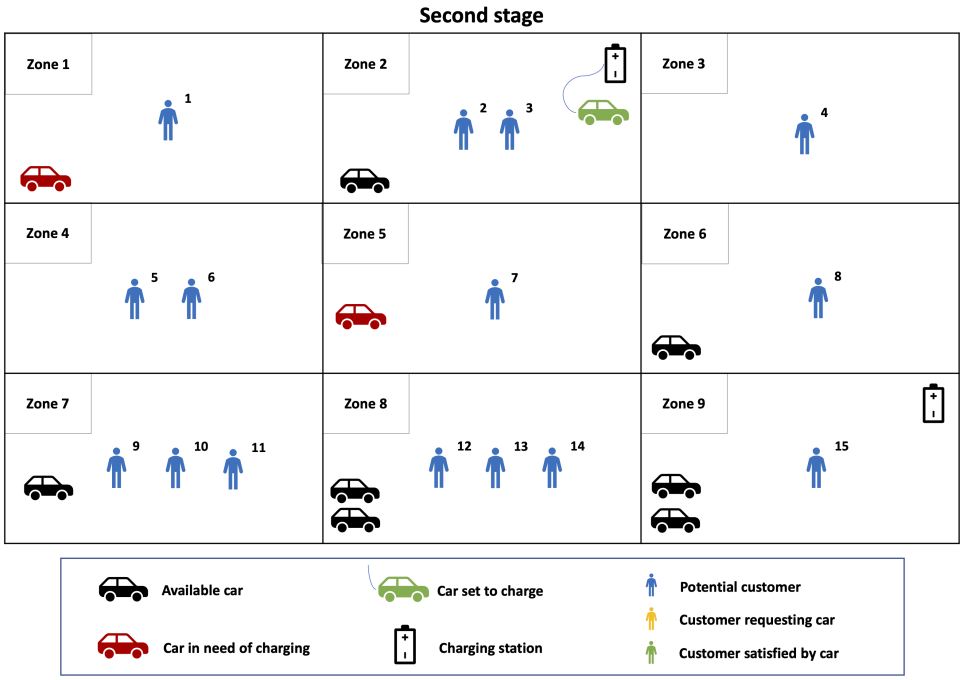


Figure C.7: Initial second-stage state information. The state information is given by decisions in the first stage, as well as the realization of potential customers.

C.3.1 Scenario 1 (Time 30 to Time 60)

In scenario 1, seven of the 15 potential customers become realized customers requesting a car to complete their trips. The remaining eight potential customers prefer other means of transport, making them superfluous to the decisions in scenario 1. The current state with the realized customer requests is given in Figure C.8.

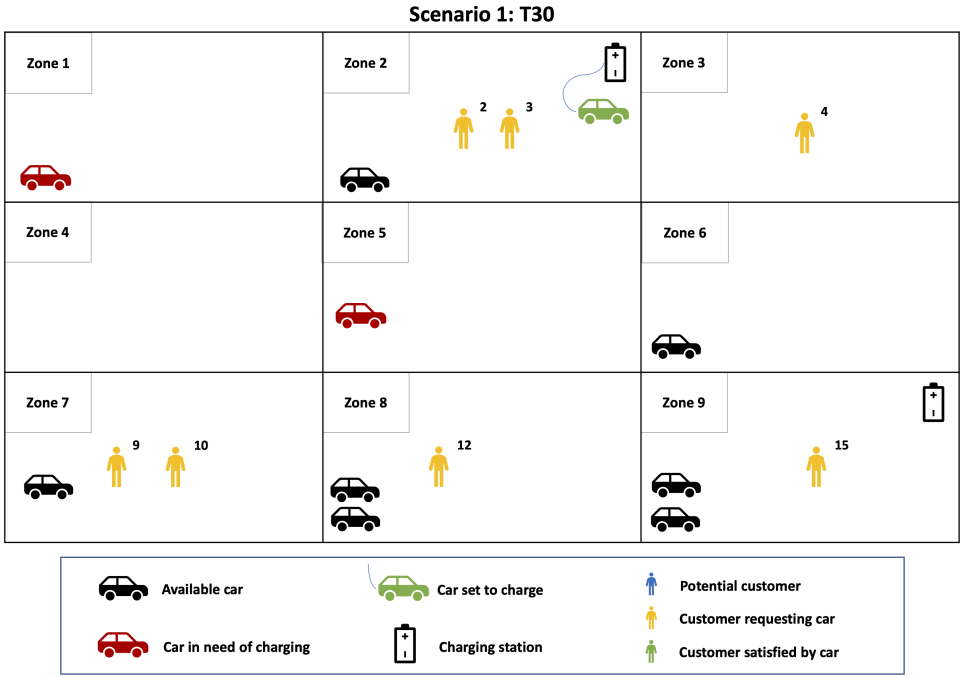


Figure C.8: The state of scenario 1 after the realization of customer requests. Customers preferring other means of transport are removed for simplicity.

Now is the time to allocate cars to customers to satisfy the requests. Both customer 2 and customer 3 are located in zone 2, requesting a car to go to zones 5 and 6, respectively. However, there is only one car available, as one of the cars in zone 2 is set to charge. The remaining car is allocated to customer 2, satisfying its trip to zone 5. Customer 2 is prioritized, as customer 2 has a lower ID number than customer 3. The car in zone 7 is similarly allocated to customer 9 over customer 10, satisfying customer 9's travel to zone 1. Customer 4 is in a zone without available cars. As customers cannot use a car if there are no available cars in their origin zone, customer 4 has to use another transportation mode. This is given by the assumptions described in Section 4.2. Finally, customers 12 and 15 pick a car from their respective zones and travel to zone 3. The trips and the final scenario state are given in Figures C.9 and C.10, respectively. Customer 3, 4 and 10 are left to travel by other means of transport, as there are no available cars in their respective zones.

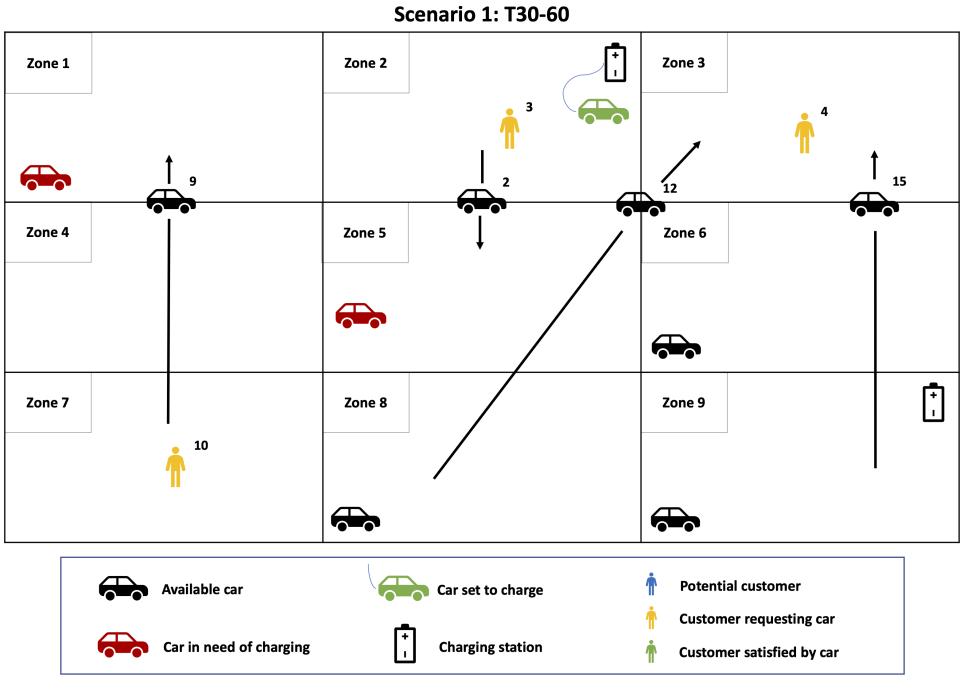


Figure C.9: The allocation of available cars to realized customers. Customers 2, 9, 12 and 15 travel by car, while customer 3, 4 and 10 are left to travel by other means of transport, as there are no available cars in their respective zones.

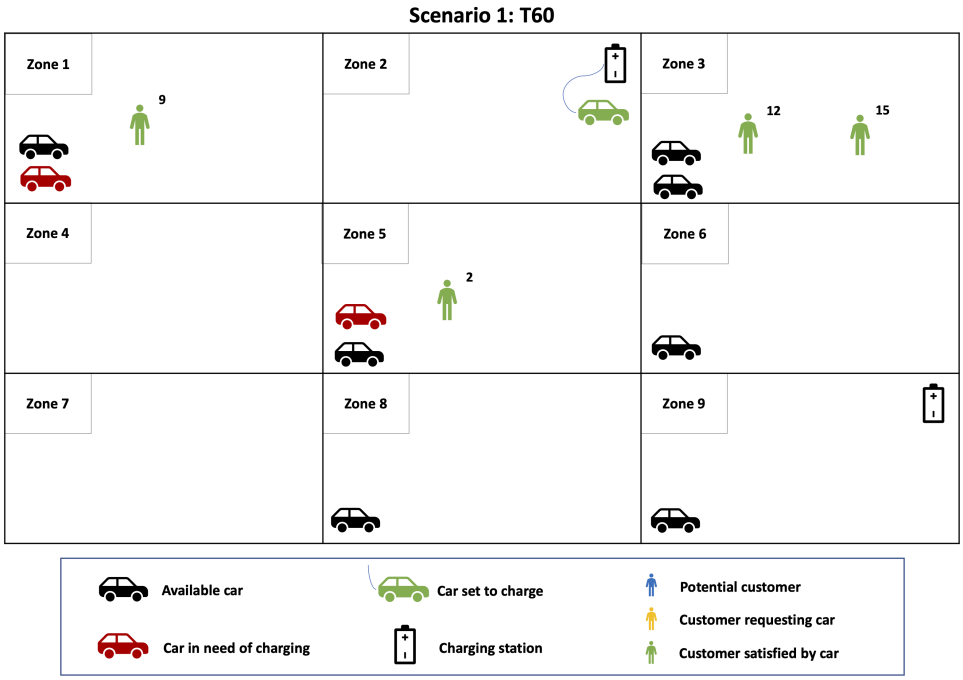


Figure C.10: The final state of scenario 1. Four customer requests were satisfied.

C.3.2 Scenario 2 (Time 30 to Time 60)

In scenario 2, the initial state is similar to scenario 1, given in Figure C.7. This time, however, six of the 15 potential customers become realized customers. Only customer 12 is recognized from scenario 1; the remaining five customers preferred other means of transport in scenario 1. The current state with the realized customer requests is given in Figure C.11. Again, the remaining nine potential customers are superfluous to the decisions to be made in scenario 2.

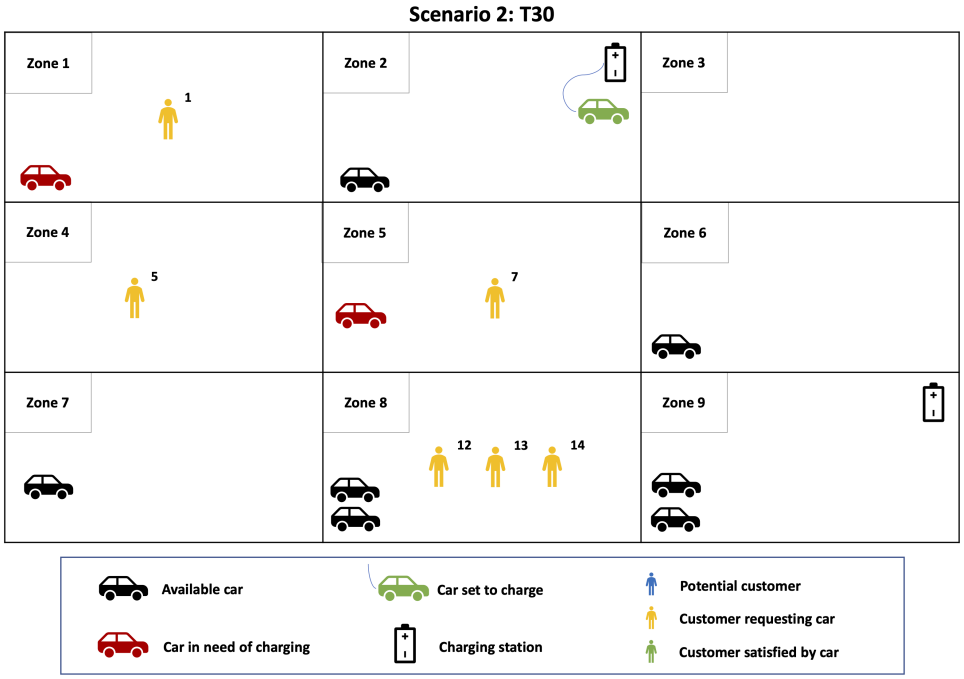


Figure C.11: The state of scenario 2 after the realization of customer requests. Customers preferring other means of transport are removed for simplicity.

Customer 1 wishes to go to zone 9 by car. There is a car available in zone 1 in need of charging. This is, however, not a problem for customer 1, as there is an available charging slot in zone 9. Customer 1 uses the car in need of charging and is thus satisfied. Customer 5 is in a zone where no cars are available and cannot travel by car. Customer 7 wishes to travel from zone 5 to zone 2 by car. However, the only available car in zone 5 needs charging, and the charging station in zone 2 is full. Therefore, customer 7 cannot travel by car. Finally, customers 12, 13, and 14 want to travel by car to zones 3, 7, and 1, respectively. Given the specific customer IDs, customers 12 and 13 are satisfied, while customer 14 is left without any available cars. The trips and the final scenario state are given in Figures C.12 and C.13, respectively.

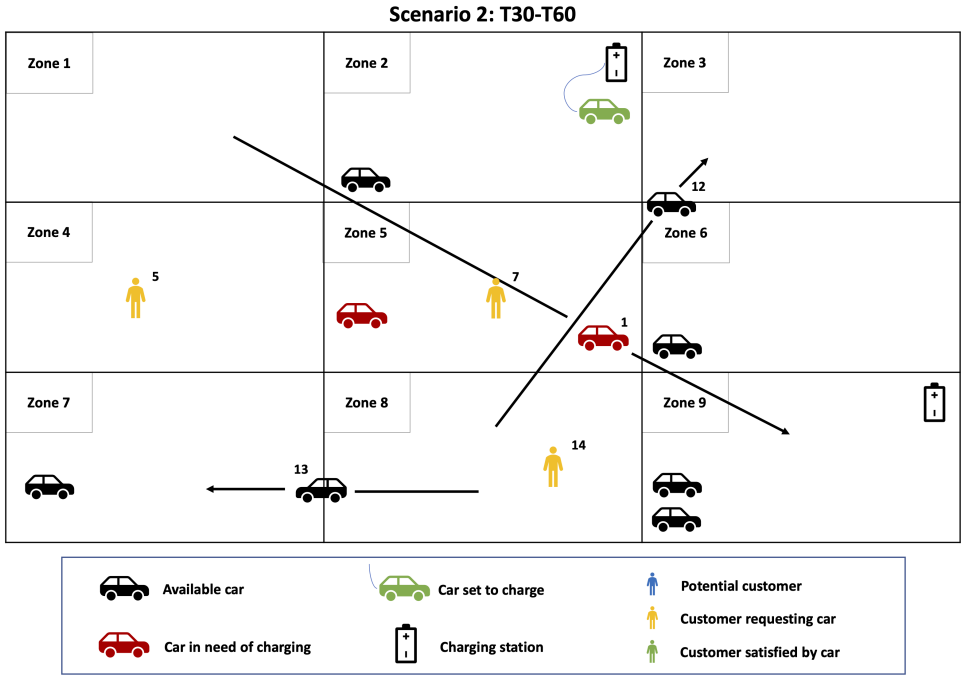


Figure C.12: The allocation of available cars to realized customers. Customer 1 travels by car needing charging to a charging zone with sufficient charging capacity. Customers 12 and 13 travel by car, while customers 5, 7 and 14 are left to travel by other means of transport due to lack of available cars in their respective zones.

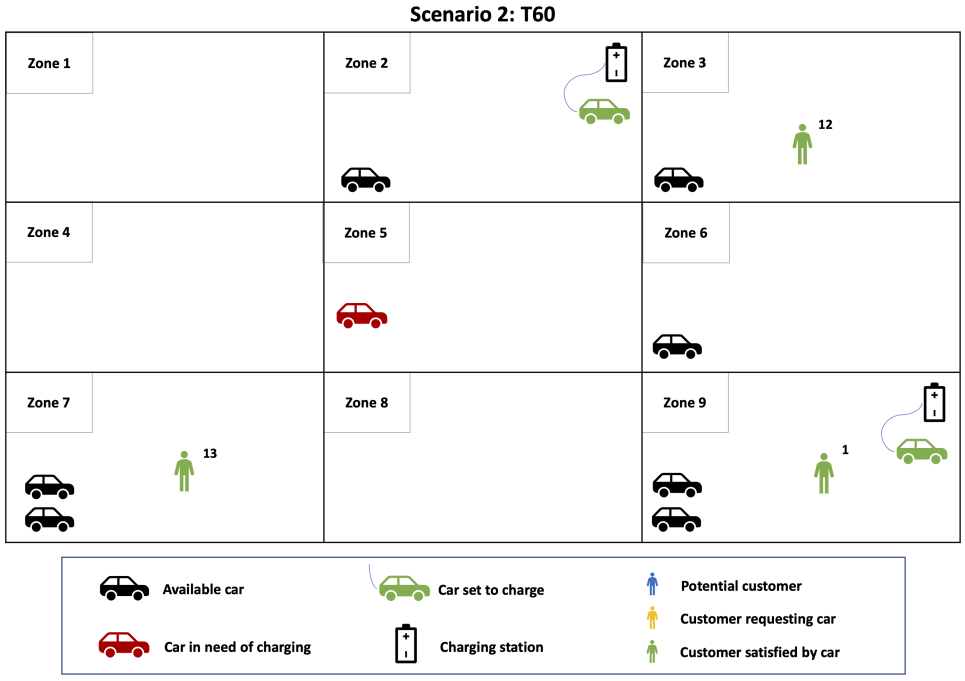


Figure C.13: The final state of scenario 2. Three customer requests were satisfied.

Appendix D

Parameter Tuning

This appendix provides extensive results from the tuning process of the ALNS heuristic parameters. Descriptions concerning the parameters tuned and the test instances used are provided in each table.

Table D.1: Results from the parameter tuning of ALNS parameter *search space parameters* Φ. Here, Φ includes $\phi_{customer}$, $\phi_{cars-to-customers}$ and $\phi_{relocation-time}$, respectively. The tuning has been performed on three different instances of each test instance type. The ALNS heuristic has attempted to solve the problem five times for each value of Φ for each instance. Columns Gap (%) indicate the gap in objective value between the best known solution for the test instance and the average objective value over the five runs. Columns Time (s) represent the average runtime over the five runs to finding the best solution for an instance.

Test Instance	Φ = **(0.3,0.3,0.3)**		Φ = **(0.4,0.4,0.4)**		Φ = **(0.5,0.5,0.5)**		Φ = **(0.6,0.6,0.6)**		Φ = **(0.7,0.7,0.7)**	
	Gap (%)	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**
5-4-1-a	0.02	600.61	0.02	600.99	0.03	601.02	0.02	600.51	0.03	600.78
5-4-1-b	0.04	601.88	0.01	602.07	0.02	493.71	0.01	601.96	0.00	496.92
5-4-1-c	0.01	487.35	0.03	378.26	0.06	389.80	0.07	601.36	0.02	519.98
Average 5-4-1	0.03	563.28	0.02	527.11	0.04	494.84	0.03	601.28	0.02	539.23
5-6-1-a	0.01	524.98	0.00	531.15	0.00	603.92	0.00	606.07	0.00	604.66
5-6-1-b	0.04	601.68	0.02	602.31	0.01	604.53	0.00	601.42	0.00	601.85
5-6-1-c	0.03	489.94	0.06	603.78	0.02	603.77	0.02	602.28	0.02	604.86
Average 5-6-1	0.03	538.87	0.03	579.08	0.01	604.07	0.01	603.26	0.01	603.79
7-8-1-a	0.08	610.66	0.07	606.10	0.07	606.70	0.04	617.73	0.01	611.29
7-8-1-b	0.10	609.45	0.01	613.64	0.01	609.66	0.00	610.26	0.00	609.38
7-8-1-c	0.18	606.07	0.05	618.38	0.05	552.47	0.03	607.96	0.01	611.48
Average 7-8-1	0.12	608.73	0.04	612.71	0.04	589.61	0.02	611.98	0.01	610.72
10-9-2-a	0.12	610.61	0.09	614.89	0.06	612.62	0.03	615.74	0.04	613.17
10-9-2-b	0.12	610.49	0.09	617.83	0.08	613.87	0.03	620.80	0.03	618.58
10-9-2-c	0.21	614.24	0.07	614.75	0.04	610.83	0.03	620.20	0.04	610.66
Average 10-9-2	0.15	611.78	0.08	615.83	0.06	612.44	0.03	618.91	0.04	614.14
15-12-2-a	0.12	611.28	0.08	618.38	0.08	621.24	0.08	610.32	0.07	621.81
15-12-2-b	0.07	618.42	0.07	618.72	0.03	627.34	0.05	620.99	0.05	621.93
15-12-2-c	0.15	627.66	0.12	618.80	0.05	630.61	0.06	617.53	0.04	624.04
Average 15-12-2	0.11	619.12	0.09	618.64	0.05	626.39	0.06	616.28	0.05	622.60
20-15-2-a	0.07	624.34	0.09	631.67	0.06	627.35	0.07	622.63	0.05	621.89
20-15-2-b	0.14	625.73	0.08	621.34	0.07	621.19	0.05	631.51	0.05	617.96
20-15-2-c	0.10	616.80	0.05	623.57	0.03	625.89	0.04	622.68	0.01	628.86
Average 20-15-2	0.11	622.29	0.07	625.53	0.05	624.81	0.05	625.61	0.04	622.90
30-20-2-a	0.14	625.83	0.11	622.01	0.06	616.53	0.07	614.67	0.03	623.05
30-20-2-b	0.15	641.08	0.11	614.98	0.10	627.18	0.06	625.12	0.04	633.08
30-20-2-c	0.13	629.67	0.06	613.46	0.05	638.85	0.04	631.03	0.03	632.59
Average 30-20-2	0.14	632.19	0.09	616.82	0.07	627.52	0.06	623.60	0.03	629.58
40-25-2-a	0.13	617.11	0.10	636.77	0.09	622.55	0.07	620.08	0.08	636.93
40-25-2-b	0.11	625.77	0.09	639.90	0.06	635.39	0.06	635.83	0.08	621.65
40-25-2-c	0.07	628.30	0.07	622.50	0.06	651.50	0.06	622.62	0.04	639.71
Average 40-25-2	0.10	623.73	0.09	633.06	0.07	636.48	0.06	626.18	0.07	632.76
50-30-2-a	0.08	612.38	0.07	655.97	0.06	668.30	0.15	700.16	0.10	679.99
50-30-2-b	0.23	670.17	0.23	681.46	0.09	687.73	0.20	675.58	0.16	736.94
50-30-2-c	0.05	668.80	0.06	668.58	0.06	668.07	0.05	681.05	0.03	641.81
Average 50-30-2	0.12	650.45	0.12	668.67	0.07	674.70	0.13	685.59	0.10	686.25
Average Total	0.10	607.83	0.07	610.83	0.05	610.10	0.05	623.63	0.04	618.00

Green cells indicate best average objective values.

Table D.2: Results from the parameter tuning of ALNS parameter *reward weight parameters* P. Here, P includes ρ_G, ρ_L and ρ_N, respectively. The tuning has been performed on three different instances of each test instance type. The ALNS heuristic has attempted to solve the problem five times for each value of P for each instance. Columns Gap (%) indicates the gap in objective value between the best known solution for the test instance and the average objective value over the five runs. Columns Time (s) represent the average runtime over the five runs to finding the best solution for an instance.

Test Instance	P = **(10,10,5)**		P = **(20,15,10)**		P = **(30,15,10)**		P = **(40,20,5)**	
	Gap (%)	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**
5-4-1-a	0.03	600.95	0.03	600.84	0.03	600.65	0.02	600.67
5-4-1-b	-0.00	381.22	-0.00	603.14	-0.00	280.80	-0.00	602.58
5-4-1-c	0.01	603.03	0.01	602.82	0.06	499.59	0.02	513.09
Average 5-4-1	0.01	528.40	0.01	602.26	0.03	460.35	0.01	572.11
5-6-1-a	0.02	604.42	0.01	601.55	0.01	602.53	0.02	603.14
5-6-1-b	0.00	601.56	0.00	604.87	0.00	601.02	0.00	602.69
5-6-1-c	0.00	603.35	0.01	550.33	0.00	602.50	0.02	602.89
Average 5-6-1	0.01	603.11	0.01	585.58	0.01	602.02	0.01	602.91
7-8-1-a	0.03	608.80	0.06	516.20	0.05	607.54	0.05	604.07
7-8-1-b	0.01	611.92	0.01	619.56	0.01	614.59	0.01	606.78
7-8-1-c	0.02	602.98	0.02	606.43	0.02	569.27	0.09	609.57
Average 7-8-1	0.02	607.90	0.03	580.73	0.03	597.14	0.05	606.81
10-9-2-a	0.02	623.38	0.05	608.92	0.03	577.42	0.03	616.00
10-9-2-b	0.05	607.98	0.05	609.03	0.04	605.73	0.03	614.77
10-9-2-c	0.06	617.45	0.06	616.25	0.06	621.93	0.08	607.47
Average 10-9-2	0.04	616.27	0.05	611.40	0.04	601.69	0.05	612.75
15-12-2-a	0.04	617.66	0.04	617.15	0.04	626.81	0.06	624.88
15-12-2-b	0.04	622.93	0.03	630.23	0.05	619.56	0.04	615.36
15-12-2-c	0.04	620.35	0.03	616.67	0.05	624.32	0.04	557.08
Average 15-12-2	0.04	620.32	0.03	621.35	0.05	623.56	0.05	599.10
20-15-2-a	0.07	626.83	0.08	627.31	0.06	621.70	0.08	627.58
20-15-2-b	0.05	615.06	0.06	617.40	0.06	620.37	0.06	620.20
20-15-2-c	0.05	612.42	0.08	609.62	0.13	611.92	0.06	608.88
Average 20-15-2	0.06	618.11	0.07	618.11	0.09	617.99	0.07	618.89
30-20-2-a	0.03	632.80	0.07	628.90	0.03	627.54	0.06	622.69
30-20-2-b	0.03	635.49	0.05	627.81	0.05	634.57	0.08	627.88
30-20-2-c	0.04	617.18	0.03	626.69	0.03	631.60	0.04	619.84
Average 30-20-2	0.03	628.49	0.05	627.80	0.04	631.24	0.06	623.47
40-25-2-a	0.12	624.88	0.07	612.65	0.11	629.87	0.12	619.05
40-25-2-b	0.04	636.43	0.07	624.59	0.07	636.10	0.05	641.81
40-25-2-c	0.14	637.84	0.09	614.72	0.14	631.23	0.13	638.80
Average 40-25-2	0.10	633.05	0.08	617.32	0.10	632.40	0.10	633.22
50-30-2-a	0.12	643.82	0.15	649.26	0.16	635.17	0.16	622.96
50-30-2-b	0.11	680.85	0.10	695.74	0.14	685.58	0.16	683.52
50-30-2-c	0.13	673.60	0.12	663.91	0.10	693.84	0.10	673.45
Average 50-30-2	0.12	666.09	0.12	669.63	0.13	671.53	0.14	659.97
Average Total	0.05	613.53	0.05	614.91	0.06	604.21	0.06	614.36

Green cells indicate best average objective values.

Table D.3: Results from the parameter tuning of ALNS parameter *reward decay parameter* α. The tuning has been performed on three different versions of each test instance type. The ALNS heuristic has attempted to solve the problem five times for each value of α for each instance. Columns Gap (%) indicate the gap in objective value between the best known solution for the relevant test instance and the average objective value over the five runs. Columns Time (s) represent the average runtime over the five runs to finding the best solution for an instance.

Test Instance	α = **0.4**		α = **0.5**		α = **0.6**		α = **0.7**	
	Gap (%)	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**
5-4-1-a	0.01	602.18	0.01	600.93	0.01	602.81	0.01	601.54
5-4-1-b	0.06	489.27	0.06	602.34	0.07	493.51	0.09	602.01
5-4-1-c	0.01	601.09	0.02	600.65	0.02	600.81	0.01	601.02
Average 5-4-1	0.02	564.18	0.03	601.31	0.03	565.71	0.04	601.52
5-6-1-a	0.00	302.50	0.02	486.20	0.01	489.75	0.00	500.92
5-6-1-b	0.04	602.11	0.04	601.10	0.04	602.19	0.04	601.99
5-6-1-c	0.01	601.82	0.01	602.69	0.02	602.89	0.02	602.65
Average 5-6-1	0.02	502.14	0.02	563.33	0.02	564.94	0.02	568.52
7-8-1-a	0.09	607.04	0.06	607.70	0.06	610.49	0.06	609.06
7-8-1-b	0.06	607.96	0.05	607.30	0.06	615.22	0.06	609.84
7-8-1-c	0.06	609.15	0.07	612.30	0.09	606.74	0.08	610.91
Average 7-8-1	0.07	608.05	0.06	609.10	0.07	610.82	0.07	609.93
10-9-2-a	0.04	620.40	0.03	564.24	0.05	614.68	0.06	613.51
10-9-2-b	0.05	608.16	0.07	615.02	0.08	617.92	0.06	604.71
10-9-2-c	0.05	609.08	0.04	614.16	0.05	611.60	0.04	610.09
Average 10-9-2	0.05	612.55	0.05	597.81	0.06	614.73	0.05	609.44
15-12-2-a	0.04	618.61	0.04	615.36	0.02	610.72	0.03	618.86
15-12-2-b	0.08	612.45	0.06	609.82	0.09	618.47	0.05	616.83
15-12-2-c	0.10	602.15	0.12	619.03	0.10	620.47	0.10	614.60
Average 15-12-2	0.07	611.07	0.07	614.73	0.07	616.56	0.06	616.76
20-15-2-a	0.06	629.23	0.06	611.61	0.05	622.32	0.08	625.69
20-15-2-b	0.07	621.11	0.03	620.51	0.04	613.27	0.06	613.95
20-15-2-c	0.07	624.34	0.09	611.81	0.07	619.44	0.07	613.71
Average 20-15-2	0.07	624.89	0.06	614.64	0.05	618.34	0.07	617.79
30-20-2-a	0.10	623.57	0.09	636.83	0.07	628.72	0.07	635.93
30-20-2-b	0.08	612.85	0.08	619.89	0.14	619.01	0.09	613.15
30-20-2-c	0.03	622.58	0.03	624.41	0.04	618.31	0.06	628.83
Average 30-20-2	0.07	619.66	0.07	627.05	0.08	622.01	0.07	625.97
40-25-2-a	0.08	629.21	0.13	640.72	0.10	637.65	0.11	621.13
40-25-2-b	0.10	644.93	0.07	621.34	0.10	638.56	0.11	621.22
40-25-2-c	0.04	627.01	0.08	626.02	0.07	621.32	0.03	624.37
Average 40-25-2	0.08	633.72	0.09	629.36	0.09	632.51	0.08	622.24
50-30-2-a	0.14	643.43	0.11	646.74	0.13	650.81	0.11	665.59
50-30-2-b	0.13	615.03	0.09	670.83	0.06	644.29	0.16	632.68
50-30-2-c	0.12	677.70	0.14	680.55	0.06	678.07	0.12	658.51
Average 50-30-2	0.13	645.39	0.11	666.04	0.08	657.72	0.13	652.26
Average Total	0.06	602.41	0.06	613.71	0.06	611.48	0.07	613.83

Green cells indicate best average objective values.

Table D.4: Results from the parameter tuning of ALNS parameter *neighborhood size parameter* η.The tuning has been performed on three different versions of each test instance type. The ALNS heuristic has attempted to solve the problem five times for each value of η for each instance.Here, Gap (%) indicates the gap in objective value between the best known solution for the test instance and the average objective value over the five runs. Columns Time (s) represent the average runtime over the five runs to finding the best solution for an instance.

Test Instance	η = **(0.05 - 0.15)**		η = **(0.05 - 0.3)**		η = **(0.15 - 0.3)**		η = **(0.15 - 0.5)**	
	Gap (%)	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**
5-4-1-a	0.03	600.87	0.00	507.60	0.01	420.69	0.01	523.77
5-4-1-b	0.03	277.19	0.03	153.50	0.02	224.54	0.02	227.22
5-4-1-c	0.00	600.50	0.00	600.49	0.00	601.46	0.00	601.83
Average 5-4-1	0.02	492.85	0.01	420.53	0.01	415.56	0.01	450.94
5-6-1-a	0.07	602.20	0.07	602.85	0.06	503.47	0.05	602.60
5-6-1-b	0.02	601.01	0.02	601.42	0.01	603.52	0.01	603.01
5-6-1-c	0.02	603.32	0.01	602.36	0.01	531.57	0.00	540.13
Average 5-6-1	0.04	602.18	0.03	602.21	0.03	546.19	0.02	581.91
7-8-1-a	0.04	606.62	0.02	618.30	0.06	610.55	0.03	616.95
7-8-1-b	0.02	603.75	0.01	608.07	0.01	610.65	0.01	617.69
7-8-1-c	0.02	604.47	0.03	610.38	0.01	611.40	0.02	606.14
Average 7-8-1	0.03	604.95	0.02	612.25	0.03	610.87	0.02	613.59
10-9-2-a	0.05	606.12	0.06	557.47	0.06	611.81	0.07	622.30
10-9-2-b	0.05	610.65	0.04	612.27	0.04	613.81	0.04	610.53
10-9-2-c	0.06	606.01	0.08	610.08	0.03	616.15	0.09	608.71
Average 10-9-2	0.05	607.59	0.06	593.27	0.04	613.92	0.07	613.85
15-12-2-a	0.05	610.30	0.06	615.20	0.05	613.45	0.04	621.25
15-12-2-b	0.07	609.25	0.04	617.76	0.04	613.01	0.05	616.24
15-12-2-c	0.01	616.74	0.03	618.87	0.02	615.64	0.05	624.59
Average 15-12-2	0.04	612.10	0.04	617.28	0.04	614.03	0.05	620.69
20-15-2-a	0.05	616.47	0.07	634.53	0.08	617.18	0.07	624.24
20-15-2-b	0.09	623.32	0.10	619.25	0.07	612.07	0.11	620.63
20-15-2-c	0.02	608.46	0.03	625.42	0.04	622.24	0.03	628.65
Average 20-15-2	0.05	616.08	0.07	626.40	0.06	617.16	0.07	624.50
30-20-2-a	0.04	614.53	0.04	626.84	0.05	618.03	0.08	625.50
30-20-2-b	0.03	624.98	0.02	616.42	0.05	627.76	0.07	632.32
30-20-2-c	0.03	622.16	0.02	631.65	0.03	627.21	0.06	615.55
Average 30-20-2	0.03	620.56	0.03	624.97	0.04	624.33	0.07	624.45
40-25-2-a	0.11	635.71	0.06	644.81	0.08	629.62	0.04	662.37
40-25-2-b	0.15	647.76	0.14	658.80	0.11	653.81	0.06	655.93
40-25-2-c	0.29	618.01	0.22	623.83	0.19	607.12	0.28	607.67
Average 40-25-2	0.18	633.83	0.14	642.48	0.12	630.18	0.13	641.99
50-30-2-a	0.26	637.61	0.15	686.51	0.11	637.44	0.13	638.67
50-30-2-b	0.12	628.12	0.09	627.54	0.09	624.72	0.09	624.22
50-30-2-c	0.19	700.14	0.17	621.03	0.08	652.42	0.08	671.86
Average 50-30-2	0.19	655.29	0.14	645.03	0.09	638.19	0.10	644.92
Average Total	0.07	605.05	0.06	598.27	0.05	590.05	0.06	601.87

Continued on next page

Green cells indicate best average objective values.

Test Instance	η = (0.15 - 0.7)		η = (0.3 - 0.5)		η = (0.3 - 0.7)		η = (0.5 - 0.7)	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
5-4-1-a	0.00	336.90	0.01	378.03	0.00	490.53	0.00	521.12
5-4-1-b	0.01	209.78	0.02	106.35	0.01	164.98	0.00	282.58
5-4-1-c	0.00	601.02	0.00	600.75	0.01	601.19	0.00	601.39
Average 5-4-1	0.00	382.57	0.01	361.71	0.01	418.90	0.00	468.36
5-6-1-a	0.09	605.47	0.08	265.16	0.03	606.01	0.07	490.65
5-6-1-b	0.01	607.42	0.01	604.28	0.01	603.42	0.00	604.61
5-6-1-c	0.00	514.14	0.02	602.64	0.00	603.58	0.01	533.76
Average 5-6-1	0.03	575.68	0.03	490.69	0.01	604.34	0.03	543.01
7-8-1-a	0.07	620.01	0.07	618.56	0.07	613.69	0.05	612.53
7-8-1-b	0.02	612.50	0.01	615.50	0.01	615.37	0.01	611.50
7-8-1-c	0.02	612.90	0.02	609.11	0.04	607.15	0.03	611.08
Average 7-8-1	0.04	615.14	0.03	614.39	0.04	612.07	0.03	611.71
10-9-2-a	0.08	610.94	0.07	611.26	0.08	613.61	0.08	615.89
10-9-2-b	0.04	524.06	0.05	614.97	0.06	609.95	0.04	612.42
10-9-2-c	0.04	614.78	0.02	613.88	0.03	612.68	0.07	618.79
Average 10-9-2	0.06	583.26	0.05	613.37	0.06	612.08	0.07	615.70
15-12-2-a	0.06	613.06	0.09	617.97	0.04	617.38	0.06	615.86
15-12-2-b	0.06	626.24	0.05	616.19	0.07	623.32	0.07	622.16
15-12-2-c	0.04	630.14	0.05	618.11	0.06	621.93	0.07	618.52
Average 15-12-2	0.05	623.15	0.06	617.42	0.06	620.87	0.07	618.85
20-15-2-a	0.10	625.12	0.08	621.33	0.10	623.77	0.12	622.37
20-15-2-b	0.09	615.70	0.11	638.53	0.11	617.02	0.14	617.09
20-15-2-c	0.07	633.60	0.06	616.85	0.05	619.87	0.07	625.16
Average 20-15-2	0.09	624.81	0.08	625.57	0.08	620.22	0.11	621.54
30-20-2-a	0.09	619.62	0.08	622.35	0.08	627.21	0.08	620.77
30-20-2-b	0.07	625.44	0.07	630.25	0.08	625.48	0.11	643.59
30-20-2-c	0.07	637.21	0.06	631.40	0.06	628.24	0.07	620.28
Average 30-20-2	0.07	627.42	0.07	628.00	0.07	626.98	0.09	628.21
40-25-2-a	0.10	630.71	0.10	659.30	0.11	617.15	0.09	653.54
40-25-2-b	0.09	639.18	0.12	632.75	0.11	646.94	0.12	668.70
40-25-2-c	0.19	610.31	0.11	638.86	0.20	624.74	0.13	637.98
Average 40-25-2	0.13	626.73	0.11	643.64	0.14	629.61	0.12	653.41
50-30-2-a	0.08	688.09	0.16	673.77	0.13	664.99	0.13	654.16
50-30-2-b	0.09	617.42	0.07	624.70	0.08	625.03	0.05	624.39
50-30-2-c	0.13	649.52	0.11	682.45	0.19	636.57	0.12	635.63
Average 50-30-2	0.10	651.67	0.11	660.31	0.13	642.20	0.10	638.06
Average Total	0.06	590.05	0.06	583.90	0.07	598.58	0.07	599.87

Green cells indicate best average objective values.

Table D.5: Results from the parameter tuning of ALNS parameter *relatedness weight parameters* Q. Here, Q includes q_o, q_d and q_f, respectively. The tuning has been performed on three different versions of each test instance type. The ALNS heuristic has attempted to solve the problem five times for each value of Q for each instance. Columns Gap (%) indicate the gap in objective value between the best known solution for the test instance and the average objective value over the five runs. Columns Time (s) represent the average runtime over the five runs to finding the best solution for an instance.

Test Instance	Q = **(0.05,0.05,0.1)**		Q = **(0.1,0.1,0.2)**		Q = **(0.15,0.15,0.3)**		Q = **(0.2,0.2,0.4)**	
	Gap (%)	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**
5-4-1-a	0.02	601.23	0.07	490.89	0.04	600.93	0.04	600.67
5-4-1-b	0.03	490.02	0.00	605.45	0.01	604.22	0.04	279.40
5-4-1-c	0.00	604.50	0.00	399.77	0.00	493.90	0.00	384.26
Average 5-4-1	0.02	565.25	0.02	498.70	0.02	566.35	0.03	421.44
5-6-1-a	0.00	492.31	0.00	601.13	0.00	603.80	0.00	608.39
5-6-1-b	0.07	601.72	0.11	602.15	0.11	603.68	0.06	603.87
5-6-1-c	0.01	525.60	0.01	605.96	0.03	498.59	0.02	565.40
Average 5-6-1	0.03	539.88	0.04	603.08	0.05	568.69	0.02	592.55
7-8-1-a	0.01	607.80	0.01	608.66	0.04	608.39	0.01	608.30
7-8-1-b	0.05	613.18	0.06	617.08	0.03	608.91	0.03	615.54
7-8-1-c	0.05	606.39	0.04	614.74	0.01	615.17	0.04	614.19
Average 7-8-1	0.04	609.12	0.04	613.49	0.03	610.82	0.03	612.68
10-9-2-a	0.04	616.69	0.03	622.70	0.03	615.85	0.03	611.06
10-9-2-b	0.04	618.17	0.03	611.15	0.04	610.80	0.03	614.23
10-9-2-c	0.01	610.79	0.03	617.95	0.03	612.52	0.02	615.29
Average 10-9-2	0.03	615.22	0.03	617.27	0.03	613.06	0.03	613.53
15-12-2-a	0.03	621.93	0.05	623.69	0.03	607.57	0.05	606.00
15-12-2-b	0.03	622.78	0.04	615.75	0.04	612.93	0.02	622.56
15-12-2-c	0.04	619.57	0.03	615.82	0.02	618.92	0.02	616.83
Average 15-12-2	0.03	621.43	0.04	618.42	0.03	613.14	0.03	615.13
20-15-2-a	0.03	618.61	0.06	627.17	0.08	619.31	0.06	621.50
20-15-2-b	0.07	623.35	0.06	619.97	0.07	635.86	0.06	630.79
20-15-2-c	0.09	614.06	0.13	613.33	0.18	608.64	0.11	622.39
Average 20-15-2	0.06	618.67	0.08	620.16	0.11	621.27	0.07	624.89
30-20-2-a	0.05	625.40	0.10	628.95	0.08	632.58	0.11	630.90
30-20-2-b	0.06	628.08	0.07	622.47	0.05	628.15	0.05	633.48
30-20-2-c	0.11	608.58	0.04	631.65	0.06	624.26	0.06	624.05
Average 30-20-2	0.07	620.69	0.07	627.69	0.07	628.33	0.07	629.48
40-25-2-a	0.03	624.54	0.06	626.85	0.05	646.84	0.09	637.76
40-25-2-b	0.04	645.28	0.05	632.10	0.07	637.40	0.03	629.94
40-25-2-c	0.05	634.32	0.05	634.25	0.06	649.96	0.05	641.55
Average 40-25-2	0.04	634.71	0.05	631.07	0.06	644.73	0.06	636.42
50-30-2-a	0.07	626.71	0.04	635.42	0.05	635.35	0.04	633.99
50-30-2-b	0.04	637.39	0.06	643.28	0.08	618.07	0.08	634.10
50-30-2-c	0.12	622.01	0.10	635.44	0.05	650.05	0.17	611.89
Average 50-30-2	0.08	628.70	0.07	638.05	0.06	634.49	0.09	626.66
Average Total	0.04	605.96	0.05	607.55	0.05	611.21	0.05	596.98

Green cells indicate best average objective values.

Table D.6: Results from the parameter tuning of ALNS parameter *worst determinism parameter* p_{worst}.The tuning has been performed on three different versions of each test instance type. The ALNS heuristic has attempted to solve the problem five times for each value of p_{worst} for each instance. Columns Gap (%) indicate the gap in objective value between the best known solution for the test instance and the average objective value over the five runs. Columns Time (s) represent the average runtime over the five runs to finding the best solution for an instance.

Test Instance	$p_{worst} = 1$		$p_{worst} = 2$		$p_{worst} = 4$		$p_{worst} = 6$	
	Gap (%)	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**
5-4-1-a	0.00	290.06	0.00	153.97	0.00	62.81	0.00	289.92
5-4-1-b	0.16	263.51	0.04	208.15	0.11	334.89	0.00	407.58
5-4-1-c	0.00	600.90	0.00	601.61	0.00	601.07	0.00	601.28
Average 5-4-1	0.05	384.82	0.01	321.24	0.04	332.92	0.00	432.93
5-6-1-a	0.00	602.98	0.00	601.71	0.00	603.24	0.00	601.21
5-6-1-b	0.01	539.72	0.00	603.57	0.01	602.98	0.02	602.72
5-6-1-c	0.00	600.89	0.00	601.44	0.00	601.40	0.00	603.46
Average 5-6-1	0.00	581.20	0.00	602.24	0.00	602.54	0.01	602.46
7-8-1-a	0.05	607.30	0.04	619.33	0.03	617.47	0.04	608.71
7-8-1-b	0.02	615.43	0.02	613.75	0.02	609.94	0.03	614.99
7-8-1-c	0.08	607.71	0.08	607.36	0.07	602.64	0.06	615.85
Average 7-8-1	0.05	610.15	0.05	613.48	0.04	610.02	0.04	613.18
10-9-2-a	0.05	616.01	0.06	614.85	0.03	612.08	0.02	611.20
10-9-2-b	0.05	617.54	0.03	610.23	0.02	615.39	0.05	612.96
10-9-2-c	0.03	613.54	0.03	618.64	0.03	615.46	0.03	613.48
Average 10-9-2	0.04	615.70	0.04	614.57	0.03	614.31	0.03	612.55
15-12-2-a	0.05	614.49	0.03	616.45	0.08	526.07	0.05	627.30
15-12-2-b	0.02	613.54	0.04	620.71	0.02	620.04	0.03	630.76
15-12-2-c	0.06	616.53	0.06	615.16	0.05	612.40	0.06	618.97
Average 15-12-2	0.05	614.85	0.04	617.44	0.05	586.17	0.05	625.68
20-15-2-a	0.07	625.51	0.08	614.19	0.05	624.59	0.05	615.15
20-15-2-b	0.08	614.35	0.06	616.89	0.05	618.04	0.06	632.62
20-15-2-c	0.06	614.84	0.05	617.88	0.08	631.34	0.06	612.47
Average 20-15-2	0.07	618.24	0.06	616.32	0.06	624.66	0.06	620.08
30-20-2-a	0.06	639.10	0.06	618.58	0.09	637.28	0.07	631.78
30-20-2-b	0.05	632.47	0.03	627.21	0.03	628.43	0.05	626.58
30-20-2-c	0.05	634.01	0.07	624.68	0.09	628.23	0.07	632.19
Average 30-20-2	0.05	635.20	0.05	623.49	0.07	631.31	0.06	630.18
40-25-2-a	0.09	652.38	0.08	636.30	0.07	639.87	0.13	645.12
40-25-2-b	0.10	631.15	0.10	634.52	0.09	629.07	0.10	637.85
40-25-2-c	0.06	608.12	0.07	623.07	0.05	620.01	0.08	626.41
Average 40-25-2	0.08	630.55	0.08	631.30	0.07	629.65	0.11	636.46
50-30-2-a	0.05	639.68	0.05	605.04	0.06	638.80	0.09	634.86
50-30-2-b	0.10	659.83	0.11	660.04	0.11	646.29	0.16	643.64
50-30-2-c	0.11	608.85	0.13	618.25	0.14	703.05	0.12	659.16
Average 50-30-2	0.09	636.12	0.10	627.78	0.10	662.71	0.12	645.89
Average Total	0.05	591.87	0.05	585.32	0.05	588.25	0.05	602.16

Green cells indicate best average objective values.

Table D.7: Results from the parameter tuning of ALNS parameter *related determinism parameter* $p_{related}$.The tuning has been performed on three different versions of each test instance type. The ALNS heuristic has attempted to solve the problem five times for each value of $p_{related}$ for each instance. Columns Gap (%) indicate the gap in objective value between the best known solution for the test instance and the average objective value over the five runs. Columns Time (s) represent the average runtime over the five runs to finding the best solution for an instance.

Test Instance	$p_{related} = \mathbf{1}$		$p_{related} = \mathbf{2}$		$p_{related} = \mathbf{4}$		$p_{related} = \mathbf{6}$	
	Gap (%)	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**
5-4-1-a	0.04	52.16	0.01	263.05	0.02	247.84	0.04	158.43
5-4-1-b	0.00	600.40	0.00	600.52	0.00	600.75	0.00	600.67
5-4-1-c	0.04	517.62	0.06	602.94	0.06	500.62	0.07	496.40
Average 5-4-1	0.03	390.06	0.03	488.84	0.03	449.74	0.03	418.50
5-6-1-a	0.06	537.97	0.03	603.06	0.09	370.08	0.07	601.46
5-6-1-b	0.05	601.67	0.03	603.22	0.05	602.77	0.04	604.72
5-6-1-c	0.03	603.17	0.05	512.61	0.04	602.41	0.03	602.71
Average 5-6-1	0.05	580.94	0.04	572.96	0.06	525.09	0.05	602.96
7-8-1-a	0.02	612.94	0.01	610.56	0.03	604.81	0.02	611.20
7-8-1-b	0.06	614.29	0.05	605.63	0.04	611.92	0.05	609.94
7-8-1-c	0.01	610.51	0.01	608.22	0.02	612.61	0.01	612.13
Average 7-8-1	0.03	612.58	0.02	608.14	0.03	609.78	0.03	611.09
10-9-2-a	0.04	613.39	0.03	614.36	0.04	616.63	0.03	616.79
10-9-2-b	0.04	614.82	0.07	616.29	0.04	611.78	0.06	619.39
10-9-2-c	0.03	611.37	0.02	615.07	0.03	610.01	0.02	611.86
Average 10-9-2	0.03	613.19	0.04	615.24	0.03	612.81	0.04	616.02
15-12-2-a	0.02	623.17	0.03	623.07	0.03	619.88	0.03	617.12
15-12-2-b	0.03	618.30	0.02	625.28	0.03	619.06	0.03	611.30
15-12-2-c	0.04	616.79	0.04	611.96	0.05	617.66	0.05	616.58
Average 15-12-2	0.03	619.42	0.03	620.10	0.04	618.87	0.03	615.00
20-15-2-a	0.06	622.80	0.06	618.74	0.07	623.08	0.04	611.94
20-15-2-b	0.10	630.40	0.07	629.51	0.15	613.08	0.11	627.43
20-15-2-c	0.04	629.10	0.03	614.84	0.05	621.67	0.07	627.95
Average 20-15-2	0.07	627.43	0.05	621.03	0.09	619.28	0.07	622.44
30-20-2-a	0.05	625.22	0.05	632.94	0.08	615.06	0.04	627.16
30-20-2-b	0.06	624.00	0.06	612.83	0.07	618.09	0.07	618.68
30-20-2-c	0.06	628.62	0.09	623.15	0.09	620.69	0.04	628.41
Average 30-20-2	0.05	625.95	0.07	622.97	0.08	617.95	0.05	624.75
40-25-2-a	0.03	612.38	0.04	620.55	0.03	618.60	0.03	630.19
40-25-2-b	0.11	627.88	0.07	643.18	0.09	636.58	0.08	639.89
40-25-2-c	0.10	641.43	0.08	623.55	0.09	618.23	0.08	630.83
Average 40-25-2	0.08	627.23	0.06	629.09	0.07	624.47	0.07	633.63
50-30-2-a	0.14	630.73	0.13	647.92	0.12	654.94	0.09	617.04
50-30-2-b	0.08	636.31	0.09	622.39	0.07	634.27	0.11	640.12
50-30-2-c	0.07	626.82	0.06	636.57	0.08	637.61	0.07	630.82
Average 50-30-2	0.10	631.29	0.09	635.63	0.09	642.27	0.09	629.32
Average Total	0.05	592.01	0.05	601.56	0.06	591.14	0.05	597.08

Green cells indicate best average objective values.

Table D.8: Results from the parameter tuning of ALNS parameter *greedy determinism parameter* p_{greedy}.The tuning has been performed on three different versions of each test instance type. The ALNS heuristic has attempted to solve the problem five times for each value of p_{greedy} for each instance. Columns Gap (%) indicate the gap in objective value between the best known solution for the test instance and the average objective value over the five runs. Columns Time (s) represent the average runtime over the five runs to finding the best solution for an instance.

Test Instance	$p_{greedy} = 1$		$p_{greedy} = 2$		$p_{greedy} = 4$		$p_{greedy} = 6$	
	Gap (%)	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**	**Gap (%)**	**Time (s)**
5-4-1-a	-0.00	35.03	-0.00	91.77	0.03	43.36	-0.00	160.34
5-4-1-b	0.00	601.37	0.00	601.15	0.00	600.95	0.00	600.76
5-4-1-c	0.03	74.29	0.01	96.88	0.03	64.15	0.00	79.02
Average 5-4-1	0.01	236.90	0.00	263.27	0.02	236.15	0.00	280.04
5-6-1-a	0.00	602.03	0.00	602.05	0.01	602.53	0.00	602.30
5-6-1-b	0.01	601.40	0.04	603.07	0.01	601.37	0.05	601.25
5-6-1-c	0.01	537.22	0.00	605.44	0.01	605.08	0.01	607.76
Average 5-6-1	0.01	580.22	0.01	603.52	0.01	602.99	0.02	603.77
7-8-1-a	0.01	607.30	0.01	610.43	0.01	601.96	0.00	605.73
7-8-1-b	0.00	612.13	0.01	609.90	0.03	608.89	0.02	612.42
7-8-1-c	0.01	608.29	0.02	615.42	0.01	615.00	0.02	613.37
Average 7-8-1	0.01	609.24	0.01	611.91	0.02	608.62	0.01	610.51
10-9-2-a	0.02	611.68	0.04	604.66	0.03	607.53	0.03	607.55
10-9-2-b	0.05	613.95	0.04	615.18	0.02	620.27	0.02	613.99
10-9-2-c	0.07	614.12	0.06	607.84	0.06	613.43	0.08	613.20
Average 10-9-2	0.05	613.25	0.04	609.23	0.04	613.74	0.04	611.58
15-12-2-a	0.06	569.58	0.06	615.61	0.03	624.73	0.10	622.61
15-12-2-b	0.07	616.19	0.06	622.57	0.04	617.36	0.03	618.15
15-12-2-c	0.05	620.11	0.05	617.45	0.06	629.24	0.04	607.70
Average 15-12-2	0.06	601.96	0.05	618.54	0.04	623.78	0.06	616.15
20-15-2-a	0.03	617.51	0.05	630.48	0.04	623.67	0.03	632.27
20-15-2-b	0.09	623.91	0.07	618.77	0.07	619.83	0.08	620.20
20-15-2-c	0.06	621.41	0.07	618.40	0.06	615.20	0.08	634.15
Average 20-15-2	0.06	620.94	0.06	622.55	0.06	619.57	0.06	628.87
30-20-2-a	0.09	604.48	0.15	611.60	0.09	614.30	0.10	606.54
30-20-2-b	0.10	624.77	0.08	626.21	0.06	623.71	0.05	634.39
30-20-2-c	0.11	618.96	0.10	612.87	0.04	631.31	0.06	635.00
Average 30-20-2	0.10	616.07	0.11	616.89	0.06	623.11	0.07	625.31
40-25-2-a	0.11	631.79	0.08	635.98	0.07	627.05	0.07	639.35
40-25-2-b	0.15	624.76	0.08	626.99	0.09	615.87	0.09	624.08
40-25-2-c	0.07	626.68	0.05	631.98	0.03	621.53	0.03	617.90
Average 40-25-2	0.11	627.74	0.07	631.65	0.06	621.48	0.06	627.11
50-30-2-a	0.14	656.57	0.11	636.92	0.07	628.02	0.08	639.52
50-30-2-b	0.06	629.29	0.07	626.66	0.06	634.62	0.05	609.80
50-30-2-c	0.14	662.38	0.18	681.81	0.19	652.71	0.20	694.54
Average 50-30-2	0.11	649.41	0.12	648.46	0.11	638.45	0.11	647.96
Average Total	0.06	572.86	0.05	580.67	0.05	576.43	0.05	583.48

Green cells indicate best average objective values.

Table D.9: Results from the parameter tuning of ALNS parameter *reheat strategy* $\tau_{strategy}$.The tuning has been performed on three different versions of each test instance type. The ALNS heuristic has attempted to solve the problem five times for each value of $\tau_{strategy}$ for each instance.Here, Gap (%) indicates the gap in objective value between the best known solution for the test instance and the average objective value over the five runs. Columns Time (s) represent the average runtime over the five runs to finding the best solution for an instance.

Test Instance	$\tau_{strategy}$ = **no-reheat**		$\tau_{strategy}$ = **reheat**	
	Gap (%)	**Time (s)**	**Gap (%)**	**Time (s)**
5-4-1-a	0.00	600.60	0.00	600.69
5-4-1-b	0.00	178.81	0.00	601.15
5-4-1-c	0.02	143.37	0.00	600.86
Average 5-4-1	0.01	307.59	0.00	600.90
5-6-1-a	0.01	607.23	0.04	603.50
5-6-1-b	0.00	603.12	0.01	601.62
5-6-1-c	0.00	103.64	0.00	602.93
Average 5-6-1	0.00	438.00	0.02	602.68
7-8-1-a	0.08	611.52	0.08	560.71
7-8-1-b	0.03	607.07	0.06	610.57
7-8-1-c	0.00	609.75	0.01	607.19
Average 7-8-1	0.04	609.44	0.05	592.83
10-9-2-a	0.04	616.42	0.08	613.96
10-9-2-b	0.03	610.30	0.06	615.15
10-9-2-c	0.05	279.83	0.05	615.23
Average 10-9-2	0.04	502.18	0.06	614.78
15-12-2-a	0.07	625.40	0.06	624.89
15-12-2-b	0.01	617.54	0.03	613.03
15-12-2-c	0.06	622.44	0.08	620.75
Average 15-12-2	0.05	621.80	0.06	619.56
20-15-2-a	0.05	615.12	0.02	610.64
20-15-2-b	0.06	621.89	0.04	617.40
20-15-2-c	0.03	625.66	0.05	623.80
Average 20-15-2	0.05	620.89	0.04	617.28
30-20-2-a	0.07	617.45	0.05	626.92
30-20-2-b	0.05	625.78	0.02	637.06
30-20-2-c	0.05	622.04	0.07	611.96
Average 30-20-2	0.06	621.76	0.05	625.31
40-25-2-a	0.06	642.56	0.04	625.88
40-25-2-b	0.08	624.11	0.05	631.13
40-25-2-c	0.05	638.55	0.11	635.36
Average 40-25-2	0.06	635.07	0.07	630.79
50-30-2-a	0.11	639.74	0.08	626.45
50-30-2-b	0.07	672.15	0.03	638.40
50-30-2-c	0.06	661.07	0.09	656.56
Average 50-30-2	0.08	657.65	0.07	640.47
Average Total	0.04	557.15	0.05	616.07

Green cells indicate best average objective values.

Appendix E

Complete Computational Study

This appendix provides extensive results from the analyses conducted in the computational study. These results form the basis for the average values presented in the computational study.

Table E.1: Evaluation of how the parameter tuning process affects the performance of the ALNS heuristic. Three versions of each test instance type are generated. Each test instance is run five times with a time limit of 600 seconds. The average objective values over the five runs are displayed in the columns Objective Value. The average runtimes the tuned and untuned ALNS heuristic spent finding the best known solution over the five runs are displayed in the columns Time (s).

Test Instance	**Before Tuning Process**		**After Tuning Process**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1-a	270.78	52.15	270.78	83.50
5-4-1-b	225.43	28.43	225.43	15.61
5-4-1-c	241.63	215.41	243.04	119.88
Average 5-4-1	245.95	98.66	246.42	73.00
6-6-1-a	339.00	271.59	337.67	111.01
6-6-1-b	413.37	137.21	413.13	80.44
6-6-1-c	369.56	300.39	359.21	179.56
Average 6-6-1	373.97	236.40	370.00	123.67
8-8-1-a	562.88	322.52	562.25	215.23
8-8-1-b	491.99	339.92	498.04	318.83
8-8-1-c	485.38	229.97	486.03	298.72
Average 8-8-1	513.42	297.47	515.44	277.59
10-9-2-a	512.11	212.13	523.97	302.29
10-9-2-b	548.69	453.72	555.01	374.27
10-9-2-c	499.00	420.21	524.86	373.04
Average 10-9-2	519.93	362.02	534.61	349.86
15-12-2-a	723.83	266.30	734.77	230.99
15-12-2-b	763.21	373.08	814.30	373.33
15-12-2-c	659.01	345.19	700.51	295.41
Average 15-12-2	715.35	328.19	749.86	299.91
20-15-2-a	815.42	208.49	839.65	350.58
20-15-2-b	875.84	260.82	904.71	368.86
20-15-2-c	886.66	267.47	912.11	350.75
Average 20-15-2	859.31	245.59	885.49	356.73
30-20-2-a	1373.66	273.26	1393.38	245.51
30-20-2-b	1308.50	239.50	1337.49	398.00
30-20-2-c	1315.93	293.38	1327.28	283.55
Average 30-20-2	1332.70	268.71	1352.71	309.02
40-25-2-a	1714.83	242.65	1697.89	174.05
40-25-2-b	1726.98	250.54	1796.60	264.10
40-25-2-c	1579.70	374.16	1601.59	163.91
Average 40-25-2	1673.84	289.12	1698.69	200.69
50-30-2-a	1763.04	439.68	1797.62	490.37
50-30-2-b	1850.83	416.70	1914.23	482.14
50-30-2-c	1786.25	381.23	1918.13	496.67
Average 50-30-2	1800.04	412.53	1876.66	489.73
Average Total	892.73	282.08	914.43	275.58

Green cells indicate best average objective values

Table E.2: Evaluation of the effects of having adaptive selection of destroy and repair operators in the ALNS heuristic as opposed to random selection. Three versions of each instance type are generated. Each version is solved ten times for 600 seconds using both adaptive and random selection of operators. The average objective values over the ten runs are displayed in the columns Objective Value. The average runtimes the LNS and the ALNS heuristic spent finding the best known solution over the ten runs are displayed in the columns Time (s).

Test Instance	**LNS (random)**		**ALNS (adaptive)**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1-a	245.66	172.69	234.69	1.05
5-4-1-b	264.80	1.13	266.58	1.28
5-4-1-c	239.34	182.47	237.62	39.81
Average 5-4-1	249.93	118.76	246.30	14.04
6-6-1-a	359.78	5.45	358.95	6.20
6-6-1-b	384.97	24.88	382.49	5.22
6-6-1-c	372.09	8.79	369.55	63.07
Average 6-6-1	372.28	13.04	370.33	24.83
8-8-1-a	576.25	29.97	576.79	20.95
8-8-1-b	389.76	516.45	515.75	313.20
8-8-1-c	444.11	5.96	450.21	16.89
Average 8-8-1	470.04	184.13	514.25	117.01
Average 10-9-2	530.89	296.96	536.32	338.94
10-9-2-a	546.27	305.33	552.89	335.33
10-9-2-b	538.07	275.02	547.86	431.82
10-9-2-c	508.34	310.52	508.22	249.67
Average 10-9-2	530.89	296.96	536.32	338.94
15-12-2-a	800.29	345.39	807.43	392.57
15-12-2-b	698.22	198.28	697.09	347.26
15-12-2-c	737.51	253.39	746.72	383.08
Average 15-12-2	745.34	265.69	750.41	374.30
20-15-2-a	905.40	372.80	919.36	399.79
20-15-2-b	939.60	337.27	952.71	395.41
20-15-2-c	868.52	202.11	883.85	239.41
Average 20-15-2	904.51	304.06	918.64	344.87
30-20-2-a	1385.08	404.96	1418.67	444.82
30-20-2-b	1397.59	164.48	1397.10	294.12
30-20-2-c	1372.96	98.83	1372.15	99.58
Average 30-20-2	1385.21	222.76	1395.98	279.51
40-25-2-a	1706.01	78.33	1700.11	337.94
40-25-2-b	1800.71	167.71	1769.87	92.95
40-25-2-c	1735.09	339.62	1739.65	336.21
Average 40-25-2	1747.27	195.22	1736.54	255.70
50-30-2-a	2001.83	133.48	2059.70	383.94
50-30-2-b	2061.44	361.67	2110.66	300.66
50-30-2-c	2082.56	180.14	2116.35	354.66
Average 50-30-2	2048.61	225.10	2095.57	346.42
Average Total	939.34	202.86	951.59	232.85

Green cells indicate best average objective values

Table E.3: Evaluation of how the Local Search component for pricing solutions (LSP) affect the performance of the ALNS heuristic. Three versions of each test instance type are generated. Each test instance is run ten times with a time limit of 600 seconds. The average objective values over the ten runs are displayed in the columns Objective Value. The average runtimes the ALNS heuristic spent finding the best known solution over the ten runs are displayed in the columns Time (s).

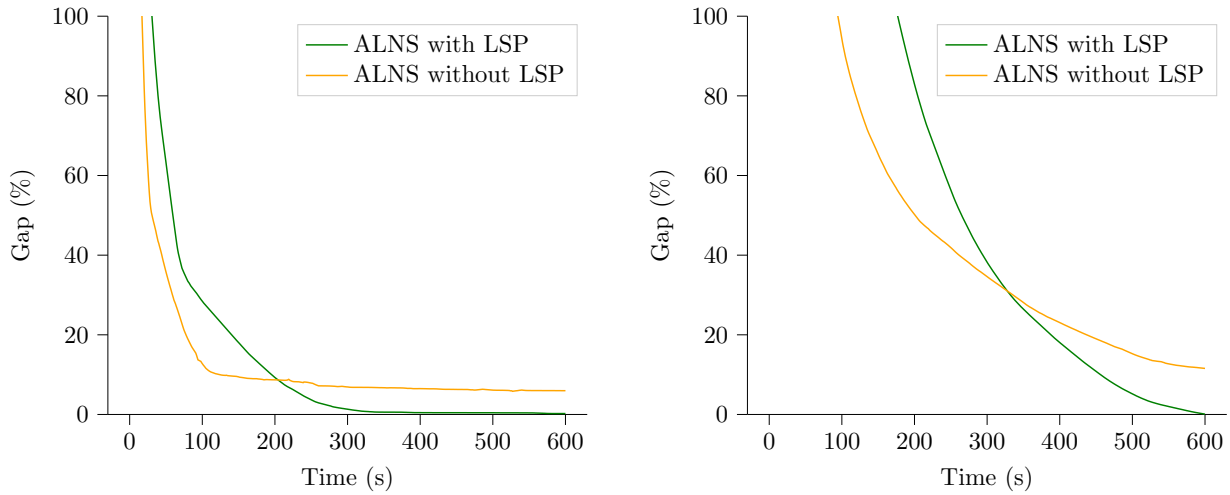
Test Instance	**ALNS without LSP**		**ALNS with LSP**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1-a	210.48	196.37	215.98	140.09
5-4-1-b	234.33	299.89	241.52	89.62
5-4-1-c	221.82	72.26	234.05	84.82
Average 5-4-1	222.21	189.51	230.52	104.84
6-6-1-a	314.02	391.50	332.76	159.03
6-6-1-b	333.24	312.27	361.78	114.65
6-6-1-c	326.39	225.49	373.83	345.10
Average 6-6-1	324.55	309.75	356.12	206.26
8-8-1-a	507.11	298.68	570.61	321.58
8-8-1-b	491.76	356.41	553.07	282.99
8-8-1-c	508.22	236.77	558.62	196.08
Average 8-8-1	502.36	297.29	560.77	266.89
10-9-2-a	461.32	291.85	531.89	451.43
10-9-2-b	459.01	245.20	533.97	378.36
10-9-2-c	407.58	274.07	478.36	332.50
Average 10-9-2	442.63	270.37	514.74	387.43
15-12-2-a	647.96	376.12	796.03	412.16
15-12-2-b	622.27	367.80	738.91	399.54
15-12-2-c	645.02	317.26	759.80	464.76
Average 15-12-2	638.42	353.73	764.91	425.49
20-15-2-a	862.14	425.17	932.96	456.35
20-15-2-b	717.32	403.83	883.05	415.24
20-15-2-c	793.97	400.19	886.72	323.21
Average 20-15-2	791.14	409.73	900.91	398.27
30-20-2-a	1111.81	354.90	1211.50	522.95
30-20-2-b	1236.09	412.66	1259.62	383.02
30-20-2-c	1395.21	347.73	1453.87	477.83
Average 30-20-2	1247.70	371.76	1308.33	461.27
40-25-2-a	1643.21	393.83	1660.52	526.19
40-25-2-b	1419.29	327.52	1467.46	460.96
40-25-2-c	1553.78	416.14	1605.27	514.42
Average 40-25-2	1538.76	379.16	1577.75	500.52
50-30-2-a	1869.35	520.05	1836.35	523.76
50-30-2-b	1869.34	474.53	1806.59	468.57
50-30-2-c	1717.31	481.44	1746.89	491.87
Average 50-30-2	1818.67	492.01	1796.61	494.73
Average Total	836.27	341.48	890.07	360.63

Green cells indicate best average objective values.

Table E.4: Evaluation of how the Local Search component for relocation solutions (LSR) affect the performance of the ALNS heuristic. Three versions of each test instance type are generated. Each test instance is run ten times with a time limit of 600 seconds. The average objective values over the ten runs are displayed in the columns Objective Value. The average runtimes the ALNS heuristic spent finding the best known solution over the ten runs are displayed in the columns Time (s).

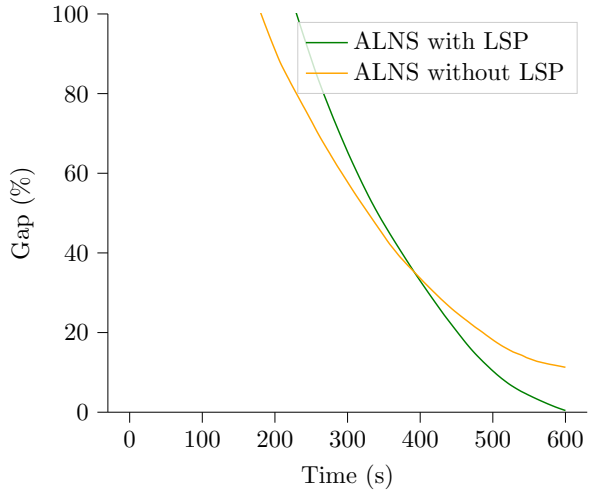
Test Instance	**ALNS without LSR**		**ALNS with LSR**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1-a	192.44	0.81	213.40	184.69
5-4-1-b	227.00	1.17	262.76	64.57
5-4-1-c	212.64	3.25	229.42	162.50
Average 5-4-1	210.69	1.74	235.20	137.26
6-6-1-a	253.48	111.36	345.70	327.33
6-6-1-b	308.12	313.20	369.91	362.60
6-6-1-c	384.56	234.16	429.92	119.45
Average 6-6-1	315.39	219.57	381.84	269.80
8-8-1-a	415.40	14.47	521.14	261.07
8-8-1-b	433.62	21.10	467.46	249.16
8-8-1-c	443.97	94.46	522.34	252.37
Average 8-8-1	430.99	43.34	503.65	254.20
10-9-2-a	560.28	247.47	588.66	416.33
10-9-2-b	335.82	29.42	519.60	391.86
10-9-2-c	525.27	312.16	550.43	340.34
Average 10-9-2	473.79	196.35	552.90	382.84
15-12-2-a	724.99	413.79	742.58	500.07
15-12-2-b	666.59	329.44	724.29	365.38
15-12-2-c	736.95	290.59	795.12	409.93
Average 15-12-2	709.51	344.61	754.00	425.13
20-15-2-a	953.01	440.46	936.82	438.11
20-15-2-b	809.73	340.21	819.32	413.82
20-15-2-c	807.32	387.75	858.19	414.65
Average 20-15-2	856.69	389.47	871.44	422.19
30-20-2-a	1271.87	387.40	1297.49	463.21
30-20-2-b	1228.89	311.99	1288.75	435.45
30-20-2-c	1172.31	419.05	1222.20	551.43
Average 30-20-2	1224.36	372.82	1269.48	483.37
40-25-2-a	1463.99	281.21	1498.72	453.93
40-25-2-b	1436.06	382.57	1550.39	455.07
40-25-2-c	1678.09	339.98	1695.01	471.74
Average 40-25-2	1526.05	334.59	1581.37	460.25
50-30-2-a	1832.03	312.62	1846.82	444.11
50-30-2-b	1693.50	452.59	1727.68	536.80
50-30-2-c	1683.74	417.64	1732.19	380.72
Average 50-30-2	1736.42	394.29	1768.89	453.88
Average Total	831.54	255.20	879.86	365.43

Green cells indicate best average objective values.

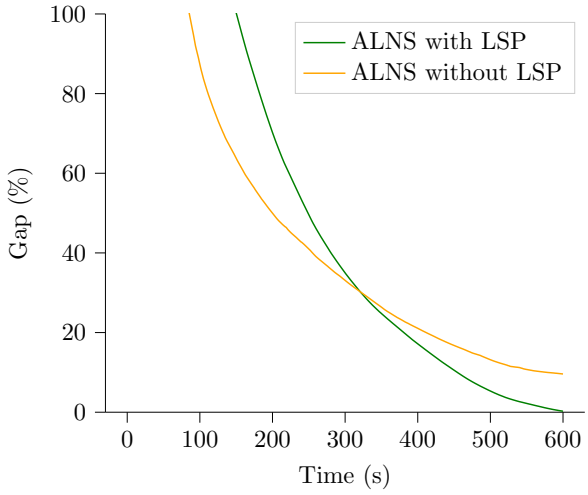


(a) Small Test Instances

(b) Medium Test Instances

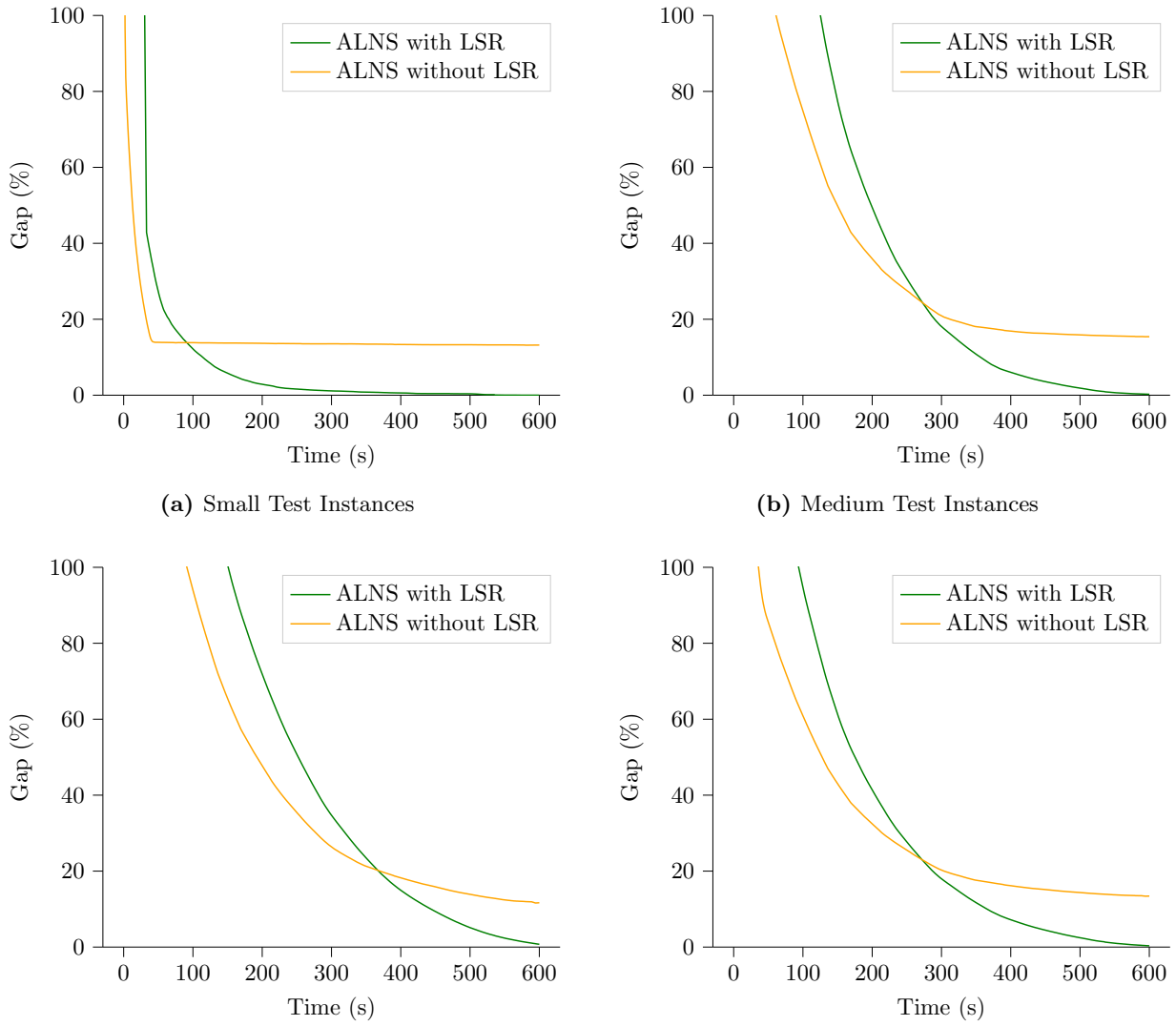


(c) Large Test Instances



(d) All Test Instances

Figure E.1: Runtime analysis for the ALNS heuristic with and without the LSP component. The analysis is conducted using linear regression based on the times of discovery of improving solutions for test instances of small, medium and large sizes, and on average for all instances. The data is collected from the same analysis presented in Table E.3. The graphs display the average gap in percentage to the overall best solution found at any time of the run for the relevant test instances.



(a) Small Test Instances

(b) Medium Test Instances

(c) Large Test Instances

(d) All Test Instances

Figure E.2: Runtime analysis for the ALNS heuristic with and without the LSR component. The analysis is conducted using linear regression based on the times of discovery of improving solutions for test instances of small, medium and large sizes, and on average for all instances. The data is collected from the same analysis presented in Table E.4.The graphs display the average gap in percentage to the overall best solution found at any time of the run for the relevant test instances.

Table E.5: Evaluation of how the ALNS heuristic improves the solution found by the construction heuristic. Three versions of each test instance type are generated. Each test instance is run ten times with a time limit of 600 seconds. The average objective values over the ten runs are displayed in the columns Objective Value. The average runtimes finding the best known solutions over the ten runs are displayed in the columns Time (s).

Test Instance	**Construction Heuristic**		**ALNS Heuristic**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1-a	45.31	0.01	247.80	340.71
5-4-1-b	23.49	0.01	243.28	171.20
5-4-1-c	-3.55	0.01	263.04	257.29
Average 5-4-1	21.75	0.01	251.37	256.40
6-6-1-a	99.04	0.01	343.35	181.32
6-6-1-b	24.98	0.01	356.12	228.28
6-6-1-c	88.74	0.01	409.49	307.24
Average 6-6-1	70.92	0.01	369.65	238.95
8-8-1-a	123.25	0.01	475.76	232.11
8-8-1-b	97.97	0.02	482.04	318.37
8-8-1-c	162.35	0.01	494.97	319.13
Average 8-8-1	127.86	0.01	484.25	289.87
10-9-2-a	24.47	0.02	544.87	362.15
10-9-2-b	51.99	0.02	495.24	337.99
10-9-2-c	178.40	0.02	539.76	422.70
Average 10-9-2	84.95	0.02	526.62	374.28
15-12-2-a	84.51	0.03	725.08	467.99
15-12-2-b	117.91	0.02	688.59	383.97
15-12-2-c	194.14	0.02	744.34	403.64
Average 15-12-2	132.19	0.02	719.34	418.53
20-15-2-a	166.42	0.03	922.91	462.05
20-15-2-b	224.92	0.03	908.58	370.31
20-15-2-c	96.29	0.03	793.63	334.72
Average 20-15-2	162.54	0.03	875.04	389.03
30-20-2-a	231.90	0.04	1276.03	483.97
30-20-2-b	332.34	0.04	1263.39	500.81
30-20-2-c	360.79	0.04	1268.46	504.52
Average 30-20-2	308.34	0.04	1269.29	496.43
40-25-2-a	742.63	0.05	1599.06	404.40
40-25-2-b	790.96	0.05	1690.16	439.92
40-25-2-c	821.12	0.05	1725.10	524.28
Average 40-25-2	784.90	0.05	1671.44	456.20
50-30-2-a	794.34	0.07	1756.08	478.95
50-30-2-b	793.08	0.19	1850.31	503.47
50-30-2-c	691.28	0.08	1782.01	378.75
Average 50-30-2	759.57	0.11	1796.13	453.72
Average Total	272.56	0.03	884.79	374.82

Green cells indicate best average objective values.

Table E.6: Evaluation of how the time limit affects the performance of the ALNS heuristic. Three versions of each test instance type are generated. Each test instance is run ten times for 600 seconds (ten minutes), 3600 seconds (one hour) and 10800 seconds (three hours). The average objective values over the ten runs are displayed in the columns Objective Value. The average runtimes the ALNS heuristic spent finding the best known solution over the ten runs are displayed in the columns Time (s).

Test Instance	**ALNS heuristic (600 s)**		**ALNS heuristic (3600 s)**		**ALNS heuristic (10800 s)**	
	Objective Value	Time (s)	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1-a	264.18	4.36	264.18	4.36	264.18	4.36
5-4-1-b	268.93	173.94	268.93	173.94	268.93	173.94
5-4-1-c	264.35	1.46	264.35	1.46	264.35	1.46
Average 5-4-1	265.82	59.92	265.82	59.92	265.82	59.92
6-6-1-a	366.28	70.33	366.28	70.33	366.28	70.33
6-6-1-b	334.43	13.54	334.43	13.54	334.43	13.54
6-6-1-c	395.07	31.00	395.07	31.00	395.07	31.00
Average 6-6-1	365.26	38.29	365.26	38.29	365.26	38.29
8-8-1-a	536.92	600.00	549.39	3600.00	557.66	5382.02
8-8-1-b	553.01	600.00	556.05	3121.54	560.30	3121.54
8-8-1-c	530.12	600.00	538.52	2303.02	538.52	2303.02
Average 8-8-1	540.02	600.00	547.99	3008.19	552.16	3602.20
10-9-2-a	571.73	600.00	578.40	3600.00	578.40	7512.70
10-9-2-b	525.53	600.00	533.17	3600.00	539.21	7478.69
10-9-2-c	559.22	600.00	561.20	3600.00	569.17	4664.90
Average 10-9-2	552.16	600.00	557.59	3600.00	562.26	6552.10
15-12-2-a	748.97	600.00	766.47	3424.99	766.47	3424.99
15-12-2-b	740.90	600.00	775.58	3600.00	775.58	6823.74
15-12-2-c	770.22	600.00	792.34	3600.00	796.95	6312.04
Average 15-12-2	753.36	600.00	778.13	3541.66	779.67	5520.26
20-15-2-a	900.40	600.00	939.29	3600.00	946.31	4397.95
20-15-2-b	853.55	600.00	888.33	3600.00	910.61	7738.18
20-15-2-c	815.22	600.00	870.21	3600.00	880.80	6879.35
Average 20-15-2	857.81	600.00	900.28	3600.00	913.67	6319.84
30-20-2-a	1260.69	600.00	1351.50	3600.00	1369.98	6340.12
30-20-2-b	1211.54	600.00	1286.02	3600.00	1307.73	6641.05
30-20-2-c	1286.52	600.00	1368.91	3600.00	1402.38	5298.31
Average 30-20-2	1252.92	600.00	1335.48	3600.00	1360.03	6093.16
40-25-2-a	1487.79	600.00	1612.02	3600.00	1646.90	5877.46
40-25-2-b	1715.72	600.00	1835.79	3600.00	1885.47	7192.97
40-25-2-c	1655.42	600.00	1703.85	3600.00	1718.87	4103.06
Average 40-25-2	1619.64	600.00	1717.22	3600.00	1750.41	5724.50
50-30-2-a	1798.77	600.00	1953.77	3600.00	2005.34	5336.72
50-30-2-b	1848.23	600.00	1980.75	3600.00	1980.75	3764.86
50-30-2-c	1919.47	600.00	2086.40	3600.00	2112.57	4032.90
Average 50-30-2	1853.28	600.00	2004.23	3600.00	2031.13	4390.07
Average Total	895.03	477.58	939.34	2738.67	953.38	4255.59

Green cells indicate best average objective values.

Table E.7: Comparison of the ALNS heuristic and the commercial solver Gurobi. Three versions of each test instance type are generated. Each test instance is run ten times with the ALNS heuristic for a maximum of 600 seconds. The results from Gurobi after 600 seconds are registered as a benchmark. The average objective values are displayed in the columns Objective Value. The average runtimes for finding the best known solutions are displayed in the columns Time (s). The average gaps to the best upper bound found by Gurobi after a maximum of 600 seconds are displayed in columns Gap (%).

Test Instance Type	**Gurobi (600 s)**			**ALNS Heuristic (600 s)**		
	Objective Value	Gap (%)	Time (s)	Objective Value	Gap (%)	Time (s)
5-4-1-a	264.18	0.00	16.10	264.18	0.00	4.36
5-4-1-b	229.74	0.00	11.63	229.74	0.00	4.54
5-4-1-c	264.35	0.00	11.61	264.35	0.00	1.46
Average 5-4-1	252.76	0.00	13.11	252.76	0.00	3.45
6-6-1-a	426.39	0.00	40.52	426.39	0.00	8.88
6-6-1-b	384.37	0.00	30.71	384.37	0.00	4.85
6-6-1-c	423.20	0.00	26.05	423.20	-0.00	11.20
Average 6-6-1	411.32	0.00	32.43	411.32	0.00	8.31
8-8-1-a	560.52	0.00	166.46	560.52	0.00	116.84
8-8-1-b	557.87	0.00	133.49	557.87	0.00	54.78
8-8-1-c	594.03	0.00	98.36	594.03	0.00	31.86
Average 8-8-1	570.81	0.00	132.77	570.81	0.00	67.82
10-9-2-a	595.08	0.00	600.00	593.00	0.00	168.68
10-9-2-b	556.79	0.05	600.00	574.46	0.02	88.07
10-9-2-c	588.07	0.05	600.00	604.04	0.02	144.74
Average 10-9-2	579.98	3.28	600.00	590.50	1.40	133.83
15-12-2-a	817.35	0.16	600.00	848.15	0.12	269.57
15-12-2-b	791.47	0.15	600.00	780.08	0.17	205.27
15-12-2-c	849.72	0.08	600.00	824.83	0.11	349.57
Average 15-12-2	819.51	13.06	600.00	817.68	13.30	274.80
20-15-2-a	952.38	0.19	600.00	956.12	0.18	304.57
20-15-2-b	601.51	1.01	600.00	1023.80	0.18	397.55
20-15-2-c	946.58	0.14	600.00	924.34	0.16	252.26
Average 20-15-2	833.49	44.56	600.00	968.09	17.62	318.13
30-20-2-a	1544.21	0.25	600.00	1616.37	0.19	471.83
30-20-2-b	1206.29	0.45	600.00	1466.58	0.19	294.84
30-20-2-c	1611.83	0.13	600.00	1618.77	0.13	362.68
Average 30-20-2	1454.11	27.65	600.00	1567.24	17.06	376.45
40-25-2-a	965.52	1.33	600.00	1878.89	0.20	691.47
40-25-2-b	1276.03	0.49	600.00	1516.40	0.26	97.31
40-25-2-c	1552.51	0.55	600.00	1963.30	0.22	490.56
Average 40-25-2	1264.69	79.10	600.00	1786.20	22.65	426.45
50-30-2-a	0.10	(-)	600.00	2175.49	0.30	311.19
50-30-2-b	0.10	(-)	600.00	1949.87	0.41	457.25
50-30-2-c	0.10	(-)	600.00	1987.55	0.40	343.88
Average 50-30-2	0.10	(-)	600.00	2037.64	37.00	370.77
Average Total	687.41	18.63	419.81	1001.47	11.91	186.30

Green cells indicate best average objective values.
(-) in *Objective Value* indicates no feasible solution found.
(-) in *Gap (%)* indicates no upper bound found.
Average values are found replacing (-) with 0.

Table E.8: Comparison of the ALNS heuristic and the commercial solver Gurobi. Three versions of each test instance type are generated. Each test instance is run ten times witn the ALNS heuristic for a maximum of 3600 seconds. The results from Gurobi after 3600 seconds are registered as a benchmark. The average objective values are displayed in the columns Objective Value. The average runtimes for finding the best known solutions are displayed in the columns Time (s). The average gaps to the best upper bound found by Gurobi after a maximum of 3600 seconds are displayed in columns Gap (%).

Test Instance	**Gurobi (3600 s)**			**ALNS Heuristic (600 s)**		
	Objective Value	Time (s)	Gap (%)	Objective Value	Time (s)	Gap (%)
10-9-2-a	617.16	3238.83	0.00	605.41	374.30	1.94
10-9-2-b	580.70	3358.36	0.00	567.71	433.39	2.29
10-9-2-c	595.08	1936.51	0.00	593.00	401.05	0.35
Average 10-9-2	597.65	2844.57	0.00	588.71	402.91	1.53
15-12-2-a	864.04	3601.52	5.24	819.72	547.56	10.93
15-12-2-b	855.49	3601.12	6.87	824.83	538.17	10.84
15-12-2-c	864.49	3600.23	2.34	819.00	399.07	8.03
Average 15-12-2	861.34	3600.96	4.82	821.18	494.93	9.93
20-15-2-a	1076.92	3600.31	2.90	993.19	629.01	11.57
20-15-2-b	1113.24	3600.23	3.90	1000.92	354.57	15.56
20-15-2-c	1109.48	3600.34	2.93	986.87	769.08	15.72
Average 20-15-2	1099.88	3600.29	3.24	993.66	584.22	14.28
30-20-2-a	1697.35	3602.63	9.40	1596.95	1794.60	16.27
30-20-2-b	1787.34	3601.24	6.36	1608.23	1441.94	18.21
30-20-2-c	1755.68	3600.70	4.07	1618.77	2168.90	12.87
Average 30-20-2	1746.79	3601.52	6.61	1607.98	1801.81	15.78
40-25-2-a	2078.43	3600.27	14.21	1957.13	2746.22	21.28
40-25-2-b	2148.96	3600.45	7.80	1977.72	2832.50	17.13
40-25-2-c	2238.22	3600.21	8.04	2045.04	3341.13	18.25
Average 40-20-2	2155.20	3600.31	10.02	1993.30	2973.28	18.89
50-30-2-a	(-)	3600.10	(-)	2322.61	1803.33	(-)
50-30-2-b	(-)	3600.32	(-)	2084.13	2043.38	(-)
50-30-2-c	(-)	3602.30	(-)	2260.79	1780.39	(-)
Average 50-30-2	(-)	3600.57	(-)	2222.51	1875.70	(-)
Average Total	1076.81	3334.17	4.11	1371.22	1355.48	10.07

Green cells indicate best average objective values.

(-) in *Objective Value* indicates no feasible solution found.

(-) in *Gap (%)* indicates no upper bound found.

Average values are found replacing (-) with 0.

Table E.9: Evaluation of how dynamic pricing affects the overall profit and runtime. Three versions of each test instance type are generated. Each test instance is run ten times with a time limit of 600 seconds using both dynamic and static pricing. The average objective values over the ten runs are displayed in the columns Objective Value. The average runtimes the ALNS heuristic spent finding the best known solution over the ten runs are displayed in the columns Time (s).

Test Instance	**Static Pricing**		**Dynamic Pricing**	
	Objective Value	Time (s)	Objective Value	Time (s)
5-4-1-a	90.91	0.00	236.60	157.15
5-4-1-b	170.26	0.00	303.83	181.19
5-4-1-c	156.73	0.00	231.39	157.37
Average 5-4-1	139.30	0.00	257.27	165.24
6-6-1-a	238.76	0.00	380.02	187.93
6-6-1-b	203.98	0.00	363.58	286.52
6-6-1-c	268.26	0.00	344.24	183.69
Average 6-6-1	237.00	0.00	362.61	219.38
8-8-1-a	419.39	0.00	503.88	60.06
8-8-1-b	388.57	0.00	558.49	197.48
8-8-1-c	361.70	0.00	523.53	376.57
Average 8-8-1	389.89	0.00	528.63	211.37
10-9-2-a	399.62	0.00	544.09	353.23
10-9-2-b	362.07	0.00	570.17	361.95
10-9-2-c	400.58	0.00	598.18	388.23
Average 10-9-2	387.43	0.00	570.82	367.80
15-12-2-a	531.08	0.00	710.43	373.20
15-12-2-b	619.78	0.00	812.60	448.71
15-12-2-c	620.70	0.00	810.30	196.49
Average 15-12-2	590.52	0.00	777.78	339.47
20-15-2-a	554.95	0.00	963.68	502.62
20-15-2-b	736.86	0.00	833.81	385.66
20-15-2-c	646.35	0.00	923.73	406.75
Average 20-15-2	646.05	0.00	907.07	431.68
30-20-2-a	1052.33	0.00	1181.19	315.96
30-20-2-b	989.45	0.00	1256.49	398.80
30-20-2-c	1199.49	0.00	1353.58	455.61
Average 30-20-2	1080.42	0.00	1263.76	390.12
40-25-2-a	1453.63	0.00	1568.50	484.08
40-25-2-b	1468.05	0.00	1459.30	435.64
40-25-2-c	1369.90	0.00	1579.88	554.41
Average 40-25-2	1430.53	0.00	1535.89	491.38
50-30-2-a	2184.86	0.00	1948.74	470.07
50-30-2-b	1658.90	0.00	1822.89	561.19
50-30-2-c	1704.89	0.00	1822.95	513.32
Average 50-30-2	1849.55	0.00	1864.86	514.86
Average Total	729.98	0.00	879.55	341.08

Green cells indicate best average objective values.

Table E.10: Evaluation of how dynamic pricing affects the employee-based relocations. Three versions of each test instance type are generated. Each test instance is run ten times with a time limit of 600 seconds using both dynamic and static pricing. Columns Relocations Performed display the average number of employee-based relocations performed over the ten runs. Columns Active Employees display on average how many employees were allocated car-moves. Columns Relocation Costs display the average overall cost relied to the relocations performed.

Test Instance	**Static Pricing**			**Dynamic Pricing**		
	Relocations Performed	Active Employees	Relocation Costs	Relocations Performed	Active Employees	Relocation Costs
5-4-1-a	1.00	1.00	-5.65	3.00	1.00	-11.60
5-4-1-b	1.00	1.00	-4.36	0.90	0.90	-2.61
5-4-1-c	1.00	1.00	-2.39	2.40	1.00	-5.78
Average 5-4-1	1.00	1.00	-4.13	2.10	0.97	-6.66
6-6-1-a	1.00	1.00	-4.24	3.00	1.00	-12.16
6-6-1-b	1.00	1.00	-4.24	3.20	1.00	-12.17
6-6-1-c	1.00	1.00	-5.65	2.10	1.00	-8.42
Average 6-6-1	1.00	1.00	-4.71	2.77	1.00	-10.92
8-8-1-a	1.00	1.00	-6.97	1.00	1.00	-2.96
8-8-1-b	1.00	1.00	-5.58	2.30	1.00	-8.01
8-8-1-c	1.00	1.00	-4.24	2.20	1.00	-11.69
Average 8-8-1	1.00	1.00	-5.60	1.83	1.00	-7.55
10-9-2-a	1.00	1.00	-4.21	2.00	1.70	-13.25
10-9-2-b	1.00	1.00	-4.21	3.40	1.70	-13.78
10-9-2-c	1.00	1.00	-3.64	1.60	1.30	-4.16
Average 10-9-2	1.00	1.00	-4.02	2.33	1.57	-10.39
15-12-2-a	1.00	1.00	-6.14	4.70	2.00	-20.35
15-12-2-b	1.00	1.00	-2.96	4.70	2.00	-16.36
15-12-2-c	1.00	1.00	-6.34	4.50	2.00	-16.57
Average 15-12-2	1.00	1.00	-5.15	4.63	2.00	-17.76
20-15-2-a	1.00	1.00	-5.53	7.90	2.00	-34.92
20-15-2-b	3.80	1.40	-13.62	7.10	2.00	-28.85
20-15-2-c	1.00	1.00	-6.06	6.80	2.00	-24.27
Average 20-15-2	1.93	1.13	-8.40	7.27	2.00	-29.34
30-20-2-a	2.40	1.20	-10.24	6.40	2.00	-28.67
30-20-2-b	1.70	1.10	-5.32	7.10	2.00	-32.27
30-20-2-c	4.50	1.50	-22.77	7.90	2.00	-37.54
Average 30-20-2	2.87	1.27	-12.78	7.13	2.00	-32.83
40-25-2-a	2.40	1.20	-12.06	6.20	2.00	-31.26
40-25-2-b	3.80	1.40	-17.75	4.70	2.00	-22.16
40-25-2-c	1.70	1.10	-9.74	6.20	2.00	-28.00
Average 40-25-2	2.63	1.23	-13.18	5.70	2.00	-27.14
50-30-2-a	7.40	2.00	-36.02	4.60	2.00	-23.46
50-30-2-b	2.00	1.00	-10.16	7.30	2.00	-35.21
50-30-2-c	2.60	1.10	-11.90	6.50	2.00	-31.79
Average 50-30-2	4.00	1.37	-19.36	6.13	2.00	-30.15
Average Total	1.50	1.06	-6.90	4.34	1.63	-18.85

Table E.11: Evaluation of how dynamic pricing affects the satisfied requests. Three versions of each test instance type are generated. Each test instance is run ten times with a time limit of 600 seconds using both dynamic and static pricing. Columns Satisfied Requests display how many customer requests were satisfied on average over the ten runs. The average profits relied to each request are displayed in columns Profit per Request.

Test Instance	**Static Pricing**		**Dynamic Pricing**	
	Satisfied Requests	Profit per Request	Satisfied Requests	Profit per Request
5-4-1-a	1.76	57.26	3.53	71.28
5-4-1-b	2.96	58.69	3.78	80.70
5-4-1-c	2.88	55.18	2.78	85.71
Average 5-4-1	2.53	57.04	3.36	79.23
6-6-1-a	4.68	52.18	4.57	85.94
6-6-1-b	3.84	53.64	5.27	71.63
6-6-1-c	4.84	56.70	4.42	79.75
Average 6-6-1	4.45	54.18	4.75	79.11
8-8-1-a	6.32	67.20	6.80	74.57
8-8-1-b	6.20	63.71	6.75	83.98
8-8-1-c	5.96	61.48	6.78	79.19
Average 8-8-1	6.16	64.13	6.77	79.25
10-9-2-a	7.08	57.19	7.66	72.58
10-9-2-b	7.32	49.92	7.81	74.76
10-9-2-c	7.80	51.83	7.66	78.88
Average 10-9-2	7.40	52.98	7.71	75.41
15-12-2-a	10.16	52.82	10.43	70.03
15-12-2-b	10.12	61.57	10.22	81.14
15-12-2-c	10.84	57.84	10.68	77.44
Average 15-12-2	10.37	57.41	10.45	76.20
20-15-2-a	10.60	52.64	12.74	78.37
20-15-2-b	12.30	60.94	11.38	75.82
20-15-2-c	11.72	55.64	12.42	76.58
Average 20-15-2	11.54	56.40	12.18	76.92
30-20-2-a	15.78	67.19	17.09	70.96
30-20-2-b	15.92	62.39	17.10	75.49
30-20-2-c	17.09	71.22	17.15	81.44
Average 30-20-2	16.26	66.93	17.11	75.96
40-25-2-a	21.14	69.24	20.46	78.42
40-25-2-b	20.90	70.98	20.79	71.22
40-25-2-c	19.74	69.74	20.19	79.89
Average 40-25-2	20.59	69.99	20.48	76.51
50-30-2-a	26.37	84.23	25.98	75.98
50-30-2-b	23.68	70.47	22.72	81.87
50-30-2-c	24.97	68.68	24.82	74.83
Average 50-30-2	25.01	74.46	24.51	77.56
Average Total	11.54	60.87	11.71	77.76

Eilertsen, Falck-Pedersen, Henriksen

The Stochastic Electric-Vehicle Relocation Problem with Dynamic Pricing

NTNU
Norwegian University of
Science and Technology

SAPIENZA
UNIVERSITÀ DI ROMA