The Dynamic Electric Vehicle Relocation Problem

An Adaptive Large Neighborhood Search

Simen Hellem
Carl Andreas Julsvoll
Magnus Nyborg Moan

Industrial Economics and Technology Management

Submission date:	May 2018
Supervisor:	Henrik Andersson, IØT
Co-supervisor:	Kjetil Fagerholt, IØT
	Giovanni Pantuso, University of Copenhagen

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Problem Description

The purpose of this thesis is to develop a solution method for the Dynamic Electric Vehicle Relocation Problem (DE-VReP). The DE-VReP is concerned with recharging and relocation of a fleet of electric cars in a free-floating carsharing system. The main goal of the DE-VReP is to maximize the customer demand served in a cost-effective way.

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 completed in the fall of 2017.

We would like to express our sincere gratitude to our supervisors Professor Henrik Andersson, Professor Kjetil Fagerholt, and Postdoctoral Fellow Giovanni Pantuso for valuable discussions and constructive feedback. Your willingness to review and engage in our work has been an essential contribution to the final result. We appreciate your enthusiasm and involvement in our research.

Trondheim, May 25, 2018

Simen Solemdal Hellem, Carl Andreas Julsvoll, Magnus Moan

Abstract

Carsharing systems have gained attention and popularity in recent years. Especially urban areas are embracing the advantages of carsharing such as economic convenience and environmental benefits. Many of the challenges faced by carsharing organizations can be modeled and solved using mathematical programming. This makes carsharing systems interesting from an operations research (OR) perspective. This thesis examines a free-floating electric carsharing system and the operational challenges associated with such a system. A free-floating carsharing system consists of a fleet of rental cars managed by a carsharing organization (CSO). Free-floating systems allow the users to pick-up and drop-off cars at any location within the operating area of the CSO. Modern technologies play an essential role in the development of new carsharing systems. Remote tracking, including fuel/battery levels and positions of cars, and mobile applications are examples of such technologies. Since electric vehicles are environmentally friendly and well suited for shorter trips, they are commonly used in carsharing systems.

Free-floating carsharing systems are prone to unbalanced distributions of rental cars due to patterns in customer demand. Therefore, the problem of optimal relocation of rental vehicles to best meet demand is of particular interest for the CSOs. Efficient relocation is paramount for the profitability of the CSO. Other operational aspects include refueling/recharging and maintenance of the car fleet.

A literature review of carsharing systems is conducted. The review focuses on operational problems and heuristics for solving them. The findings indicate that few of the solution approaches in the literature include all aspects of a realistic carsharing system. Elements usually excluded are the recharging/refueling and maintenance of cars.

This thesis presents a solution method for solving the Dynamic Electric Vehicle Relocation Problem (DE-VReP), adopting a Rolling Horizon framework. The DE-VReP is concerned with routing a set of service employees, as they relocate and recharge cars, in a free-floating carsharing system with electric vehicles. The goal of the problem is to maximize the customer demand served, cost-effectively. In the Rolling Horizon framework, a static sub-problem of the DE-VReP, denoted the E-VReP, is solved iteratively. An Adaptive Large Neighborhood Search

(ALNS) heuristic is developed to solve the E-VReP. Solving the DE-VReP yields routes for the service employees throughout the day.

The solution method for solving the DE-VReP is tested in a simulation model mimicking a theoretical carsharing system based on the city of Oslo, Norway. All travel times are real travel information. Performance of the solution method for the DE-VReP is evaluated based on the amount of customer demand served.

The ALNS heuristic for the E-VReP provides effective relocations and recharging of cars. Basic maintenance is also incorporated. The heuristic shows promising results in solving the E-VReP, achieving on average a deviation of 0.5% to best-known solutions on a set of test instances. Comparably, a greedy approach achieves an average deviation of 45.6%.

The solution method for the DE-VReP provides high-quality solutions in reasonable computation time for problem instances with over 380 rental cars. The solution method is versatile and adaptive to fit the preferences of CSOs. Compared to a greedy solution method for the DE-VReP, the proposed solution method increases demand serves by 7.86 percentage points.

Sammendrag

Bildelingssystemer har blitt populære i nyere tid. Økonomiske og miljøvennlige fordeler har gjort bildelingssystemer til et attraktivt alternativ til eie av egen bil, særlig i urbane strøk. Flere av utfordringene som må løses for å drifte et bildelingssystem kan løses ved hjelp av blandet heltalls-optimering. Bildeling er dermed interessant fra et operasjonsanalytisk perspektiv. Denne masteroppgaven tar for seg de operasjonelle utfordringene ved å drifte et frittflytende bildelingssystem med elektriske biler. Et slikt system består av en flåte med elektriske biler driftet av en bildelingsorganisasjon (Carsharing Organization - CSO). I et frittflytende bildelingssystem kan kundene parkere bilene hvor de vil innenfor driftsområdet til bildelingsorganisasjonen. Et driftsområde er vanligvis på størrelse med en by. Moderne teknologi, særlig mobilteknologi, har vært avgjørende for fremveksten av bildelingssystemer. Bildelingssystemer med elektriske biler har økt i popularitet, mye fordi elektriske biler er miljøvennlige og godt egnet til korte kjøreturer i storbyer.

I et frittflytende bildelingssystem kan distribusjonen av biler bli ujevn som en konsekvens av bruksmønstre. For å dekke mest mulig etterspørsel er det derfor viktig med effektiv reposisjonering. Effektiv håndtering av bilene er en avgjørende faktor for lønnsomheten til bildelingsorganisasjonene. Andre operasjonelle problemer inkluderer lading og vedlikehold av bilflåten.

En litteraturundersøkelse av bildelingstjenester er gjennomført. Hovedfokuset i litteraturundersøkelsen har vært på operasjonelle problemer samt heuristiske løsningsmetoder for disse. Få av løsningsmetodene avdekket i litteraturundersøkelsen tar høyde for alle aspektene ved et bildelingssystem. Blant annet er lading og vedlikehold av bilflåten ofte utelatt.

Denne masteroppgaven presenterer en løsningsmetode for det Dynamiske Reposisjoneringsproblemet for Elektriske Biler (Dynamic Electric Vehicle Relocation Problem - DE-VReP). En løsning på DE-VReP git ruter for de ansatte. Målet er å finne de rutene som tilfredsstiller mest mulig kundeetterspørsel på en kostnadseffektiv måte. Den foreslåtte løsningsmetoden bruker et rullende horisont (Rolling Horizon) rammeverk. Innenfor et slikt rammeverk oppstår det statiske subproblemer, kalt E-VReP. Et adaptivt, variabelt nabolagssøk (Adaptive Large Neighborhood Search - ALNS) har blitt utviklet for å løse E-VReP. En løsning på DE-VReP inkluderer reiseruter for de ansatte i bildelingsorganisjonen i løpet av en arbeidsdag. Rutene indikerer hvor de ansatte skal reise og hvilke biler de skal reposisjonere og lade.

Løsningsmetoden er testet i et simuleringsrammeverk som etterligner et teoretisk bildelingssystem i byen Oslo, Norge. Alle reisetider er basert på sanntidsinformasjon. Kvaliteten av løsningsmetoden for DE-VReP evalueres etter mengden kunde-etterspørsel tilfredsstilt.

Løsningsmetoden for E-VReP lager effektive reiseruter for de ansatte. Lading og enkel vedlikehold av bilene er også inkludert i rutene. På et sett av tester oppnår løsningsmetoden et gjennomsnitt som er 0.5% unna de beste løsningene som noen gang er funnet. En grådig løsningsmetode oppnår derimot løsninger som i gjennomsnitt er 45.6% unna de beste løsningene.

Løsningsmetoden for DE-VReP oppnår også løsninger av høy kvalitet etter kort tid på problemer med over 380 biler. Løsningsmetoden er lett å tilpasse preferansene til en CSO. Sammenliknet med en grådig løsningsmetode for DE-VReP, så er den foreslåtte løsningsmetoden 7.86 prosentpoeng bedre.

Contents

List of Figures

List of Tables

Abbreviations

CSO	=	Carsharing Organization
DE-VReP	=	Dynamic Electric Vehicle Relocation Problem
E-VReP	=	Electric Vehicle Relocation Problem
DVRP	=	Dynamic Vehicle Routing Problem
VRP	=	Vehicle Routing Problem
MDP	=	Markov Decision Process
OR	=	Operations Research
MIP	=	Mixed Integer Program
ALNS	=	Adaptive Large Neighborhood Search
TS	=	Tabu Search
GA	=	Genetic Algorithm
LST	=	Local Search Transformation
DES	=	Discrete Event System
DS	=	Demand Served

Chapter 1

Introduction

Carsharing systems have existed in various forms for at least 70 years. Recently, carsharing as means of transportation has gained traction due to larger urban populations and increased focus on the environment. Furthermore, privately owned cars suffer from low utilization, with an average usage of one hour a day (Li et al., 2016). Each car in carsharing systems replaces at least four to eight privately owned cars, consequently reducing the pressure on parking spots in urban areas (Loose, 2009). Traditional carsharing systems are station-based systems, in which a fleet of cars is distributed among different stations. The users usually reserve a car before use, and then pay a time and distance based fee (Shaheen et al., 2015). The majority of carsharing systems are still station-based. However, the increased demand for flexibility from users has encouraged the development of free-floating systems. In such modern systems, cars can be picked up and delivered at any location within a specified area. Demand patterns can lead to unbalanced distributions of cars, both in station-based and free-floating systems. Hence, owners of these systems strive to maintain distributions of cars that meet the customers' demand patterns.

1.1 Purpose of Thesis

The purpose of this thesis is to investigate and solve the Dynamic Electric Vehicle Relocation Problem (DE-VReP) in free-floating carsharing systems using electric vehicles. The goal of this operational problem is to maximize demand served, in a cost-effective way, by relocating cars. Relocation includes recharging and transportation of rental cars to improve the distribution of cars in the system. Studies have shown that using bikes as mean of travel for the service employees is an effective strategy for carsharing systems (Bruglieri et al., 2014). When solving operational problems, it is common to face trade-offs between the system owner's profits and customer satisfaction. Higher customer satisfaction would make the system more attractive to new customers, which again could lead to higher profits.

The first goal of this thesis is to develop a solution method for realistic instances of the DE-VReP. The proposed solution method adopts a Rolling Horizon framework, following the most recent research on dynamic vehicle routing problems. In the Rolling Horizon framework, a static sub-problem to the DE-VReP, the Electric-vehicle Relocation Problem (E-VReP) is solved at each decision epoch. The second goal of this thesis is to develop a solution method for the E-VReP to be used in the Rolling Horizon framework. Both a Mixed Integer Programming (MIP) model and an Adaptive Large Neighborhood Search (ALNS) are heuristic presented. A literature review is also conducted.

1.2 Structure of Thesis

This thesis begins with an introduction to the carsharing industry in Chapter 2. A literature review focusing on the operational and dynamic aspects of carsharing as well as heuristic methods are presented in Chapter 3. Chapter 4 describes the DE-VReP in detail. Chapter 5 proposes a solution method for the DE-VReP adopting a Rolling Horizon framework. Chapters 6 and 7 describe and model the E-VReP. An ALNS heuristic for solving the E-VReP is presented in Chapter 8. Chapter 9 presents a suitable simulation model. Chapter 10 discusses implementation details of the DE-VReP and the E-VReP, respectively. Chapter 11 presents the results of the computational study. Finally, Chapter 12 concludes this thesis and presents possibilities of further research.

Chapter 2

Background

The world's population is expected to grow to approximately 10 billion by the year 2050 (UN, 2017). The majority of this growth is anticipated to occur in urban areas. With existing transport services, a proportional increment in traffic would lead to large increments in pollution. A significant change to existing transportation systems and infrastructures is needed to cope with this growth. Carsharing systems are transportation systems which attempt to solve these problems (Martin and Shaheen, 2016). This chapter introduces the modern carsharing systems and their origins. Section 2.1 presents the characteristics of carsharing systems. A brief overview of the history of carsharing is given in Section 2.2.

2.1 Carsharing Systems

Carsharing systems enable customers to rent cars for a short period of time. A carsharing system is owned and maintained by a Carsharing Organization (CSO). First-time users typically sign up through a website or a mobile application to get access to the system. Users of the system can book available rental cars. In contrast to traditional car rentals, users rent cars without any direct interaction with the CSO. Accessing the rental cars is usually done either through smartphone applications or electronic keys sent to the customers from the CSO. Station-based and free-floating systems are the two main categories of carsharing systems.

2.1.1 Types of Carsharing Systems

In station-based systems, the users can only pick up and deliver rental cars at specific stations owned by the CSO. Station-based systems are either one-way or two-way systems. In one-way systems, the users can pick up and deliver cars at different locations. In two-way systems, the users deliver the rental car to the same location as they picked it up.

Free-floating systems differ from station-based by allowing the customers to pick up and deliver rental cars at any location within the operating area of the CSO. Compared to station-based systems, free-floating carsharing systems are more convenient for the customers, as they offer more flexibility. However, this flexibility comes at a price for the CSO. Free-floating systems are prone to uneven distributions of rental cars caused by patterns in customer usage. Similar problems may occur in one-way systems, but not to the same degree as in a free-floating system. Carsharing systems usually consist of either gasoline or electrical cars. An example of an electric rental car is shown in Figure 2.1.

Figure 2.1: A electric car owned by BMWs CSO, ReachNow (www.reachnow.com).

2.1.2 Revenues and Costs

CSOs generate revenue by collecting subscription fees from the customers. Most CSOs also charge a fee dependent on distance driven or time used in a rental car. Costs for the CSOs include capital costs associated with the rental cars and charging stations as well as operational costs. Service employees maintain, recharge/refuel and relocate the fleet of rental cars. Operational costs for the CSOs mainly consists of wear, toll fees, and electricity usage of the cars in use.

2.1.3 Challenges

CSOs face challenges on three levels: strategic, tactical and operational. The strategic level is concerned with the number of stations and their locations. On the tactical level, the CSOs wish to find the optimal number of rental cars and service employees needed in the system. The operational level is concerned with optimizing the day to day operations of the CSOs. Recharging/refueling and maintaining the cars are operational concerns faced by all CSOs. In one-way station-based or free-floating systems, the CSOs must also consider whether or not the relocation of cars is necessary to meet demand. Demand in the different parts of a CSO's operating area may change dynamically during a day or a week, and relocation may enable the CSO to increase its revenues by satisfying more demand. Tasks on the operational level are performed either by service employees hired by the CSO or by the customers themselves. Some systems use different reward schemes to incentivize customers to charge/fuel and relocate cars. Service employees may travel between the rental cars and the stations in several ways, including public transportation, folding bikes (see Figure 2.2) and shuttle buses carrying several service employees. This thesis proposes a solution method for the DE-VReP, which is an operational problem faced by free-floating electric carsharing systems.

Figure 2.2: Folding bike in the trunk of a car (www.decathlon.it).

2.2 Carsharing History

The following section, describing the history of carsharing, is based on Shaheen et al. (1998) and Shaheen et al. (2015).

The history of carsharing started in Europe in the late 1940s. One of the first known carsharing systems known as "Sefage" was established in Zurich, Switzerland, in 1948. Carsharing systems started out as a solution for individuals who could not afford to own a car. From the late 1940s and until the mid-1980s there were very few successful commercial carsharing ventures. The primary challenge for carsharing systems in this period was the transition from neighborhood-based programs to actual business ventures. In 1987, Mobility Carsharing Switzerland was founded, with a fleet of 1 000 cars. A year later, StattAuto Berlin was founded with around 200 cars. These two companies are regarded as worldwide pioneers within the field of carsharing. Both companies exist today, and they have seen a steady growth in both fleet size and number of customers. In total, the carsharing industry has shown increasing growth since the mid-2000s, see Figure 2.3.

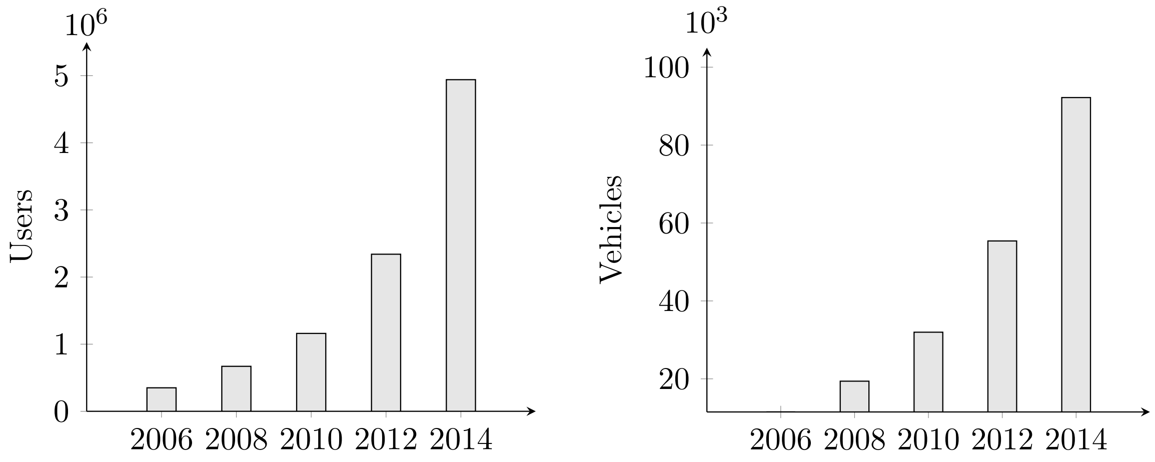


Figure 2.3: Development of carsharing users and number of cars owned by CSOs globally from year 2006 to 2014 (Frost and Sullivan, 2014).

Modern carsharing systems originated in Europe. Europe is still the continent with the highest number of customers and rental cars. In recent times, North America, Asia (mostly Japan and Singapore) and Europe have witnessed growth in the number of carsharing users. Modern technologies have played a prominent role in this growth. For instance, GPS, mobile applications, keyless car access and electric cars have all been paramount in modernizing the carsharing industry. These technologies among others have enabled 24-hour access to rental cars without forcing CSOs to hire additional service employees.

There still exist a wide range of strategic, tactical and operational problems faced by CSOs. These problems have gained considerable interest in recent years, especially from the Operation Research (OR) community. Most research done in the past regarding carsharing systems have focused on station-based systems. These systems are simpler to operate for the CSOs. Today, many CSOs focus on free-floating systems as these systems provide more flexibility to the users. In such systems, finding the optimal ways to relocate the cars to satisfy the most demand is the main problem.

Chapter 3

Literature Survey

This chapter discusses recent literature on solution methods to challenges related to carsharing systems. Section 3.1 discusses the types of optimization decisions faced by CSOs. Section 3.2 discusses different approaches to modeling and solving the DE-VReP and dynamic car routing problems in general. Section 3.3 introduces heuristic methods applicable to solving the E-VReP. Finally, Section 3.4 summarizes this survey. For a more extensive literature survey covering carsharing literature, please see Folkestad and Hansen (2017).

3.1 Optimization in Carsharing Systems

Boyaci and Geroliminis (2015) divide the optimization of carsharing problems into a hierarchy of three categories; the strategic, the tactical, and the operational level. When optimizing on one level, the decisions made in the above levels are usually considered fixed. However, some papers, such as Martinez et al. (2012), solve problems from both the strategic and the tactical levels simultaneously. Sections 3.1.1, 3.1.2 and 3.1.3 present different challenges in each of the three categories together with possible solution approaches found in the literature. Figure 3.1 gives an overview of the different challenges in the three categories.

3.1.1 The Strategic Level

Decisions at the strategic level are concerned with the determination of the optimal number, size, and location of stations in a fixed-station carsharing system. Kathrin S. Kühne and Breitner (2016) propose a decision support system using Mixed Integer Programming (MIP) to optimize the location of charging stations for electric cars. Their model considers parameters such as annual lease payment, expected travel time of customers and charging time of cars. However, Boyaci and Geroliminis (2015) show that a CSO's profits are correlated with customer satisfac-

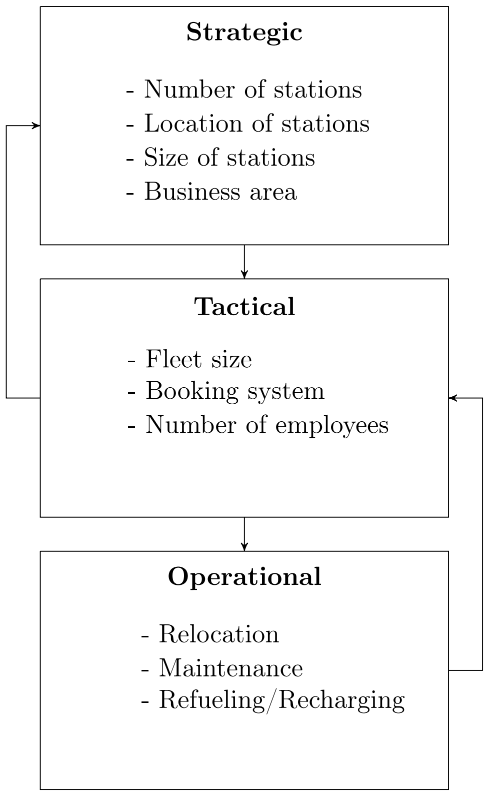


Figure 3.1: Planning levels for a CSO

tion. Hence, station accessibility as incorporated in the MIP of Correia and Antunes (2012) can lead to increased profits for the CSO.

In free-floating carsharing systems, the strategic level is concerned with the optimal distribution of cars in the system to increase the demand met. In the paper by Simone Weikl (2012), the authors use classification and demand clustering algorithms to determine the optimal car locations to satisfy the customer demand. They classify the potential parking spots into hot and cold spots based on identified demand patterns. Such an approach can be done in advance of the relocations and is relevant for the solution method used in this paper.

3.1.2 The Tactical Level

With the optimal number, size and location of stations decided, the tactical level is concerned with optimizing the fleet size of rental cars. To optimize the fleet size of an existing carsharing system George and Xia (2011) and Cepolina and Farina (2012) analyze the demand imbalance patterns in the operating area. George and Xia (2011) deal with the customer costs of waiting by using a queuing network model that gives indications of beneficial fleet changes. Cepolina and Farina (2012) include customer costs that are dependent on their waiting time into the objective function.

Also, decisions made on the tactical level deal with how customers make their bookings. The trip-booking scheme chosen by the CSO has a significant impact on both the usage patterns of the system as well as the ease of car relocation. For instance, if users are only allowed to book cars in advance of their trips, the system would be less attractive for impulsive users. However, bookings in advance make the process of car relocation more efficient, especially if the users also specify their planned destinations. Both Correia and Atunes (2014) and Kaspi and Tzur (2014) show that when users allow the CSO to get access to more information, the CSO ends up with higher profits. Correia and Atunes (2014) use a MIP to demonstrate that a CSO not exploiting real-time information about the rental cars can have its profits reduced by as much as 80%[1].

3.1.3 The Operational Level

The operational level is concerned with optimizing the core business of the CSO which mainly consists of three operational decisions; car relocation, maintenance, and refueling/recharging. Relocation done by the CSO, known as employee-based or operator-based relocation, has received the most attention in the OR carsharing literature. Employee-based relocation is the term used in this thesis and is elaborated in this section.

Most research on employee-based relocation suggests solution methods using simulation models or Mixed Integer Programming. In Barth and Todd (1999) the researchers develop a queuing-based discrete event simulation model for performance analysis of a station-based carsharing system. Their model determines if the system is in need of relocation or not. Other researchers combine simulation models with simple rules of thumb for decision making, including Kek et al. (2006), Kek et al. (2009) and Jorge et al. (2014). Common for these papers is that stations with lack of cars are filled up with cars from nearby stations with surplus of cars.

Boyaci and Geroliminis (2015) use a multi-objective MIP model to solve the employee-based relocation problem for a one-way station-based system with electric cars. The model tries to maximize the profits of the CSO using a weighted sum approach. Boyaci and Geroliminis (2015) explore the efficient frontier of solutions by varying the weighting of the objectives. Boyaci et al. (2017) further develop the approach by Boyaci and Geroliminis (2015) by modeling the problem as a MIP and tests it in a discrete event-simulation framework. Boyaci et al. (2017) iteratively resolve the MIP to find feasible solutions in an environment with stochastic demand. As was done by Kek et al. (2009), Boyaci et al. (2017) also include the routing of service employees in their model and not only the relocation of rental cars.

3.2 Dynamic Vehicle Routing Problems

The DE-VReP shares characteristics with Dynamic Vehicle Routing Problems (DVRP). Vehicle routing with dynamic customer demand is one of those characteristics. Therefore, it is appro-

[1]User flexibility with regards to parking policies affects this number.

priate to include a description of DVRP's modeling approaches and solution methods. The interest for DVRPs among operation researchers has increased in the recent years. This interest is evident in the survey by Psaraftis et al. (2016), where 51 out of 117 DVRP papers cited are published after 2011. DVRPs do however have a long history in the OR community, with the first known article on the subject being Wilson and Colvin (1977). Pillac et al. (2011) argue that the increased interest in DVRPs is a consequence of technological advances in both computer hardware and software. More efficient hardware has made previously intractable problems solvable. Also, the current wave of research in the field of machine learning and predictive analysis has made forecasting, which often is a crucial part of solving DVRPs, more reliable.

Psaraftis et al. (2016) categorize a vehicle routing problem (VRP) to be either dynamic or static, and either stochastic or deterministic. Hence, four VRPs are possible as seen in Table 3.1.

Table 3.1: VRP Classification

	Static	Dynamic
Deterministic	A static VRP is deterministic if all input is known. Can be solved before the start of the planning period.	A VRP where input is revealed over time. No information about the input is known. Solved many times, when new information is available.
Stochastic	A VRP where input is not known, but a distribution for the input is known. The problem is solved before the input is known.	A VRP where input is revealed over time. A probability distribution for the input is known. Solved many times, when new information is available.

3.2.1 Modeling DVRPs

Modeling a DVRP is considerably more difficult than modeling its static counterpart. The need for incorporation of continuous change in the input has forced operation researchers to alter the way a dynamic problem is modeled, compared to that of a static. Ulmer et al. (2017) present a Markov Decision Process (MDP) framework for DVRPs. The authors argue that current DVRP research lacks a connection between solution method and modeling, and that this gap can be bridged by introducing a MDP framework. Ulmer et al. (2017) show how a route-based MDP can be used to model a DVRP while at the same time being closely coupled with solution methods that optimizes iteratively. MDPs have been used as a tool to solve DVRPs before Ulmer et al. (2017) presented their model. Both Thomas (2007) and Secomandi and Margot (2009) use Markov Chains to model a DVRP. However, Ulmer et al. (2017) generalize the MDP framework, making it reusable for a variety of different DVRP problems. The model presented by Ulmer et al. (2017) extends traditional MDPs by making the current planned route a part of the MDP framework.

A conventional MDP models a process over time where a actions occur at different time steps referred to as *decision epochs*. At each decision epoch, the process is in a specific *state*, which describes all information about the process. The process evolves from one state to the next during a transition which is caused by a particular *action*. Each action has a reward associated with it. A transition may or may not be deterministic. The objective is to maximize or minimize the sum of all expected rewards over all decision epochs.

3.2.2 DVRP Solution Approaches

In modern research, most solution approaches try to solve a DVRP using periodic re-planning. This approach, often called a *Rolling Horizon* approach, has been used with success by Chen and Xu (2006), Kilby et al. (1998), Yang et al. (2002) and others. In a Rolling Horizon approach, a static vehicle routing problem (VRP) is defined for the DVRP. The VRP is solved and re-planned multiple times in succession with updated input to mimic the dynamics of a DVRP.

Chen and Xu (2006) solve a DVRP with hard time windows by using a Rolling Horizon approach. The time between each re-planning is a parameter and does not change over the duration of the planning period. Kilby et al. (1998) present a different criterion for re-planning. Re-planning is triggered when new demand arrives. Yang et al. (2002) show how different implementations of a Rolling Horizon approach can be used to solve a real-time multi-car truckload pick-up and delivery problem. They show how the approach can be used in combination with a variety of solution methods for the underlying static VRP, including both heuristics and exact optimization methods.

3.2.3 Solution Evaluation

When solving a DVRP using a series of static VRPs in a Rolling Horizon framework, one must keep in mind that short-term objectives do not necessarily yield good long-term solutions. This trade-off complicates the formulation of the objective function for the short-term static problem. In a static VRP, the objective function often maximizes profits or customer satisfaction or minimizes cost or customer dissatisfaction. Using such an objective function in a Rolling Horizon framework, routes minimizing costs in the short-term could end up being expensive in the long-term.

3.3 Relevant Heuristics for the E-VReP

Solving the E-VReP in a dynamic setting has similarities with solving the classical VRP problem. Computation time does, however, play an extra important role in the Rolling Horizon environment due to the high frequency of re-planning. Exact methods for solving large real-world instances are, therefore, often ruled out. Many different heuristics are proposed in the literature, including various neighborhood searches and evolution-inspired methods. This section introduces relevant heuristics for solving the E-VReP. Sections 3.3.1 and 3.3.2 present the two main categories of search heuristics found in the literature, namely Neighborhood Search and Evolution-Inspired Search.

3.3.1 Neighborhood Search

The most common neighborhood searches found in the literature related to the E-VReP are *Adaptive Large Neighborhood Search* (ALNS) and *Tabu Search* (TS).

ALNS is based on the Large Neighborhood Search from Shaw (1997) which extends the classic local search to include larger changes to the solution instead of marginal changes. The set of marginal changes is referred to as the *local neighborhood*. The larger changes consist of so-called *destroy and repair* heuristics. The destroy heuristics make drastic changes to the solution to explore new areas of the solution space, while the repair heuristics try to improve the solution given the result of the destroy methods. These methods enable the search to easier escape local optima. ALNS includes an adaptive approach in which the neighborhoods are chosen based on past success in obtaining solution improvements. Ropke and Pisinger (2006) describe these methods as a fruitful diversification process which increases the probability of finding good solutions.

TS, on the other hand, exploits only the local neighborhood, but in a sophisticated manner. In each iteration, the search greedily chooses the marginal change that yields the best solution improvement. In cases where no solution improvements exist, the TS selects the least worsening marginal change. TS prohibits recently chosen marginal changes for a given number of iterations, called the *tabu tenure*. The tabu tenure prevents the search from getting stuck in local optima.

Adaptive Large Neighborhood Search

The paper Bruglieri et al. (2018) proposes an ALNS for the Electric Vehicle Relocation Problem (E-VReP) in a one-way station-based carsharing system. Based on the paper Bruglieri et al. (2014), their model aims to find the optimal set of routes to maximize the number of *relocation requests* in a cost-efficient way. These requests are based on customer demand in the different stations. The relocation requests are considered *a priori* information in the model. They compare their heuristic to results from solving a MIP and a TS approach. Tested on real-life data from Milan, ALNS outperforms both of these models.

A drawback of the model from Bruglieri et al. (2018) is its applicability in a dynamic context. In the model, the service employees start and stop at a given depot, and the possibility of re-planning when new data becomes available seems absent. Chen et al. (2018) propose an

ALNS aimed to solve the static subproblems of a DVRP with heterogeneous vehicles. Similar to Bruglieri et al. (2018), they introduce several ALNS and TS operators utilizing the structure of the underlying problem. Both papers also conclude that accepting solutions based on the criteria from Simulated Annealing is beneficial. In addition, Chen et al. (2018) show that the performance of the ALNS improves when allowing for infeasible solutions during the search. The authors conclude that the use of ALNS in a dynamic VRP environment is suitable.

Tabu Search

In the recent paper Ait-Ouahmed et al. (2018), a TS for solving the E-VReP in a one-way, station-based carsharing system is proposed. Based on a carsharing service in Nice Metropolis, the model aims to maximize customer demand served. Compared to Bruglieri et al. (2018), Ait-Ouahmed et al. (2018) present an adaptation of the TS with a two-level architecture, each of them corresponding to a subproblem. The first subproblem considers only the optimal relocation of cars in order to meet the demand. The second subproblem consists of assigning service employees to the relocations found in the first subproblem. This approach speeds up the search for larger test instances.

Kirchler and Calvo (2013) suggest a way to speed up the TS by proposing a *granular* TS. Apart from the standard TS, their heuristic reduces the search space by using a reduction of the neighborhood, hence the term granular. Relocations unlikely to be part of an optimal solution are removed when defining the input to the search. The reduction of the search space is useful in solving very large problem instances. However, similar to that of Ait-Ouahmed et al. (2018), the simplification may remove the possibility of finding optimum. Kirchler and Calvo (2013) also incorporate a dynamic variation of the tabu tenure that balances between intensification and diversification. When solution improvements are found, the search intensifies by reducing the tabu tenure allowing for more marginal changes. However, with lack of improvements, the tabu tenure increases to ensure diversification.

3.3.2 Evolution-Inspired Search

Prins (2004) and Nagata and Bräysy (2009) have both proven that evolutionary algorithms, such as the Genetic Algorithm (GA) from Holland (1975), are efficient at solving VRPs. In brief, the GA mimics the process of natural evolution where the best solutions from one generation have a larger probability of surviving.

Prins (2004) was the first to introduce an efficient GA due to a novel solution representation. The representation consists of an ordered list of all nodes to visit for the service employees, but without any trip delimiters. Routes for service employees are generated by using a splitting procedure that runs in polynomial time. The new splitting procedure strengthens the genetic transmission of information from parents to children, making the search more efficient. The solution representation has been adapted to a broad class of VRP problems as described in Penna et al. (2018).

Based on the promising results of Vidal et al. (2012), Folkestad and Hansen (2017) apply the Hybrid Genetic Search with Adaptive Diversity Control (HGSADC) to a version of the E-VReP. The HGSADC of Vidal et al. (2012) incorporates new metaheuristics and population-diversity management methods. These metaheuristics utilize the breadth of the GA while at the same time enforcing aggressive improvement strategies. The population-diversity management ensures the preservation of good solution characteristics while maintaining diversity. In the E-VReP version of Folkestad and Hansen (2017), the service employees are picked up and dropped off at the locations and destinations of relocation requests. Herbawi et al. (2016) solved the same version of E-VReP using an evolutionary algorithm.

3.4 Summary of Literature Review

Recent carsharing research focuses on a variety of problems within the strategic, the tactical and the operational levels of a carsharing organization. A majority of the reviewed papers focus on operational problems, emphasizing employee-based relocation. One-way station-based carsharing systems have received the most attention. From an operational point of view, the challenges faced in a free-floating system are more difficult to solve than those for a one-way station-based system. Free-floating systems do, however, offer more flexibility to the users.

A variety of different strategies have been proposed to solve the relocation problem for both one-way and free-floating systems, including simulation models and MIP models. Most models do, however, have some shortcomings in modeling all aspects of reality. Mukai and Watanabe (2005), Kek et al. (2006), Kek et al. (2009), Nair and Miller-Hooks (2011) and Nair and Miller-Hooks (2011) all ignore routing of service employees. Boyaci and Geroliminis (2015) and Boyaci et al. (2017) both present models which include routing of service employees. However, they model the routing of service employees and the routing of rental cars as two separate problems. This separation may lead to solutions where the total costs and revenues of relocations are not properly balanced.

General dynamic vehicle routing problems have received increased attention in recent years. Most recent articles concerned with DVRPs use a Rolling Horizon approach. A variety of different solution methods for the underlying static vehicle routing problem have been proposed and tested.

The goal of this thesis is to create a solution method for large-scale versions of the DE-VReP, in free-floating carsharing systems. For practical use, it is crucial that the proposed solution is efficient with low computation times. A solution where the members of the service employees travel by public transport or foldable bikes is proposed. Bikes are used due to the efficiency described in Bruglieri et al. (2014).

Chapter 4

Problem Description of the DE-VReP

This chapter presents the *Dynamic Electric Vehicle Relocation Problem* (DE-VReP). The DE-VReP is a problem on the operational level of free-floating carsharing systems. The main goal of the problem is to maximize the demand served by the system in a cost-effective way. Section 4.1 gives an overview of the DE-VReP. Section 4.2 defines important terminology and necessary definitions. Finally, Section 4.3 introduces key considerations with regards to objectives, routing of service employees and travel times.

4.1 Overview

The DE-VReP is defined as the operational problem of deciding how to route a crew of *service employees* in a free-floating electric carsharing system. The service employees perform relocations of rental cars. Relocation in the DE-VReP denotes either relocation of rental cars between two locations to better serve demand or to transport a car for recharging. The problem is solved throughout the whole day, i.e., during the business hours, denoted the *planning horizon*. The outcome of the decision problem is *routes* for the service employees during the planning horizon, which cars each service employee relocates, and where to relocate each car. Routes consist of relocations of cars as well as travels without cars. The decision maker is the owner or manager of the system, usually the CSO. It is in the interest of the CSO that the relocation of the rental cars is carried out in the most time and cost-effective way possible.

The overall goal of the CSO is to maximize demand served, in a cost-effective way. Customer demand is served if a user requests a car, and a car is available the desired location. Demand served constitutes the primary revenue source for the CSO. Service employees relocate cars in order to better serve demand patterns. To serve demand long-term, the CSO is dependent on recharging the car fleet. There are several types of costs to consider in the DE-VReP. At the operational level, each relocation carried out are associated with costs of wear, tolls and electricity for the car in use. This cost implies that there is a trade-off between meeting demand, and accumulated cost for every relocation being carried out. There is also a cost associated with the recharging of cars, due to the cost of electricity. Nonetheless, this cost is ignored in the DE-VReP, since the recharging is a necessity for the CSO. Finally, it is assumed that decisions and costs regarding the number of service employees, the size of the car fleet and the number of charging stations are fixed. However, these decisions, considered at the strategic and tactical level, affect the ability of the CSO to serve demand.

In the DE-VReP, supply, demand, and travel times change dynamically. Cars with a remaining range below a certain threshold, or in need of maintenance, are assumed to be unavailable to customers. All other cars are available for rental. Customers can book cars that are parked and not reserved by other customers. In the DE-VReP, customers book cars on-demand. However, they are not able to reserve cars for future rentals. All bookings are made through the CSOs website or mobile application, which in real time show the available rental cars, their locations, and their battery levels.

Service employees travel using folding bikes or public transportation, as assumed in Weikl and Bogenberger (2015) and Bruglieri et al. (2014).

Figure 4.1 shows an overview of the DE-VReP.

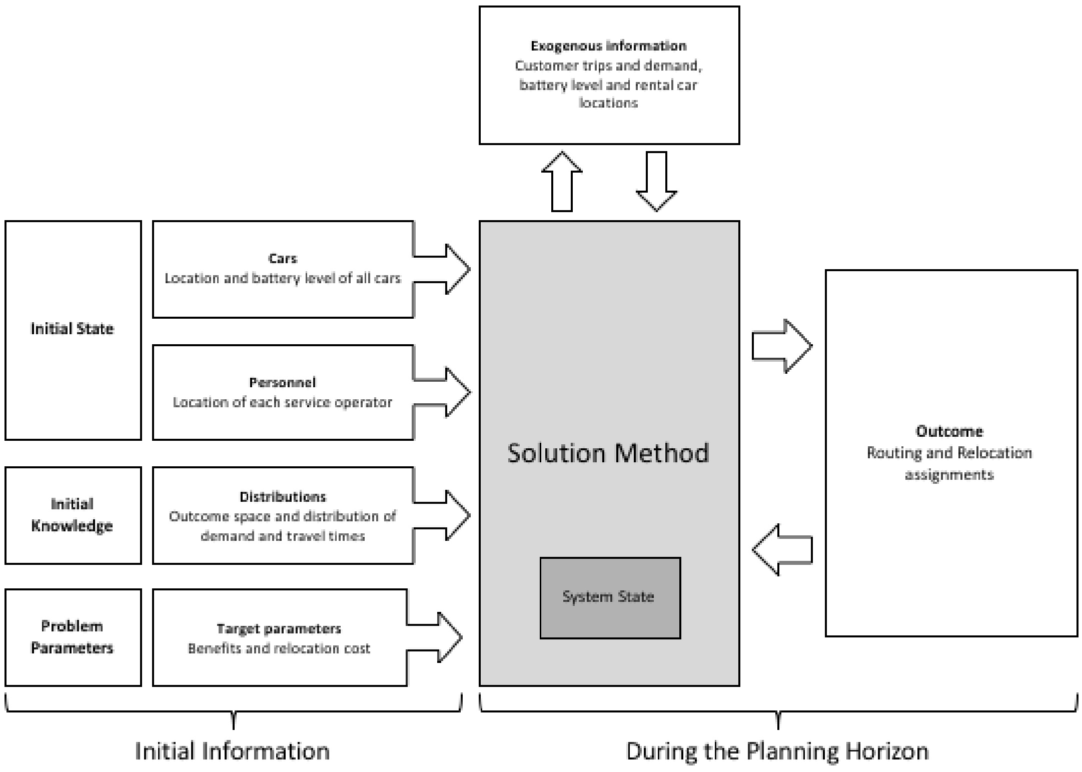

Exogenous information
Customer trips and demand, battery level and rental car locations
Initial State
Cars
Location and battery level of all cars
Personnel
Location of each service operator
Initial Knowledge
Distributions
Outcome space and distribution of demand and travel times
Problem Parameters
Target parameters
Benefits and relocation cost
Solution Method
System State
Outcome
Routing and Relocation assignments
Initial Information
During the Planning Horizon

Figure 4.1: Conceptual overview of the DE-VReP. Routes and relocations are determined based on the state of the system. The model receives exogenous information repeatedly. The system state is updated based on tasks done and exogenous information received.

4.2 Definitions

As in Weikl and Bogenberger (2015) and Folkestad and Hansen (2017), the geographical locations of cars are divided into *zones*. Each zone defines a separate geographical area. A *parking zone* is defined to be a zone where rental cars are parked and picked up by the customers. Each parking zone can contain multiple cars. A parking zone may include a charging station. A *charging zone* is defined to be the area of the charging station. Each charging station has a maximum capacity. Only charging rental cars are present in the charging zones. Charging zones are located inside parking zones as shown in Figure 4.2. With this approach, the entire operating-area can be represented as a graph. Each node in the graph corresponds to a zone. Edges are defined between all pairs of nodes. Rental cars and service employees travel between the nodes along the edges. The weight of each edge corresponds to the estimated travel time between the associated nodes. Travel times are not necessarily symmetric between two nodes.

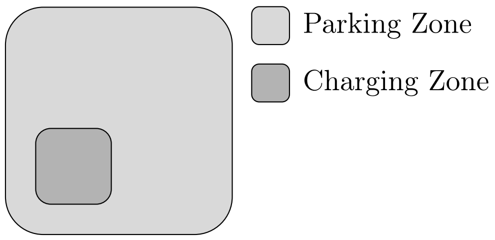


Figure 4.2: Location of a charging zone. The charging zone is separate from its associated parking zone

Figure 4.1 shows an overview of the DE-VReP. The *initial state* defines the distribution of rental cars among all nodes, battery levels of all rental cars as well as the starting positions for each service employee. The problem uses *initial knowledge* about the distributions of demand and travel times between the zones. During the planning horizon, the optimization model track the *system state* consisting of the current knowledge of the world. Target parameters include benefit of meeting demand and cost of relocating cars. *Exogenous information* consists of the trips and booking request made, the battery levels for all rental cars, and the current capacities of the charging stations.

4.3 Problem Specification

In the following sections, key concepts concerning the objective, the routing of service employees, and the time usage in the DE-VReP, are discussed.

4.3.1 Objective

The first objective of the DE-VReP is to maximize demand served throughout the planning horizon. This is done by relocating cars to nodes with high demand. Cars are recharged in order to make them available for future rentals. The benefit accumulated by each action may vary with time due to fluctuating demand. The second objective is to relocate the cars cost-effectively. The cost of relocation is similar to the one discussed in Section 4.1. An additional objective considered is the minimization of the idle time for the service employees. Reducing the idle time encourages employees to share the work-load.

4.3.2 Routing of Service Employees

The routing of service employees consists of sequences of travels between nodes. The travels consist of car relocations, travel with folding bikes or by public transport. Service employees are equipped with folding bikes that fit the trunk of the rental cars. It is assumed that there are always parking spaces available in parking zones. It is also assumed that service employees only travel to charging nodes when moving rental cars for recharging. When cars are fully charged, they are automatically relocated to the surrounding parking node, unassisted by service employees. Automatic relocation is supported by the fact that if customers demand cars in parking nodes, they can pick up the car from the associated charging station. The same argument goes for service employees relocating cars. Consequently, employees only relocate cars from parking nodes.

Service employees can do to two types of car relocation; either they relocate sufficiently charged cars between parking nodes, or they recharge cars in need of charging, transporting them to charging nodes. It is assumed that necessary maintenance is carried out when recharging cars. Cars in need of charging can only be moved to charging nodes with available capacity.

4.3.3 Travel Times and Time Usage

Real-time traffic data should be used when solving the DE-VReP. Due to changing traffic, the travel times between zones may vary during the planning horizon. Travel times between zones are calculated based on their geographical centers. When service employees travel without cars, they ride a bike, use public transport, or some combination of the two. The time used to relocate cars to parking nodes includes the additional time required to find an available parking spot. Similarly, the time to relocate cars to charging stations includes additional time required to start the charging process. Finally, cars currently charging are assumed to be unavailable for customers and service employees.

Chapter 5

Solution Method for the DE-VReP

The DE-VReP shares many characteristics with general dynamic routing problems, as discussed in Section 3.2. Service employees are routed when they relocate rental cars to serve customer demand. Information regarding demand and travel times in the system changes dynamically over the planning horizon, and there exist possibilities to iteratively update routes and relocation decisions based on revealed demand and travel information. Hence, the problem could be modelled as a route-based Markov decision process, MDP, as introduced by Ulmer et al. (2017). This model includes all elements of conventional MDP models, and the approach allows a formal dynamic model, that deals with sequential decision making on new information. States correspond to the current distribution of rental cars, relocation needs, and the previous route plan, while tasks correspond to the routes available for service employees. On new information, the problem transitions into a new state. This modeling technique formally links the DE-VReP to solution methods that statically deals with the optimization of tasks taken in each state. However, modeling the DE-VReP as a route-based MDP could easily lead to an uncontrollable number of states, and any realistic version of the problem would not be possible to solve to optimality within a reasonable amount of time. The Rolling Horizon approach on a static subproblem is thus deemed a better fit for the problem at hand.

This chapter presents a solution method for the DE-VReP which adopts a Rolling Horizon framework. Section 5.1 briefly introduces the components constituting the Rolling Horizon framework as well as the interaction between them. Section 5.2 discusses considerations that must be taken in the Rolling Horizon Framework, to meet the objectives of the DE-VReP.

5.1 Rolling Horizon Framework

According to Sethi and Sorger (1991), Rolling Horizon decision making is common business practice for making decisions in stochastic environments. By definition, the method involves making the most immediate decisions based on forecasts of relevant information for a certain number of periods into the future. When applied to DE-VReP, it means that an E-VReP, referred to as the *static subproblem* in this chapter, is solved iteratively as new information becomes available to the CSO. The time when the static subproblem is solved is called *decision epoch*. Each static subproblem is solved over portions of the planning horizon defined in Chapter 4. After solving a subproblem, routes, relocations and customer demand is realized. The state of the system is updated, and the subproblem is solved over again.

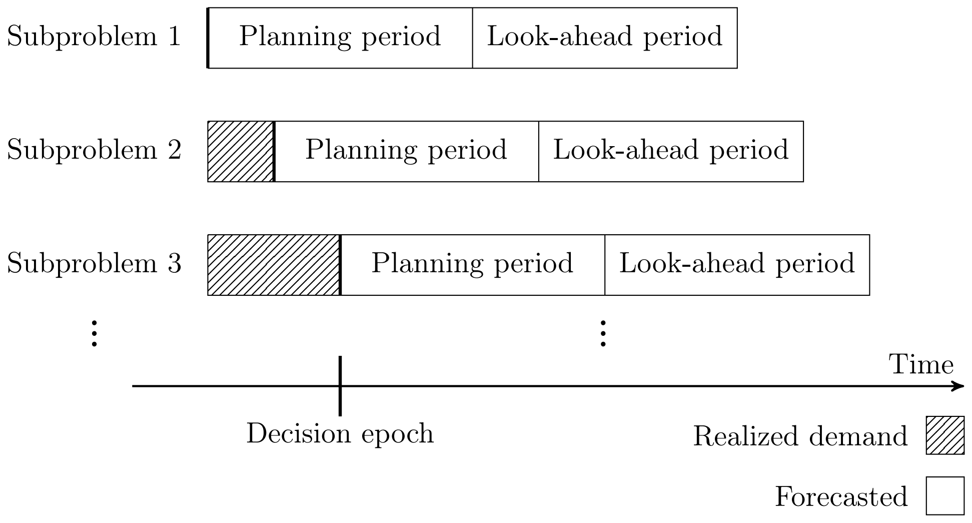


Figure 5.1: The Dynamics of The Rolling Horizon Framework

Figure 5.1 uses the terms *planning period* and *look-ahead period*. As described in Chapter 4, the planning horizon is the business hours of the carsharing system, in which the service employees relocate or recharge cars. The planning period is a subset of the planning horizon and is much shorter. The solution to every subproblem, in a Rolling Horizon framework, aims to find optimal tasks for this shorter period. There is a trade-off between choosing tasks that optimize for the short-term over the planning period, and long-term over the planning horizon. The look-ahead period partly represents this trade-off, as it is the additional time in excess of the planning period used to gather demand forecasts. These forecasts are used as input to the subproblems indicating how rental cars should be distributed at the end of the planning period. The use of demand forecasts are further elaborated in Chapter 6.

5.1.1 Components

The Rolling Horizon framework for the DE-VReP consists of the three components; the *E-VReP Solver*, the *Customer Demand* and the *Simulation Model*. The E-VReP Solver (hereby the Solver) solves the static subproblem of the DE-VReP, i.e., the E-VReP. The Customer Demand provides data on both predicted and realized customer demand. The Simulation Model simulates the business hours of the CSO by keeping track of information regarding the service employees, rental cars, and customers. The three components, as well as the basic layout of the data flow between them, is shown in Figure 5.2.

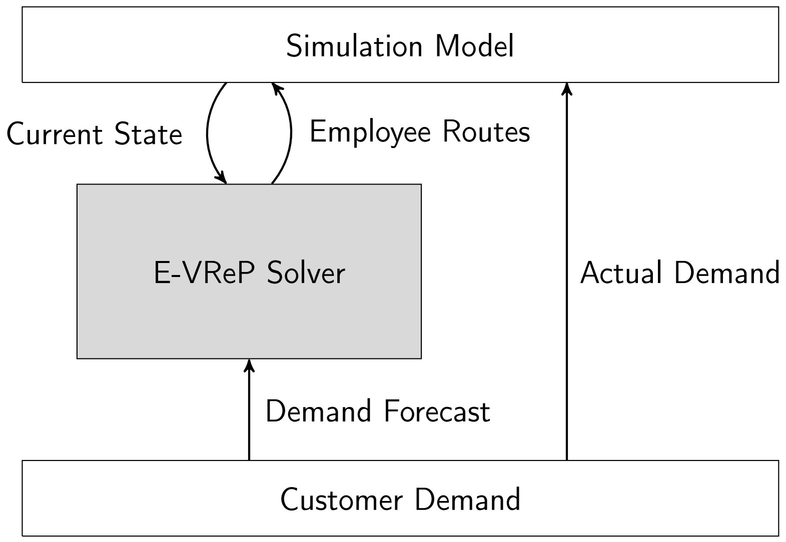


Figure 5.2: The Rolling Horizon framework components. The Simulation Model can be exchanged by a component tracking actual events in a real world scenario.

In Figure 5.2, the Simulation Model feeds the Solver with all information necessary to find optimal routes for the service employees. This data flow contains the current tasks and locations of the service employees as well as the battery level and locations of rental cars. The Solver solves the E-VReP yielding all routes and relocations to be done given the current system state. The resulting routes for each service employee are fed back to the Simulation Model which subsequently simulates both the travels of the employees and the realized customer demand. The Customer Demand delivers demand forecasts to the Solver. In addition, the Customer Demand provides realized customer demand to the Simulation Model. During the planning horizon, the Simulation Model tracks predefined performance measures set by the CSO in order to assess the system's overall performance at the end of the planning horizon.

There are two E-VReP solvers implemented in this thesis. The first is based upon solving a Mixed Integer Program (MIP) model. Chapter 7 presents the MIP formulation. The second is an implementation of an Adaptive Large Neighborhood Search (ALNS) heuristic, presented in Chapter 8. Chapters 9 and 10 present the Simulation Model and the Customer Demand, respectively.

5.1.2 Notation

The parameters specific to the Rolling Horizon framework are shown in Table 5.1. Paramters specific to the Solver are presented in Chapters 6 and 8.

Table 5.1: Parameters used in The Rolling Horizon framework

Parameters	
T_{start}	Start time planning horizon
T_{end}	End time planning horizon
$T_{increment}$	Time increments
$\overline{T}$	Length of planning period

T_{start} and T_{end} represent the start and end time of the planning horizon of the DE-VReP. $T_{increment}$ represents how often the subproblems are solved, i.e., the time difference between subsequent decision epochs from Figure 5.1. $\overline{T}$ represents the planning period. For simplicity, the time of the look-ahead period is identical to the time of the planning period.

5.1.3 Rolling Horizon Algorithm

To concretize the dynamics of the Rolling Horizon framework and the interaction between the components, a brief algorithmic model is provided in Algorithm 1. The framework starts with an initial state as defined in Section 4.2. Initially, the planning horizon is divided into time steps of length $T_{increment}$. After the initialization, the framework iterates over all of these time steps. Each time step represents a decision epoch from Figure 5.1. In each time step, lines 4 to 8, demand forecasts of the current planning period and the look-ahead period are fed to the Solver. The Solver then solves the static subproblem to find the desired tasks to be performed by the service employees. When the Solver is done, the Simulation Model updates the system state with the actions done by the service employees and customers.

Algorithm 1: Rolling Horizon

```
1 State = Initial State
2 $t = T_{start}$
3 while $t \leq T_{end}$ do
4     DemandForecast = CustomerDemand.getForecast($t, t + 2 \cdot \overline{T}$)
5     EmployeeRoutes = Solver.solve(State, DemandForecast)
6     ActualDemand = CustomerDemand.getDemand($t, t + 2\overline{T}$)
7     State = SimulationModel(t,EmployeeRoutes,ActualDemand)
8     $t = t + T_{increment}$
9 end
```

5.2 Objective Function Considerations

When solving different instances of the E-VReP iteratively in a Rolling Horizon framework, it is essential that the objectives used in the Solver coincide with the overall objectives of the DE-VReP, even though they may not be the same. There is no obvious answer to which objectives that approximate the objectives of the DE-VReP. However, obtaining a strong connection between the components in the Rolling Horizon framework is essential. For instance, for the Solver to produce routes that in expectation maximize the customer demand served, the Customer Demand component must be able to produce accurate ideal states that coincides with the same goal.

Chapter 6

Problem Description of the E-VReP

This chapter introduces the main characteristics of the *Electric Vehicle Relocation Problem* (E-VReP). The E-VReP is solved by the Solver from Chapter 5, at each decision epoch in the Rolling Horizon framework. A problem overview is given in Section 6.1. Section 6.2 introduces relevant definitions, before Section 6.3 presents objectives, relocation considerations, and time usage on a more detailed level. Finally, Section 6.4 shows an example of a solution to the E-VReP.

6.1 Problem Overview

The E-VReP shares many characteristics with the DE-VReP from Chapter 4. The decision maker, decisions made, costs, booking scheme, and travel configurations, are all the same. The main differences to the DE-VReP are concerned with the use of the planning period, how the problem receives new information, and the objectives used.

The objectives of the E-VReP are closely linked to those of the DE-VReP. The primary objectives of the E-VReP are to maximize expected demand served and the number of cars recharged, in a cost-effective way. Expected demand served is maximized by minimizing the deviation from an estimated optimal car distribution at the end of the planning period. This distribution is calculated based on the expected demand in the look-ahead period introduced in Chapter 5, i.e., the planning period of the next subproblem. The actual benefits and costs incurred by the decisions made can, however, only be approximated. For instance, the benefit of recharging of cars is only observable after the current *planning period*. Likewise, the expected demand served is only approximated. The costs considered in the E-VReP are related to the wear, toll, and electricity costs of relocating cars. An overview of the E-VReP is given in Figure 6.1.

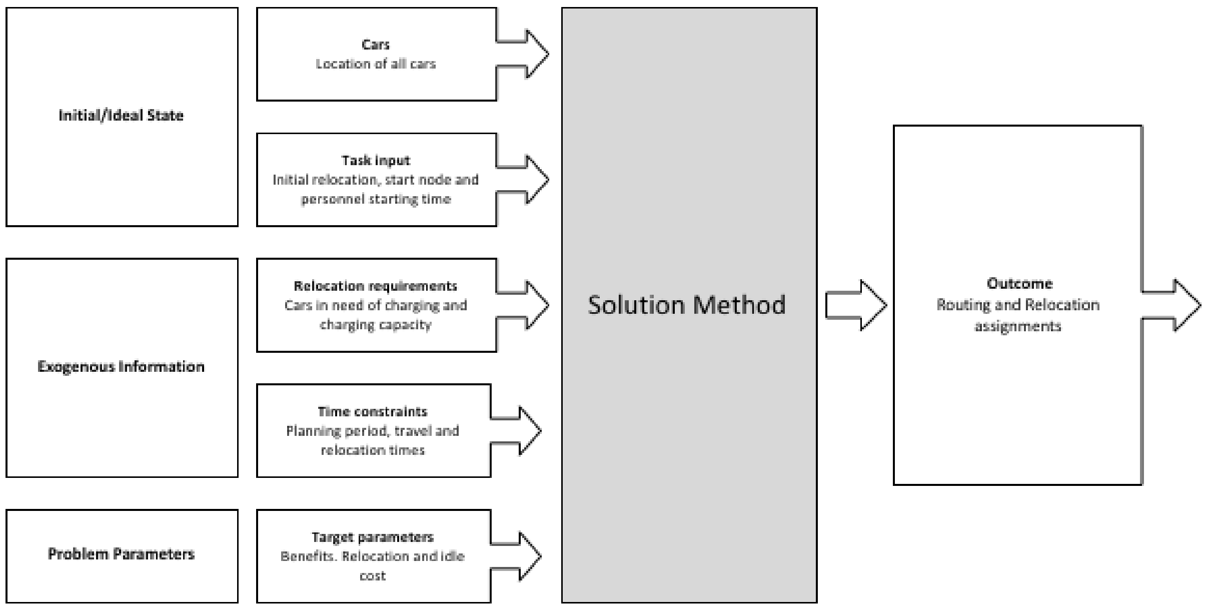

Initial/Ideal State
Exogenous Information
Problem Parameters
Cars
Location of all cars
Task input
Initial relocation, start node and personnel starting time
Relocation requirements
Cars in need of charging and charging capacity
Time constraints
Planning period, travel and relocation times
Target parameters
Benefits. Relocation and idle cost
Solution Method
Outcome
Routing and Relocation assignments

Figure 6.1: E-VReP Problem Overview The model receives information before determining relocation routes for the following planning period. The ideal state is given as input. The ideal state is calculated based on the expected demand of the next planning period.

6.2 Definitions

The *initial state* of the problem defines the distribution of rental cars among all nodes. It also gives information about the position, destination, current task, and remaining travel time for all service employees. Service employees may, at the beginning of a planning period, be in the process of relocating a rental car as a result of unfinished assignments. Exogenous information includes battery levels for all rental cars, capacities of charging nodes and time remaining for the cars currently being charged.

The *ideal state* refers to the estimated optimal distribution of rental cars in the parking nodes at the end of the planning period. The deviation from the ideal state is the distribution of rental cars at the end of the planning period versus the ideal state. This deviation is calculated from the initial state, adjusted by the relocation done, and compared with the ideal state.

6.3 Problem Specification

This section introduces the objective function, the routing of service employees, the possible types of relocation and time usage in the E-VReP. Information already introduced in Chapter 4 is excluded in this section.

6.3.1 Objective

The objective of the E-VReP is to maximize expected demand served, and the number of cars recharged. This should be done cost-effectively. The objectives are designed to encourage behavior that is beneficial for periods beyond the current planning period. A surplus of cars in a node, compared to the ideal state, is not considered beneficial. Cars are likely to be picked up from surplus nodes in order to meet the ideal state in other nodes. The cost of negative deviation from the ideal state reflects the loss of demand incurred by having too few cars in a node. Relocation costs are equivalent to those discussed in Section 4.1. Time usage outside the planning period is allowed to some extent. However, the time used outside the planning period by service employees is penalized proportionally to the excess time. Additionally, a possible objective is to minimize the total time used by each service employee. This objective encourages service employees to share their workload.

In summary, the primary objectives of the E-VReP are to maximize expected demand served, and the number of cars recharged.

6.3.2 Routing

Routing in the E-VReP adheres to the rules as those from Section 4.3.

6.3.3 Relocation of Rental Cars

As for DE-VReP, the E-VReP deals with two types of assignments. Service employees may relocate cars between parking nodes, or from parking nodes to charging nodes. Basic maintenance is performed while relocating to charging nodes. When rental cars charge, they are considered unavailable for the rest of the planning period.

6.3.4 Travel Times and Time Usage

When deciding the length of the planning period, several aspects are taken into consideration. The time period must be long enough to permit several relocations for each service employee. However, the time should not be too long as the relocations can become aggravating due to unforeseen events during the planning period. All service employees have a remaining travel time associated with them, due to the fact that they may initially be traveling. Service employees are considered unavailable until they finish their current task. The remaining time before a service employee reaches the first node is known in advance of solving the E-VReP.

There is also a remaining charging time for the rental cars that will be fully charged during the planning period. These fully charged rental cars will be available for both customers and service employees when fully charged.

6.4 Solution Example to the E-VReP

This section presents a solution example to the E-VReP. A summary of the solution example can be found in Appendix C.

Figure 6.2 shows the initial state. For simplicity, each node is represented in the Cartesian plane with Euclidean distances between them. Traveling using a rental car is twice as fast as using a bike in this example. There are six parking nodes numbered 1, 2, 3, 4, 5 and 6. Nodes 7 and 8 are charging nodes located within parking nodes 1 and 6 respectively. Each parking node has an ideal state of sufficiently charged rental cars indicated in the upper right corner of each node. Nodes containing a charging station does also have a capacity, indicated in the upper right corner. The capacity indicates the maximum number of cars that can charge at the same time. Orange, red and green cars in the bottom of the nodes indicate cars charging, cars in need of charging and sufficiently charged cars, respectively.

Service employees are illustrated using black cars, bikes and person figures. A unique number above their icon identifies employees. The employees might be in-between two nodes at time step 0. This is due to unfinished tasks from the previous subproblem.

In Figure 6.2 employee 1 is traveling to parking node 5 with a car. The other two employees are idle in parking nodes 1 and 3 respectively.

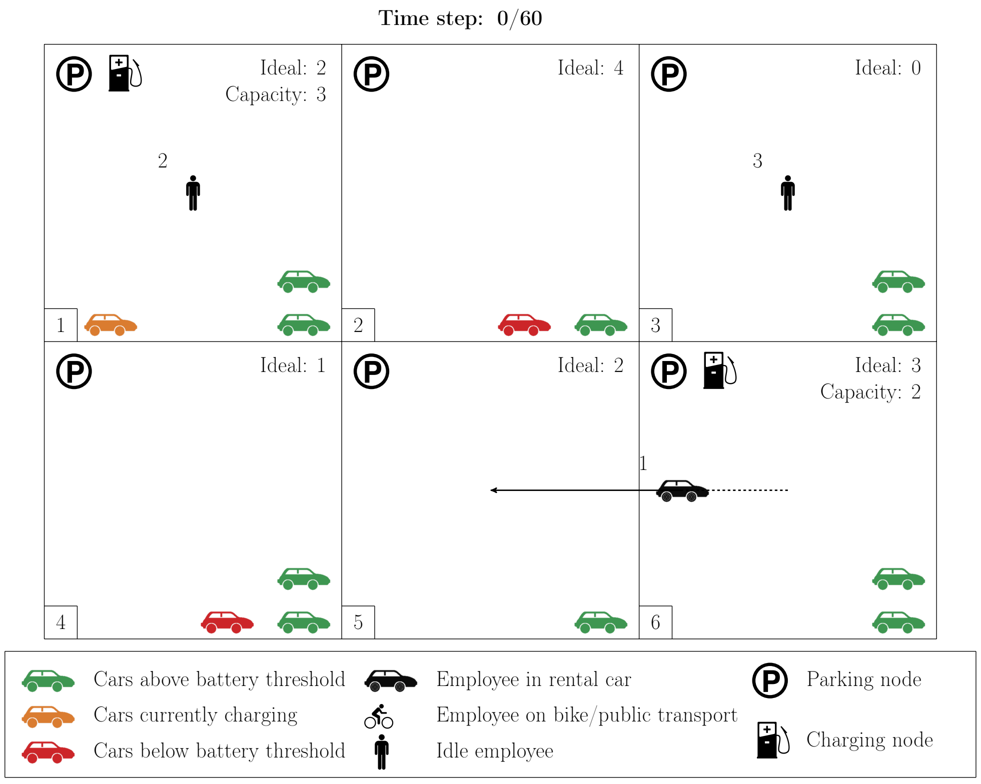


Figure 6.2: Initial state

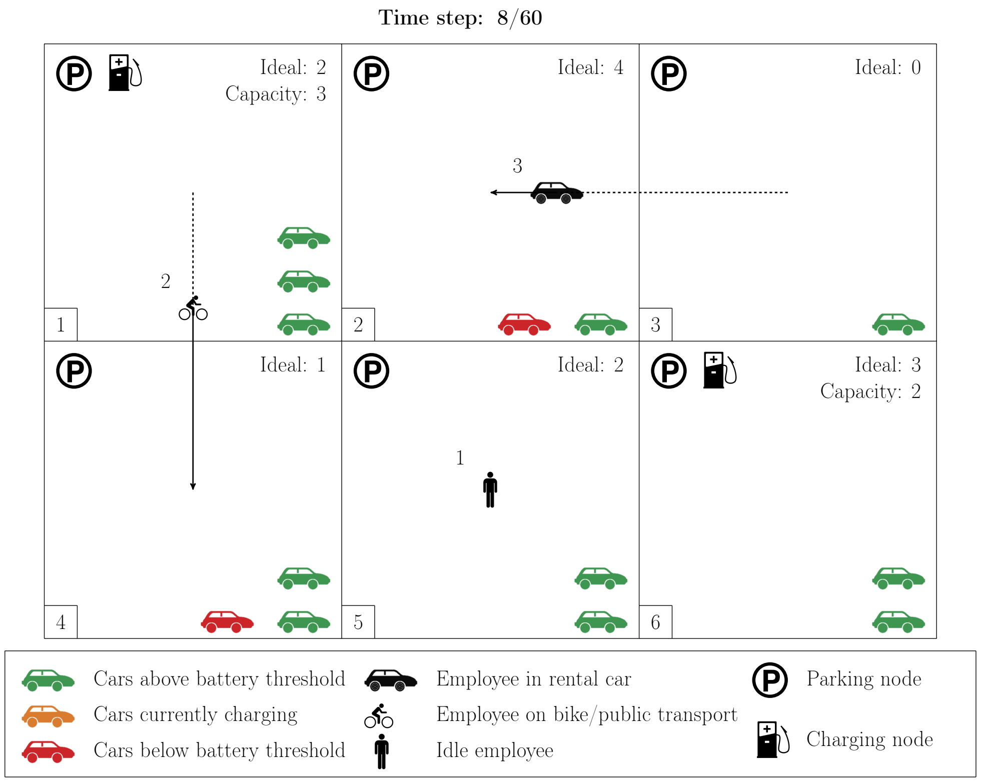


Figure 6.3: Intermediate state 1

In Figure 6.3 employee 1 has arrived in node 5 with a car which is indicated by the extra green car in node 5. Employee 2 is biking from node 1 to node 4 while employee 3 relocates a car from node 3 to node 2.

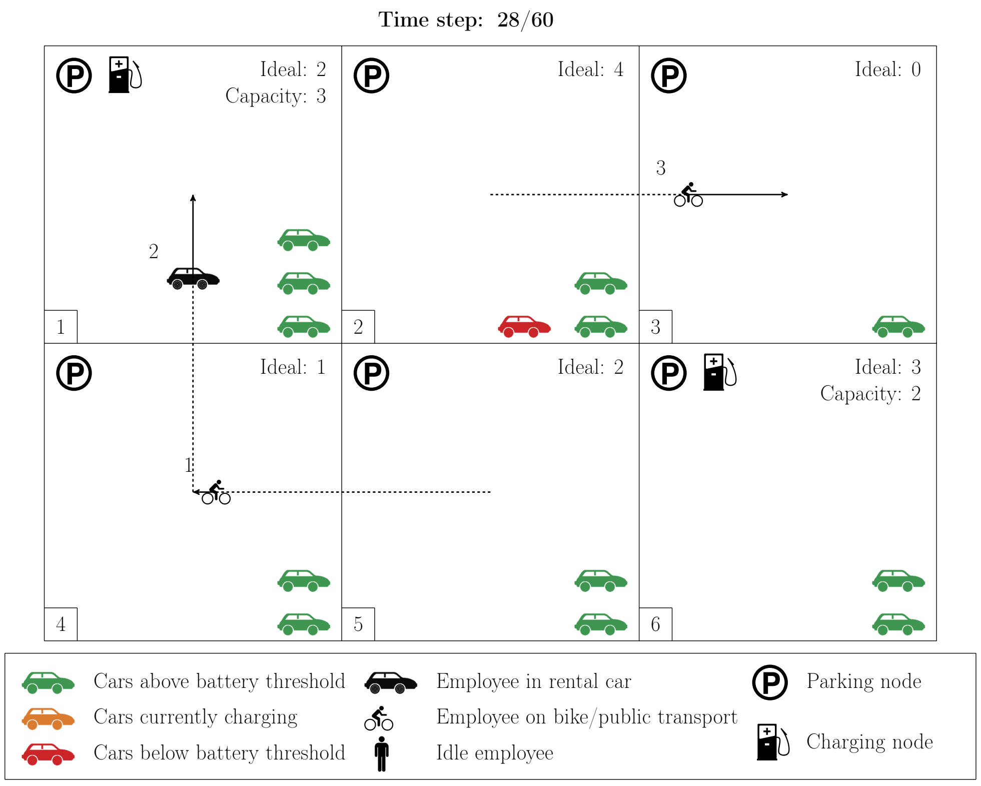


Figure 6.4: Intermediate state 2

Figure 6.4 illustrates the situation after 28 time-steps. Employee 1 is biking between parking node 5 and 4. Employee 2 has, since the last snapshot, reached node 4, and started to relocate a car in need of charging to the charging node inside parking node 1. Employee 3 has delivered a car in parking node 2 and is traveling to parking node 3 to relocate a car from node 3.

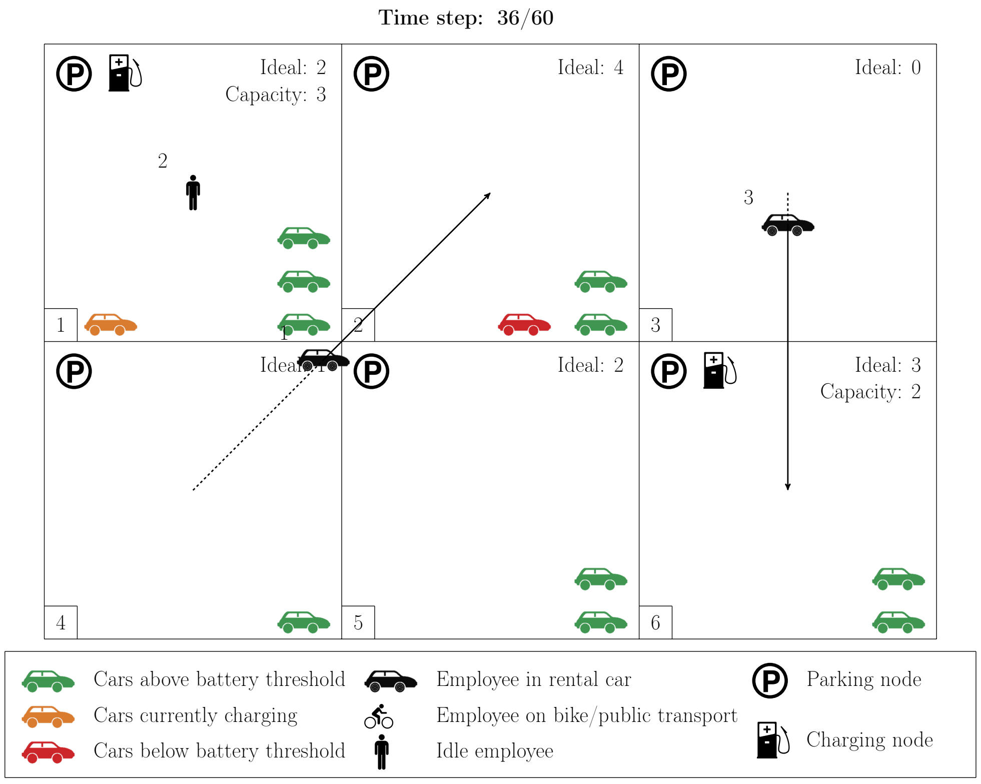


Figure 6.5: Intermediate state 3

In Figure 6.5 employees 1 and 3 relocate cars from node 4 to node 2 and from node 3 to node 6, respectively. Employee 2 has delivered a car for charging in the charging node inside parking node 1.

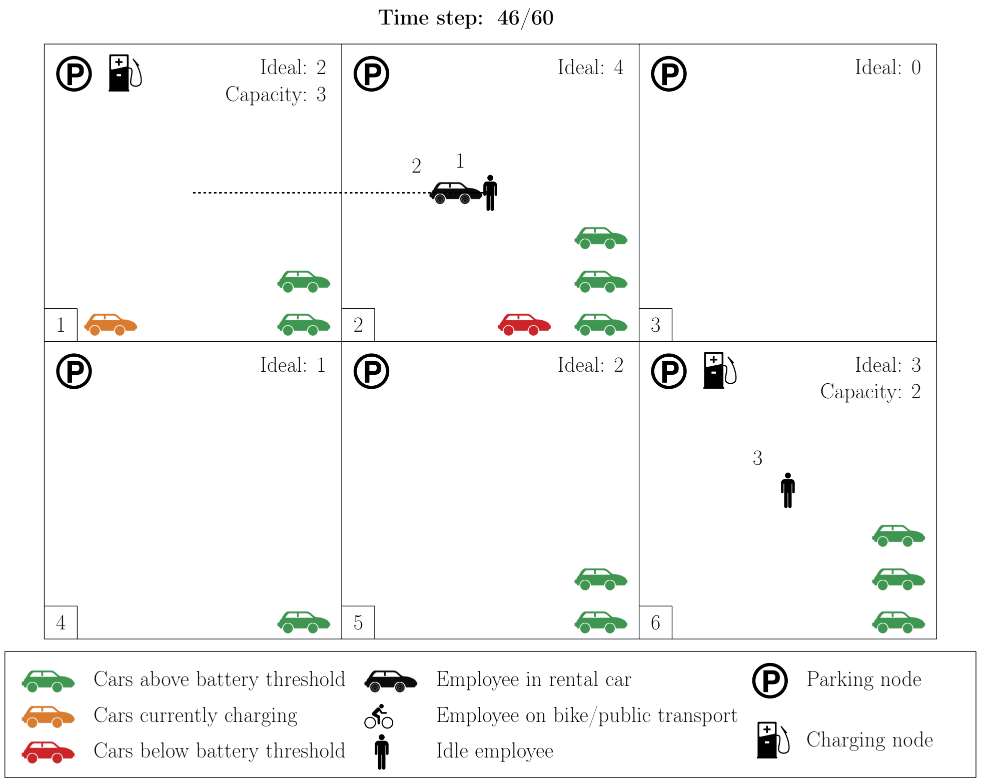


Figure 6.6: Intermediate state 4

Both employee 1 and 3 have reached their destinations in Figure 6.6. Employee 2 relocates a car from node 1 to node 2.

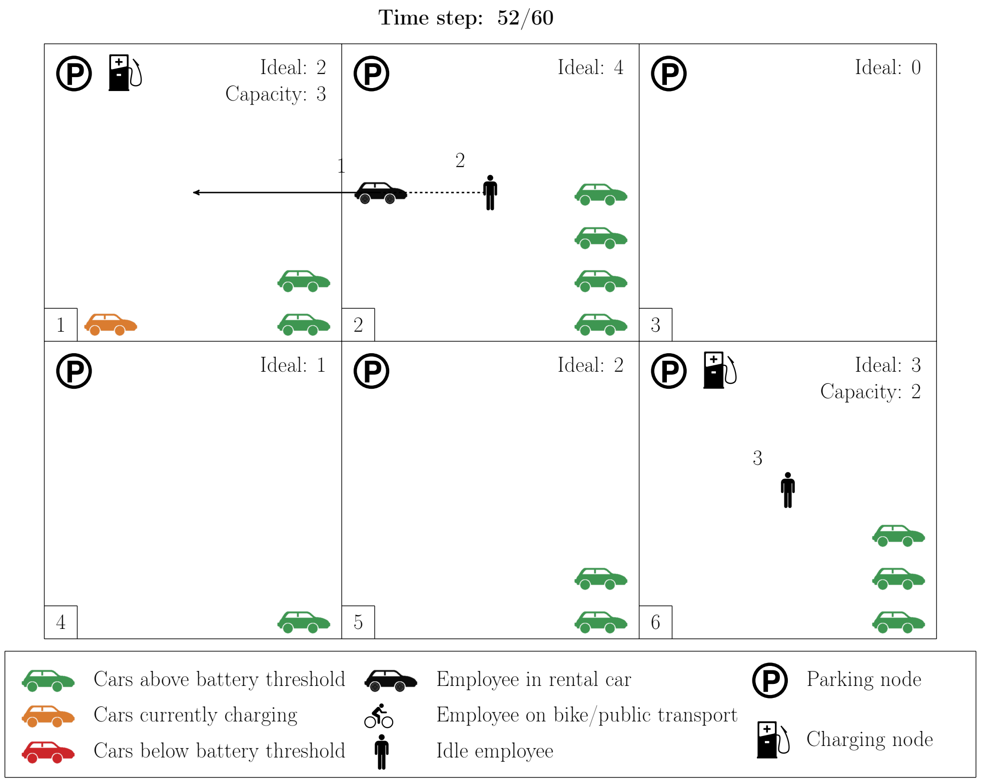


Figure 6.7: Intermediate state 5

In Figure 6.7 both employees 2 and 3 are idle. Employee 1 is moving a car in need of charging from parking node 2 to the charging node associated with parking node 1.

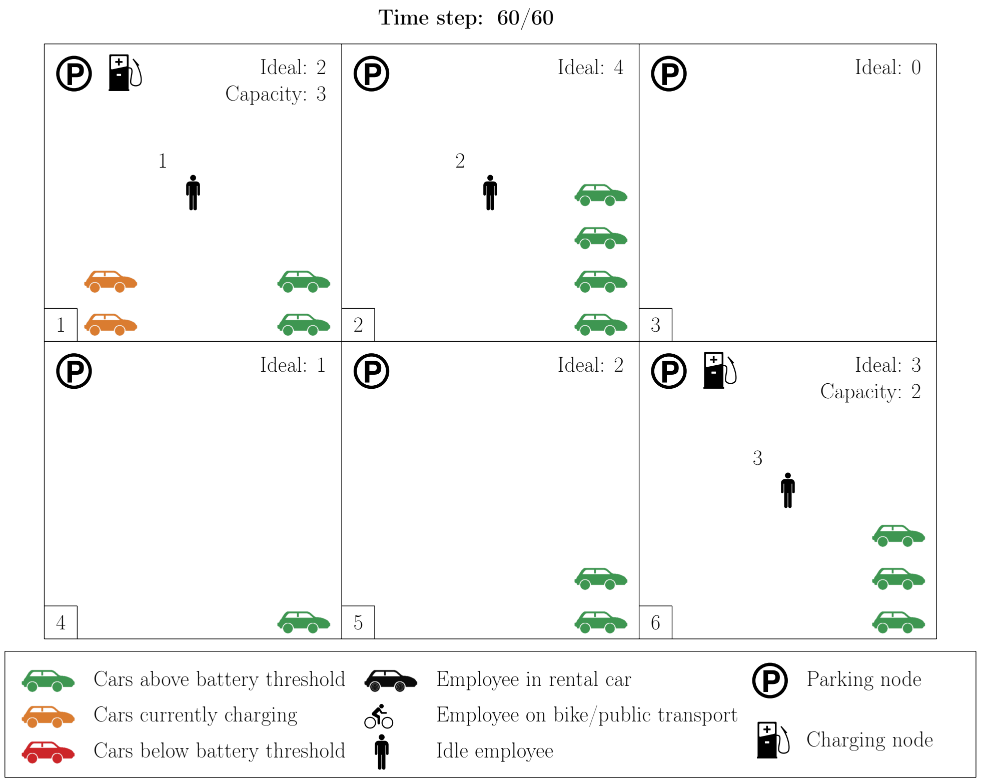


Figure 6.8: Final state

Figure 6.8 shows the final state of the system. There are no cars in need of charging, and all ideal states have been met in all parking nodes.

Figure 6.9 provides an overview of the routes traveled by the three employees. The text above the arcs indicates means of transportation. Since employee 1 initially is between two nodes, the first node on the route of employee 1 is indicated by a question mark. This implies that we do not know where the employee is coming from, just where the employee is going . The location where the employees come from is not valuable in order to find the next set of optimal routes for the employees.

$$\begin{tikzcd}
\text{Employee 1} & ? \arrow[r, "\text{car}"] & 5 \arrow[r, "\text{bike}"] & 4 \arrow[r, "\text{car}"] & 2 \arrow[r, "\text{car}"] & 1 \\
\text{Employee 2} & 1 \arrow[r, "\text{bike}"] & 4 \arrow[r, "\text{car}"] & 1 \arrow[r, "\text{car}"] & 2 & \\
\text{Employee 3} & 3 \arrow[r, "\text{car}"] & 2 \arrow[r, "\text{bike}"] & 3 \arrow[r, "\text{car}"] & 1 &
\end{tikzcd}$$

Figure 6.9: Routes traveled by the service employees. Any visit to a charging node ha been exchanged with the associated parking node number.

Chapter 7

A MIP Model for the E-VReP

This chapter formulates a Mixed Integer Program (MIP) for the E-VReP. The E-VReP combines features of the *VRP* and *the pickup and delivery problem*, and involves routing of service employees traveling by car, bike or public transport. The service employees pick up and deliver cars to relocate them between locations in the operating area. The problem is *open-ended* because service employees may individually start and end at different locations. Each location may also be visited several times, making the E-VReP similar to a pickup and delivery problem with split deliveries.

Section 7.1 discusses the MIP design choices. Section 7.2 presents additional assumptions for the MIP that were not introduced in Chapter 6. Section 7.3 defines the notation used, and Section 7.4 presents the mathematical formulation of the problem.

7.1 MIP Design Considerations

There exists an important simplification to the E-VReP; each service employee can only move one car at a time. Hence, a possible modeling approach is to treat each combination of pickup and delivery nodes as individual tasks referred to as *car-moves*. Service employees should only consider car-moves that are beneficial in terms of meeting demand or charging cars. An approach is, therefore, to only include car-moves which contribute to meeting these objectives. In other words, only car-moves from pickup to delivery nodes are considered. This results in at most $\mathcal{O}(|\mathcal{N}|^2)$ car-moves, where $\mathcal{N}$ is the set of all nodes. The set of car-moves is denoted $\mathcal{R}$. The goal of the MIP is to decide the optimal subset of car-moves, and how to distribute and order them among the service employees. Using this concept of car-moves, two MIP models are considered.

The first MIP is an arc-flow model, referred to as the *Flow-based MIP*, commonly used to solve VRPs. Arcs are defined between all pairs of car-moves. The goal is to decide the optimal flow over all arcs while making sure that each car-move only is included once. The decision variables in this MIP, defined as x_{rvk}, indicate travel by service employee $k \in \mathcal{K}$ from car-move $r \in \mathcal{R}$ to car-move $v \in \mathcal{R}$. The total number of variables would be in the order of $\mathcal{O}(|\mathcal{K}| \cdot |\mathcal{R}|^2)$.

The second MIP, referred to as the *Task-based MIP*, assigns car-moves to service employees and decides the order in which they are carried out. Each employee has a set of tasks $\mathcal{M}$ which can be assigned to car-moves. Each task has a number indicating precedence, and tasks are performed in ascending order. Hence, the decision variables in this MIP, defined as x_{krm}, indicate if a service employee $k \in \mathcal{K}$ relocates car-move $r \in \mathcal{R}$ as task number $m \in \mathcal{M}$. With a maximum number of tasks that can be done during a planning period, the total number of variables equals $\mathcal{O}(|\mathcal{K}| \cdot |\mathcal{R}|)$.

Chapter 11 compares the performance of both MIP models. In terms of solution quality and computation time required, the Task-based model outperforms the Flow-based model. Therefore, the Task-based model is presented in this chapter. The Flow-based MIP formulation is presented in Appendix B.

7.2 Assumptions

This section presents key assumptions not introduced in Chapter 6. The assumptions are divided into three parts; nodes and states, routing and relocation, and time usage.

7.2.1 Nodes and States

The problem formulation includes a set of charging nodes and a set of parking nodes. Each charging node has a maximum capacity of cars. Each parking node is associated with an ideal number of parked cars, referred to as the ideal state. Each parking node may be a surplus or a deficit node, determined by its number of available rental cars compared to the ideal state. Service employees relocate cars from surplus nodes to deficit nodes. In addition, service employees relocate cars in need of charging from parking to charging nodes.

7.2.2 Service Employees

The number of service employees is considered fixed, and the service employees are considered homogeneous. Car-moves assigned to service employees are always completed and not canceled midway.

7.2.3 Time Use

Time is continuous and monotonically increasing for every task done by the service employees. Employees are assumed to use the fastest means of transport available, either folding bikes or public transport, when traveling between car relocations.

7.3 Notation

This section introduces all sets, indices, parameters and variables used in the mathematical formulation. A summary can be found in Tables A.1, A.2, A.3 and A.4. Sets and indices are introduced in Section 7.3.1, parameters in Section 7.3.2 and variables in Section 7.3.3

7.3.1 Sets and Indices

The following section introduces set and indices for nodes, car-moves, and service employees used in the mathematical formulation.

Nodes

The operating area of the CSO is divided into non-overlapping zones represented as nodes. The set of all nodes used in the mathematical formulation is denoted $\mathcal{N}$, referenced by the indices i, j. $\mathcal{N}$ is divided into two disjoint subsets; the set of charging nodes $\mathcal{N}^C$ and the set of parking nodes $\mathcal{N}^P$. Nodes in $\mathcal{N}^C$ corresponds to charging stations, while the nodes in $\mathcal{N}^P$ are disjoint areas available for parking. The parking nodes are divided into two categories; the surplus nodes $\mathcal{N}^{P+}$ and the deficit nodes $\mathcal{N}^{P-}$. The surplus and deficit nodes have a positive and negative deviation from the ideal state, respectively. There is also an additional set, $\mathcal{N}^{PC}$, which contains the parking nodes with cars in need of charging. Note that $\mathcal{N}^{PC}$ may be disjoint from both $\mathcal{N}^{P+}$ and $\mathcal{N}^{P-}$ if the node satisfies the ideal state. See Figure 7.1 for an illustration.

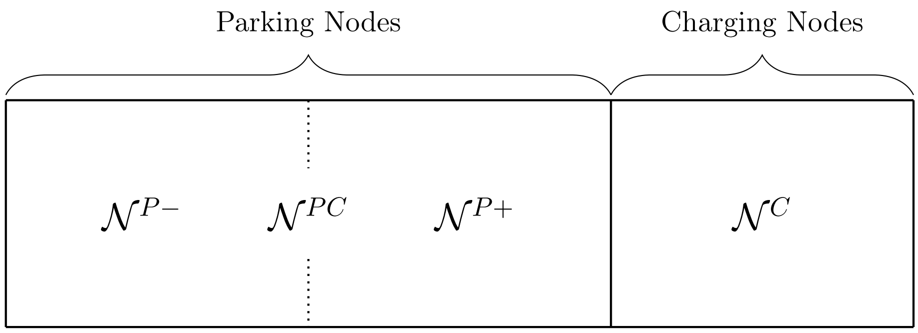


Figure 7.1: Parking and charging nodes

Car-Moves

For sufficiently charged cars, car-moves are defined from surplus nodes in $\mathcal{N}^{P+}$ to deficit nodes in $\mathcal{N}^{P-}$. For cars in need of charging, car-moves are defined from parking nodes in $\mathcal{N}^{PC}$ to charging nodes in $\mathcal{N}^C$. For simplicity, car-moves to parking and charging nodes are called *parking-moves* and *charging-moves*, respectively. Parking-moves and charging-moves are illustrated in Figure 7.2. The set of all car-moves is denoted $\mathcal{R}$, indexed by both r and v. The origin node $o(r)$ of a car-move $r \in \mathcal{R}$ is always a parking node. The destination node $d(r)$ is either a parking node or a charging node.

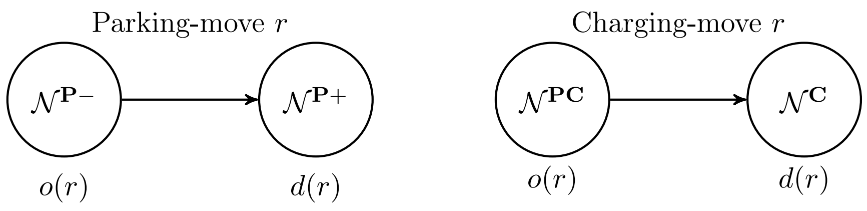

Parking-move r
$\mathcal{N}^{\mathbf{P-}}$
$\mathcal{N}^{\mathbf{P+}}$
$o(r)$
$d(r)$
Charging-move r
$\mathcal{N}^{\mathbf{PC}}$
$\mathcal{N}^{\mathbf{C}}$
$o(r)$
$d(r)$

Figure 7.2: Car-moves divided into parking-moves and charging-moves

A car-move r is associated with a car c among the set of cars $\mathcal{C}$ subject to relocation during the planning period. Cars subject to relocation are those currently in surplus nodes or those in need of charging. Incoming cars to surplus nodes are also subject to relocation. However, it is not necessary to include car-moves for more cars than the number of cars in excess of the ideal state. In other words, if a node has a surplus of one car, it is sufficient to define car-moves for only one car from that node. In addition, according to Proposition 1, defining car-moves from other nodes than the cars' origin nodes, cannot improve the solution quality. Hence, it is sufficient to only define car-moves from cars' origin nodes.

Proposition 1. *In a given decision epoch, relocating sufficiently charged cars more than once cannot improve the solution quality* [1]

Proof. The proof considers the simplest cases of relocating cars multiple times; using one and two service employees. The cases using more than two employees can be constructed combining the cases of one and two service employees. Each case presents a way of relocating more efficiently than relocating a car multiple times while achieving the same number of relocations. The proof is based on the triangular inequality and that travel times T_{ij} represent the shortest possible travel time between nodes $i, j \in \mathcal{N}$.

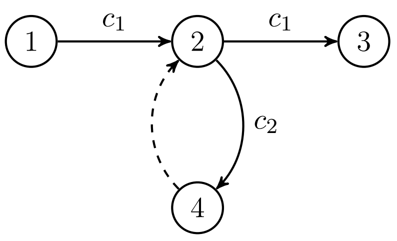
1. *Single employee.* A service employee relocates car c_1 from node 1 to node 2, before relocating a second car c_2 from node 2 to node 4. The employee then returns to node 2 to relocate the car c_1 to node 3. Due to the triangular inequality and that travel times T_{ij} represent shortest paths, the following will always hold:

$$T_{12} + T_{24} + T_{42} \geq T_{14} + T_{42} \tag{7.1}$$

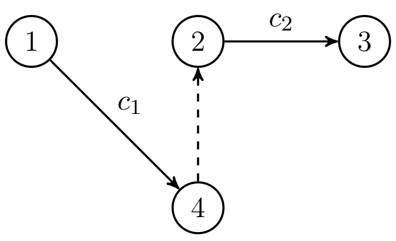
   This implies that relocating the first car directly to node 4 and relocating the second car from 2 to 3 dominates relocating a car twice. The same argument holds if node 4 is replaced with a series of relocations. See Figure 7.3 for an illustration.

2. *Two employees.* A service employee relocates car c_1 from node 5 to node 3 and bicycles to node 1. Another service employee bicycles from node 4 to node 3, before relocating the car c_1 to node 2. According to the triangular inequality, it will always be beneficial for the first service employee to relocate the car directly to node 2, and for the second service employee to bicycle directly to node 1. See Figure 7.4 for an illustration.

In each of the cases above, relocating cars more than once achieved the same number of relocations, but slower. Hence, relocating cars multiple times is not beneficial. ■



(a) Relocating cars more than once



(b) Relocating cars once

Figure 7.3: Single service employee

[1]This does not necessarily hold for cars in need of charging. However, it is assumed that cars in need of charging are moved directly to charging nodes.

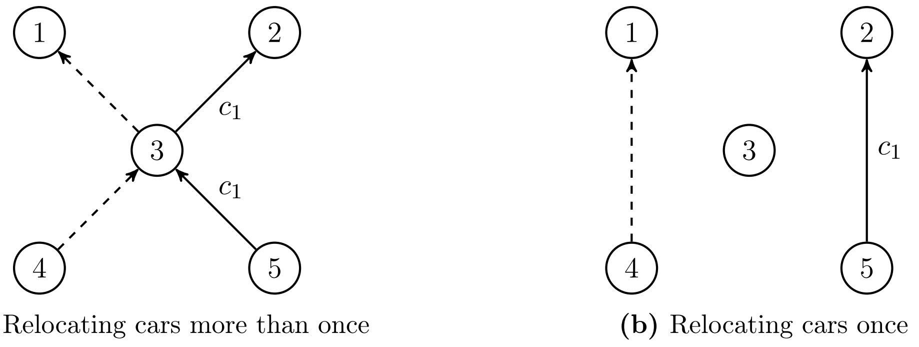


Figure 7.4: Two service employees

Additional sets related to car-moves are defined to simplify the formulation. For every deficit node $i \in \mathcal{N}^{P-}$, there is a set of car-moves that have that node as its destination $d(r)$. This set is denoted $\mathcal{R}_i^D$. Similarly, sets of car-moves R_i^C, that originates in node $i \in \mathcal{N}^{PC}$ and ends in node $j \in \mathcal{N}^C$, are defined. Finally, a set of car-moves which has a charging node as destination node are defined $\mathcal{R}_i^{DC}$ for all $i \in \mathcal{N}^C$.

Service Employees and Tasks

Each service employee $k \in \mathcal{K}$ has a set of possible tasks $\mathcal{M}$ which can be assigned to car-moves. The task number indicates precedence, in ascending order, i.e, task number one is assigned before task number two. Not all tasks need to be assigned car-moves. The cardinality of $\mathcal{M}$ decides the maximum number of car-moves service employees can do. If a service employee $k \in \mathcal{K}$ relocates a car to the destination specified by a car-move $r \in \mathcal{R}$, the employee k is said to relocate the car-move r for simplicity. Service employees' individual start nodes are denoted $o(k)$ for all $k \in \mathcal{K}$.

7.3.2 Parameters

The following section introduces parameters for the objective, time and states used in the MIP model.

Objective Function Parameters

The objective functions seek to maximize benefit. There are five parameters. C^D is the benefit per unit of ideal state met for nodes $i \in \mathcal{N}^{P-}$ at the end of the planning period. C^{Ch} is the benefit per car charged. C^{ET} is the cost per time unit used to relocate cars outside the planning period. The cost of relocation is proportional to the relocation time of the car-moves, penalized at a rate of C^R per time unit. This cost is introduced to reduce the wear, toll and electricity use, caused by the relocations. Finally, the cost of time C^T is introduced, to reduce the idle time of the service employees.

Time Usage

The relocation time needed to perform car-move $r \in \mathcal{R}$ is denoted T_r^H. This time includes transport and parking time, as well as time for basic maintenance. Since cars may be in transit at the beginning of the planning period, each car-move r is associated with T_r^S indicating the earliest time a car-move can be performed. The same case applies to the service employees. The earliest start time for a service employee $k \in \mathcal{K}$ is denoted T_k^S.

Travel times between all nodes $i, j \in \mathcal{N}$, using public transport or bicycle, are represented by T_{ij}. Service employees will use these means when traveling between car-moves $r, v \in \mathcal{R}$, i.e., between $d(r)$ and $o(v)$. All tasks performed by the service employees should be carried out during the planning period, $\overline{T}$. However, some overtime is allowed. This additional time is denoted $\overline{T}^L$.

States and Capacities

One of the objectives of the MIP model is to approach the ideal state in the parking nodes at the end of the planning period. Thus, the initial deficit of cars in every parking node $i \in \mathcal{N}^{P-}$, compared to the ideal state, is denoted by S_i^{0-}. Calculation of S_i^{0-} takes the expected demand of the current planning period into account. The second objective is to charge cars in need of charging. For parking nodes $i \in \mathcal{N}^{PC}$, S_i^C denotes the initial number of cars in node i that requires charging. Every charging node $i \in \mathcal{N}^C$ has an available capacity of $\mathcal{N}_i^{CS}$.

7.3.3 Variables

The following sections introduce the variables used in the formulation of the MIP model. The variables are divided into three categories; relocation, time tracking and state variables.

Relocation Variables

The variables x_{krm} indicate that a service employee $k \in \mathcal{K}$, relocates car-move $r \in \mathcal{R}$, as its task number $m \in \mathcal{M}$. The route of the service employee is derived by looking at each task in order. A service employee performing car-moves $1, 2, 3$ is illustrated in Figure 7.5.

Time Variables

Time at the beginning of each task $m \in \mathcal{M}$, for every service employee $k \in \mathcal{K}$, is tracked by the continuous variables t_{km}. Since overtime is allowed, the total deviation from the planning period $\overline{T}$ is tracked by the continuous variables t_k^+ and t_k^-. t_k^+ and t_k^- track the time used in excess and in short of $\overline{T}$, respectively.

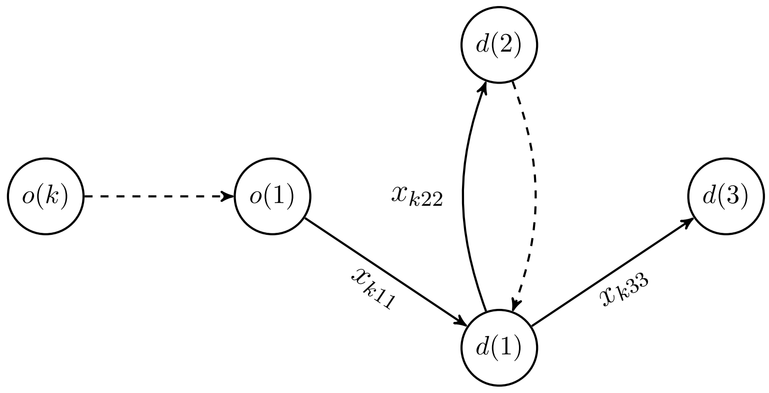


Figure 7.5: Service employee k traveling from origin node $o(k)$ and relocate car-moves $1, 2, 3$ in ascending order. Note that $d(1) = o(2) = o(3)$. The service employee bicycles from $o(k)$ to $o(1)$ and from $d(2)$ to $d(1)$, indicated by the dashed lines.

State Variables

To track the number of deficit cars in a parking node $i \in \mathcal{N}^{P-}$ at the end of the planning period, the variables s_i^- are introduced. Similarly, the variables s_i^C track the remaining cars in need of charging in node $i \in \mathcal{N}^{PC}$.

7.4 Formulation

This section formulates the E-VReP as a Mixed Integer Program (MIP). A summary of the introduced MIP can be found in Appendix A. The objective function is introduced in Section 7.4.1. Sections 7.4.2 - 7.4.5 introduce the linear constraints of the MIP as well as the domain definitions for the variables.

7.4.1 Objective Function

The Objective Function (7.2) seeks to maximize the sum of five terms. The first term is the benefit of meeting the ideal state. This is equivalent to minimizing the deviation from the ideal state. The second term is the benefit of charging cars. The third term minimizes total time used by service employees to reduce the idle time between relocations. The fourth term penalizes each time unit used outside the planning period. Finally, the fifth term penalizes relocations according to travel time used.

$$\begin{aligned}\max z = &\sum_{i \in \mathcal{N}^{P-}} C^D(S_i^{0-} - s_i^-) + \sum_{i \in \mathcal{N}^{PC}} C^{Ch}(S_i^C - s_i^C) - \sum_{k \in \mathcal{K}} C^T t_{k|M|} \\ &- \sum_{k \in \mathcal{K}} C^{ET} t_k^+ - \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} \sum_{m \in \mathcal{M}} C^R T_r^H x_{krm} \qquad (7.2)\end{aligned}$$

7.4.2 Relocation of Rental Vehicles

Constraints (7.3) ensure that each car is moved at most once. Constraints (7.4) ensure that tasks can at most be assigned one car-move. Finally, constraints (7.5) are introduced so that each task is performed in ascending order.

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_c}\sum_{m\in\mathcal{M}} x_{krm} \leq 1 \qquad c\in\mathcal{C} \tag{7.3}$$

$$\sum_{r\in\mathcal{R}} x_{krm} \leq 1 \qquad k\in\mathcal{K}, m\in\mathcal{M} \tag{7.4}$$

$$\sum_{r\in\mathcal{R}} x_{kr(m+1)} \leq \sum_{r\in\mathcal{R}} x_{krm} \qquad k\in\mathcal{K}, m\in\mathcal{M}\setminus\{|\mathcal{M}|\} \tag{7.5}$$

7.4.3 Node State Balancing

The following section introduces constraints regarding changes to the state of parking nodes and charging nodes, respectively.

State Balance in Parking Nodes for Sufficiently Charged Rental Cars

Constraints (7.6) track the total number of cars relocated to each deficit parking node. They also ensure that the maximum number of cars relocated to a deficit node $i\in\mathcal{N}^{\mathcal{P}-}$, is the total deficit.

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_i^D}\sum_{m\in\mathcal{M}} x_{krm} + s_i^- = S_i^{0-} \qquad i\in\mathcal{N}^{P-} \tag{7.6}$$

State Balance in Parking Nodes for Cars in Need of Charging

Constraints (7.7) calculate the remaining cars in a parking node, in need of charging, at the end of the planning period.

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_i^C}\sum_{m\in\mathcal{M}} x_{krm} + s_i^C = S_i^C \qquad i\in\mathcal{N}^{PC} \tag{7.7}$$

$$\tag{7.8}$$

State Balance in Charging Nodes

Constraints (7.9) ensure that the charging station capacities are not broken during the planning period.

$$\sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_i^{DC}} \sum_{m \in \mathcal{M}} x_{krm} \leq N_i^{CS} \qquad i \in \mathcal{N}^C \tag{7.9}$$

7.4.4 Time Tracking of Node Visits

The following sections define constraints concerned with time usage of the service employees.

Time Usage for Routing of Service Employees

Constraints (7.10) ensure that time increases for each task carried out, based on the relocation time of the previous task and the travel time to the next pickup. Constraints (7.11) enforce that no task can be done before a task with a lower number. Constraints (7.12) ensure that no car-move is carried out before its earliest start time. Constraints (7.13) force employees' first task to begin after each service employee's start time.

$$\begin{aligned} t_{km} + T_r^H x_{krm} + \sum_{v \in \mathcal{R}} T_{d(r)o(v)} x_{kv(m+1)} & \\ - M_r(1 - x_{krm}) \leq t_{k(m+1)} & \qquad k \in \mathcal{K}, r \in \mathcal{R}, m \in \mathcal{M} \setminus \{|\mathcal{M}|\} \end{aligned} \tag{7.10}$$

$$t_{km} \leq t_{k(m+1)} \qquad k \in \mathcal{K}, m \in \mathcal{M} \setminus \{|\mathcal{M}|\} \tag{7.11}$$

$$T_r^S x_{krm} \leq t_{km} \qquad k \in \mathcal{K}, r \in \mathcal{R}, m \in \mathcal{M} \tag{7.12}$$

$$(T_p^S + T_{o(k)o(r)}) x_{kr1} \leq t_{k1} \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{7.13}$$

Time Usage Outside Planning Period

Constraints (7.14) track positive and negative time deviations from the planning period for each service employee. Constraints (7.15) ensure that service employees finish their tasks within the planning period $\overline{T}$, allowing an extra time of $\overline{T}^L$.

$$t_{k|\mathcal{M}|} + \sum_{r\in\mathcal{R}} T_r^H x_{kr|\mathcal{M}|} + t_p^- - t_k^+ = \overline{T} \qquad k \in \mathcal{K} \tag{7.14}$$

$$t_{k|\mathcal{M}|} + \sum_{r\in\mathcal{R}} T_r^H x_{kr|\mathcal{M}|} \leq \overline{T} + \overline{T}^L \qquad k \in \mathcal{K} \tag{7.15}$$

7.4.5 Binary, Non-negativity and Integer Definitions

$$x_{krm} \in \{0,1\} \qquad k \in \mathcal{K}, r \in \mathcal{R}, m \in \mathcal{M} \tag{7.16}$$

$$t_{km} \geq 0 \qquad k \in \mathcal{K}, m \in \mathcal{M} \tag{7.17}$$

$$t_k^+ \geq 0 \qquad k \in \mathcal{K} \tag{7.18}$$

$$t_k^- \geq 0 \qquad k \in \mathcal{K} \tag{7.19}$$

$$s_i^- \in \mathbb{Z}^+ \qquad i \in \mathcal{N}^{P-} \tag{7.20}$$

$$s_i^C \in \mathbb{Z}^+ \qquad i \in \mathcal{N}^{PC} \tag{7.21}$$

7.4.6 Big-M Calculation

To make the formulation tight, the big-M should be as small as possible. Constraint (7.10) uses M_r as big-M. Equations (7.22) show the calculation of the smallest M_r for all $r \in \mathcal{R}$. The equation takes into account the largest possible time difference accumulated by Constraints (7.10), if $x_{krm} = 0$.

$$M_r = \max_{v\in\mathcal{R},\, i\in\mathcal{N}^P\setminus\mathcal{N}^{P-}} T_{d(r)i} - (T_v^H + T_{d(v)i}) \qquad r \in \mathcal{R} \tag{7.22}$$

$$\tag{7.23}$$

Chapter 8

An ALNS Heuristic for the E-VReP

This chapter introduces an Adaptive Large Neighborhood Search (ALNS) heuristic for solving the Electric Vehicle Relocation Problem (E-VReP). The heuristic is based on the ALNS introduced by Ropke and Pisinger (2006). ALNS, in general, has proved efficient in solving large-scale VRPs. In addition, the use of ALNS in a dynamic VRP environment has produced good results in the benchmark comparison of Chen et al. (2018).

The ALNS heuristic from Ropke and Pisinger (2006) is adapted in several ways. The overall solution representation is now based upon finding an optimal set of car-moves, identical to what was done in Chapter 7. Instead of using Simulated Annealing (SA) as the local search strategy, a Tabu Search (TS) is adopted. Finally, similarly to Ropke and Pisinger (2006), LNS heuristics like *Shaw removal* and *k-regret* are utilized. These heuristics are modified to be applicable, given the chosen solution representation.

Section 8.1 gives a detailed overview of the ALNS. Section 8.2 describes the solution representation. Section 8.3 introduces the calculation of the objective function used to asses the quality of the solution. The construction of the initial solution is described in Section 8.4. The local and large neighborhood searches are presented in Sections 8.5 and 8.6, respectively. Finally, the technique that makes the search strategies adaptable is presented in Section 8.7.

8.1 Overview of the ALNS Heuristic

The ALNS heuristic is divided into two recurring processes: the Tabu Search (TS) component, and the Large Neighborhood (LNS) component. The initial solution is fed to the TS which locally searches for better solutions in a greedy manner. The local search continues as long as better solutions consistently are found. If I^{des} iterations are run without improvement to the global best, the LNS component is activated. The LNS component destroys and repairs the current solution, guiding the search into a new area of the search space. The TS is reactivated in the new neighborhood. This process is repeated until one of two termination criteria are met. The algorithm either terminates after T^{max} seconds or after I^{R} iterations without improvement to the global best solution. The ALNS process is illustrated in Figure 8.1.

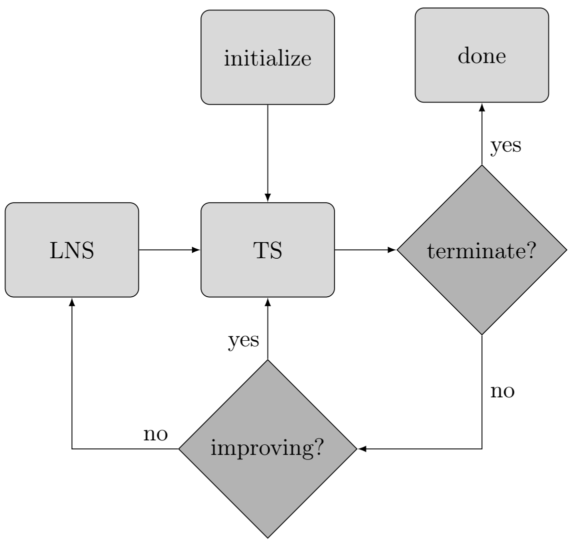


Figure 8.1: ALNS flow chart

A brief algorithmic model of the ALNS heuristic is provided in Algorithm 2. A set of available car-moves is initially provided to the model. This input is presented in Section 8.2. The goal of the heuristic is to find the best allocation and order of car-moves for each service employee $k \in \mathcal{K}$ according to the objective function described in Section 8.3.

Algorithm 2: Adaptive Large Neighborhood Search Heuristic

```
Input: $\mathcal{R}$ Set of car-moves
Output: Ordered list of car-moves for each service employee $k \in \mathcal{K}$
1  Solution $s$ = ConstructionHeuristic($\mathcal{R}$)                    Section 8.4
2  Best solution $s_{best} = s$
3  Iteration $I = 0$
4  while stopping criteria not met do
5      $\mathbb{M}$ = FindLocalNeighborhood(s)                           Section 8.5
6      $m_{best} = \arg\max_{m \in \mathbb{M}} f(m(s))$
7      $s = m_{best}(s)$
8      if $f(s) > f(s_{best})$ then
9          $s_{best} = s$
10     end
11     else if non-improving iterations $\geq I^{des}$ then
12         $s$ = LargeNeighborhoodSearch($s$)                           Section 8.6
13     UpdateWeights()                                                  Section 8.7
14     $I = I + 1$
15 end
```

Given the available car-moves, an initial solution is produced by the function *Construction-Heuristic* from Section 8.4. From line **3**, the heuristic starts the recurring process illustrated in Figure 8.1. In each iteration, the function *FindLocalNeighborhood* from the TS component provides a set $\mathbb{M}$ of possible local search operators (LSOs). An LSO causes a marginal change to the current solution. The heuristic chooses the LSO that results in the greatest improvement to the objective value, m_{best}, or the LSO that results in the least reduction if no improvements are found. Alternatively the ALNS may be tuned to choose the first improving LSO. $f(s)$ denotes the objective function value of solution s. The marginal change, caused by the selected LSO, to the current solution s is performed in line **6**.

If I^{des} non-improving iterations subsequently occur, the destroy and repair heuristics are evoked. A detailed description of these heuristics is provided in Section 8.6.

8.2 Solution Representation

This section re-introduces the concept of car-moves and introduces the solution representation used in the ALNS heuristic. The description adheres to all notation introduced in Chapter 7 unless stated otherwise.

Chapter 7 introduced the novel concept of car-moves, which consists of a car, an origin, and a destination. Given the problem instance, a set of possible car-moves for every car $c \in \mathcal{C}$ is fed as input to the heuristic. This set is created based on the current state of the carsharing system. In this chapter, car-moves are denoted r_c^i, where c denotes the car and i denotes the destination node.

A solution s consists of γ and β. γ consists of routes γ_k for each service employee $k \in \mathcal{K}$. γ_k is an ordered list of car-moves which employee k is going to carry out. β contains the unused car-moves, not present in γ. The two data structures are shown in Figure 8.2. By iteratively visiting the origin and destination of each car-move $r_c^i \in \gamma_k$ for employee $k \in \mathcal{K}$, the route and total travel time for all service employees are derived. The lth positioned car-move in γ_k is indexed by γ_{kl}.

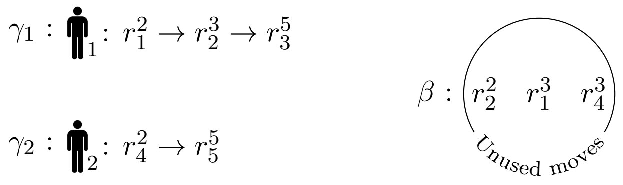

γ_1 : 1 : $r_1^2 \rightarrow r_2^3 \rightarrow r_3^5$
γ_2 : 2 : $r_4^2 \rightarrow r_5^5$
β : r_2^2 r_1^3 r_4^3
Unused moves

Figure 8.2: Solution representation. γ_k denotes the set of car-moves, in order, relocated by service employee k. β denotes the set of unused car-moves.

At most one car-move for each car may be present in γ, as each car may only be moved once during the planning period. β maintains a list of all car-moves not present in γ. Several search techniques are utilized to create interaction between γ and β as the algorithm progresses. These techniques are described in Section 8.5 and 8.6.

8.3 Feasibility and Objective Function

This section discusses how the ALNS handles infeasible solutions, and how a solution s is evaluated according to the objective function $f(s)$.

8.3.1 Feasibility

The ALNS allows infeasible solutions during the search to widen the search space. There are two feasibility criteria which may be broken. The first criteria is that each service employee $k \in \mathcal{K}$ can relocate all the assigned car-moves in γ_k within the planning period. An infeasible solution has car-moves in γ_k that are in excess due to the time available. The number of car-moves in γ outside the planning period $\overline{T}$ is calculated in Equation (8.6). Solutions are punished at a cost C^L for each car-move that exceeds the planning period. γ_k can easily be made feasible by moving the car-moves outside the planning period from γ_k to β. The second criteria state that γ must adhere to the maximum capacity constraints of the charging stations. Contrary to the time criteria, the capacity criteria consider all service employees $k \in \mathcal{K}$ instead of each employee individually. The number of violations is punished at a cost C^I per excess car. The criteria is mathematically defined in Equations (8.3) and (8.5).

8.3.2 Objective Value Calculation

The objective value is calculated in a bottom-up fashion by the objective function in Equation (8.7). The relocations that are carried out, and the time usage of every service employee k, are derived from γ_k.

Time Usage

The total time used by service employee k after relocating the lth car-move is denoted t_{kl}. t_{kl} is mathematically defined in Equation (8.1). Note that $o(\gamma_{kl})$ and $d(\gamma_{kl})$ refers to the origin and destination node of the lth car-move in γ_k. $|\gamma_{kl}|$ is the number of car-moves currently in γ_{kl}.

$$\begin{aligned} t_{kl} &= t_{k(l-1)} + T_{d(\gamma_{k(l-1)})o(\gamma_{kl})} + T^H_{\gamma_{kl}} \qquad \forall l \in \{1, \dots, |\gamma_k|\} \\ t_{k0} &= T^S_k \end{aligned} \tag{8.1}$$

To track the total time used by employee $k \in \mathcal{K}$ within the planning period (adjusted for extra time), t_k is introduced in Equation (8.2).

$$t_k = \max_{l \in \{0,\dots,|\gamma_{kl}|\}} \{t_{kl} : t_{kl} \le \overline{T} + \overline{T}^L\} \tag{8.2}$$

Relocation and Capacity Constraints

Car-moves in γ_k may be both parking-moves and charging-moves. The number of cars relocated to node $i \in \mathcal{N}$ within the planning period is denoted ϕ_i. ϕ_i is calculated in Equation (8.3).

$$\phi_i = \sum_{k \in \mathcal{K}} \sum_{l=1}^{|\gamma_k|} \begin{cases} 1, & \text{if } t_{kl} \le \overline{T} + \overline{T}^L, d(\gamma_{kl}) = i \\ 0, & \text{otherwise} \end{cases} \tag{8.3}$$

The deviation from the ideal state and violations of charging capacities can be derived from ϕ_i. It is possible to relocate more cars to a deficit node, than the total deficit, but this is not rewarded nor penalized. The number of parking-moves that are rewarded is denoted τ_i^P. τ_i^P is calculated in Equation (8.4). The capacity violation in charging node $i \in \mathcal{N}^{\mathcal{C}}$ is denoted τ_i^C. τ_i^C is calculated using Equation (8.5).

$$\tau_i^P = \min\{S_i^{0-}, \phi_i\} \tag{8.4}$$

$$\tau_i^C = \max\{\phi_i - N_i^{CS}, 0\} \tag{8.5}$$

Car-Moves Outside the Planning Period

μ_{kl} tracks if the l'th car-move in γ_k is in excess. μ_{kl} is calculated in Equation (8.6). Note that a solution with any $\mu_{kl} > 0$ is considered infeasible. However, a feasible solution may easily be extracted by removing excess car-moves.

$$\mu_{kl} = \begin{cases} 1, & \text{if } t_{kl} \geq \overline{T} + \overline{T}^L \\ 0, & \text{otherwise} \end{cases} \tag{8.6}$$

Objective Function

The objective function value $f(s)$ of a solution s is calculated in Equation (8.7). The ALNS heuristic seeks to maximize $f(s)$. Given feasible solutions, the objective function is equivalent to the objective function of the MIP model introduced in Chapter 7.

$$\begin{aligned} f(s) = & \sum_{i \in \mathcal{N}^{P-}} C^D \tau_i^P + \sum_{i \in \mathcal{N}^C} C^{Ch} \phi_i - \sum_{k \in \mathcal{K}} C^T t_k - \sum_{k \in \mathcal{K}} C^{ET}(t_k - \overline{T}) \\ & - \sum_{k \in \mathcal{K}} \sum_{l=1}^{|\gamma_k|} C^R T^H_{\gamma_{kl}} (1 - \mu_{kl}) - \sum_{n \in \mathcal{N}^C} C^I \tau_i^C - \sum_{k \in \mathcal{K}} \sum_{l=1}^{|\gamma_k|} C^L \mu_{kl} \end{aligned} \tag{8.7}$$

An objective not considered in the MIP formulation is the time of arrival in charging nodes when relocating cars for charging. Early arrivals may be beneficial in a dynamic setting to reduce the number of cars unavailable due to low battery levels. Hence, an additional term in the objective function (8.7) is presented in Equations (8.8) and (8.9). σ_{kl} indicates if car-move γ_{kl} is a charging-move, and performed within the planning period. R^{DC} is the set of all charging moves. C^{ChE} is the reward per time unit of charging cars early in the planning period $\overline{T}$.

$$\sigma_{kl} = \begin{cases} 1, & \text{if } t_{kl} \leq \overline{T}, \gamma_{kl} \in \mathcal{R}^{DC} \\ 0, & \text{otherwise} \end{cases} \tag{8.8}$$

$$\sum_{k \in \mathcal{K}} \sum_{l=1}^{|\gamma_k|} C^{ChE} \sigma_{kl} (\overline{T} - t_{kl}) \tag{8.9}$$

8.4 Construction of the Initial Solution

The initial solution is created by the construction heuristic outlined in Algorithm 3. The construction heuristic is greedy, resulting in low run-times. The heuristic tries to identify a fair distribution of car-moves among the service employees with respect to the objective function.

The construction heuristic derives solution s, γ, and β in one integrated greedy approach. Initially, β contains all car-moves and γ is empty. The heuristic will sequentially, for each service employee $k \in \mathcal{K}$, iterate through all cars that do not have a car-move present in γ. For each car, $c \in \mathcal{C}$, the car-moves belonging to the car in β are evaluated. The objective value, from inserting either of the car-moves at the end of γ_k is denoted $f(s^{+r_c^i})$. The car-move, yielding the best results according to Equation (8.10), is chosen for insertion.

$$\arg\max_{r_c^i \in \beta} f(s^{+r_c^i}) - f(s) \tag{8.10}$$

The heuristic terminates when all cars have a car-move present in γ.

Algorithm 3: Construction Heuristic

```
Input: $\mathcal{R}$ Set of car-moves, $\mathcal{C}$ Set of cars
Output: $\gamma$, $\beta$
1 Add all car-moves in $\mathcal{R}$ to $\beta$
2 while $\mathcal{C} \neq \emptyset$ do
3     for $k \in \mathcal{K}$ do
4         $r_c^i = \arg\max_{r_c^i \in \beta | c \in \mathcal{C}} f(s^{+r}) - f(s)$
5         Add $r_c^i$ to the end of $\gamma_k$
6         Remove c from $\mathcal{C}$
7         Remove $r_c^i$ from $\beta$
8     end
9 end
```

8.5 Local Neighborhood Search

This section introduces the details of the local neighborhood search used in the ALNS heuristic. Tabu Search (TS) is the overall search strategy. The TS proceeds by generating a local neighborhood, $\mathbb{M}$, consisting of local search operators (LSOs). LSOs are categorized into LSO types. Section 8.5.1 presents the different LSO types. The different ways to generate the neighborhood $\mathbb{M}$ are presented in Section 8.5.2. Section 8.5.3 introduces the tabu list. Finally, Section 8.5.4

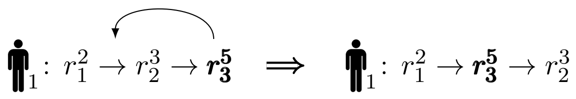
shows the techniques for selecting an LSO in the generated neighborhood.

8.5.1 Local Search Operators

The TS proceeds by applying local search operators (LSOs) that marginally changes the current solution. This section describes the set of LSO types, Q. Q is divided into two subsets, basic LSO types, Q^B and ejection LSO types Q^E, each consisting of four different LSO types. The basic LSO types are inspired by the heuristics introduced by Gendreau et al. (1992). The basic LSO types are *Intra Move*, *Inter Move*, *Inter-2 Move*, and *Inter Swap*. The ejection LSO types swap car-moves between γ and β. The ejection LSO types are *Ejection Insert*, *Ejection Remove*, *Ejection Replace* and *Ejection Swap*.

Intra Move

An Intra Move LSO moves a car-move within the list of car-moves for one service employee. See Figure 8.3 for an example.

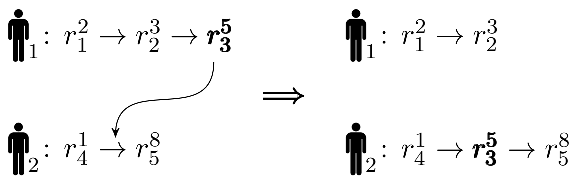


$_1$: $r_1^2 \to r_2^3 \to \boldsymbol{r_3^5}$ $\Longrightarrow$ $_1$: $r_1^2 \to \boldsymbol{r_3^5} \to r_2^3$

Figure 8.3: Intra Move example. One car-move is moved within γ for a single service employee.

Inter Move

An Inter Move LSO moves a car-move from one service employee to another. See Figure 8.4 for an example.



$_1$: $r_1^2 \to r_2^3 \to \boldsymbol{r_3^5}$
$_2$: $r_4^1 \to r_5^8$
$\Longrightarrow$
$_1$: $r_1^2 \to r_2^3$
$_2$: $r_4^1 \to \boldsymbol{r_3^5} \to r_5^8$

Figure 8.4: Inter move example. A car-move, r_3^5 is moved from γ_1 to γ_2.

Inter 2-Move

An Inter 2-Move LSO moves two consecutive car-moves from one employee to another employee. See Figure 8.5 for an example.

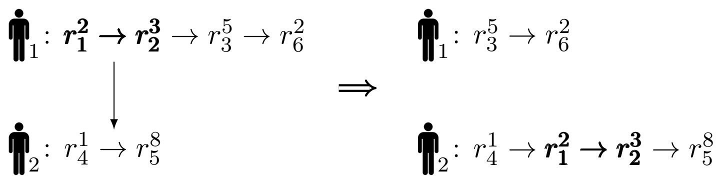

1: $\boldsymbol{r_1^2} \to \boldsymbol{r_2^3} \to r_3^5 \to r_6^2$
2: $r_4^1 \to r_5^8$
$\Longrightarrow$
1: $r_3^5 \to r_6^2$
2: $r_4^1 \to \boldsymbol{r_1^2} \to \boldsymbol{r_2^3} \to r_5^8$

Figure 8.5: Inter 2-move example. Two consecutive car-moves, r_1^2 and r_2^3, are moved from γ_1 to γ_2.

Inter Swap

An Inter Swap LSO swaps two car-moves between two service employees. See Figure 8.6 for an example.

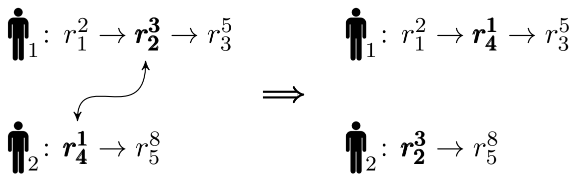

1: $r_1^2 \to \boldsymbol{r_2^3} \to r_3^5$
2: $\boldsymbol{r_4^1} \to r_5^8$
$\Longrightarrow$
1: $r_1^2 \to \boldsymbol{r_4^1} \to r_3^5$
2: $\boldsymbol{r_2^3} \to r_5^8$

Figure 8.6: Inter Swap example. Two car-moves, r_2^3 and r_4^1, are swapped between γ_1 and γ_2.

Ejection Insert

An Ejection Insert LSO picks a car-move from β, and randomly inserts it into γ. See Figure 8.7 for an example.

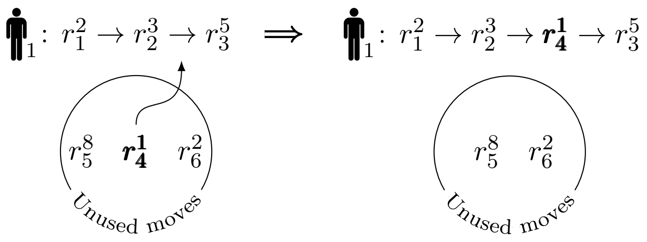

1: $r_1^2 \to r_2^3 \to r_3^5$
r_5^8 $\boldsymbol{r_4^1}$ r_6^2
Unused moves
$\Longrightarrow$
1: $r_1^2 \to r_2^3 \to \boldsymbol{r_4^1} \to r_3^5$
r_5^8 r_6^2
Unused moves

Figure 8.7: Ejection Insert example. Car-move r_4^1 is moved from the pool of unused car-moves into γ_1.

Ejection Remove

An Ejection Remove LSO removes a car-move from γ and place it into the pool of unused car-moves β. See Figure 8.8 for an example.

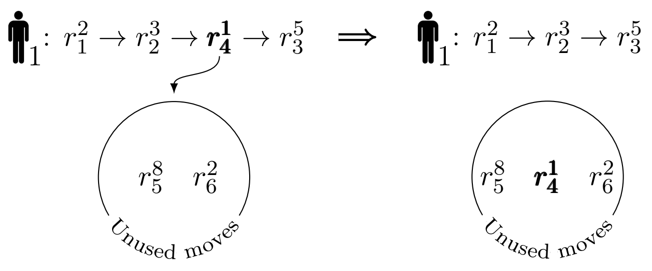

$_1$: $r_1^2 \rightarrow r_2^3 \rightarrow \boldsymbol{r_4^1} \rightarrow r_3^5$
$\Longrightarrow$
$_1$: $r_1^2 \rightarrow r_2^3 \rightarrow r_3^5$
r_5^8 r_6^2
Unused moves
r_5^8 $\boldsymbol{r_4^1}$ r_6^2
Unused moves

Figure 8.8: Ejection Remove example. Car-move r_4^1 is moved from γ_1 to the pool of unused car-moves.

Ejection Replace

An Ejection Replace LSO swaps a car-move from γ with a car-move from β associated with the same car. In other words, the destinations of a car-move in γ is changed. See Figure 8.9 for an example.

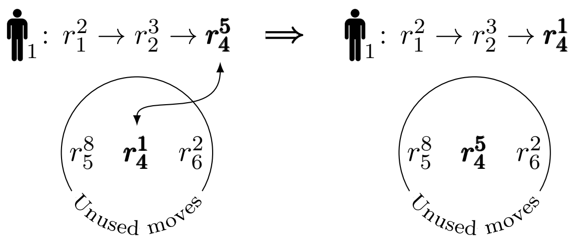

$_1$: $r_1^2 \rightarrow r_2^3 \rightarrow \boldsymbol{r_4^5}$
$\Longrightarrow$
$_1$: $r_1^2 \rightarrow r_2^3 \rightarrow \boldsymbol{r_4^1}$
r_5^8 $\boldsymbol{r_4^1}$ r_6^2
Unused moves
r_5^8 $\boldsymbol{r_4^5}$ r_6^2
Unused moves

Figure 8.9: Ejection Replace example. Car-move r_4^5 is swapped with r_4^1. r_4^5 is taken from γ_1 and r_4^1 is taken from the pool of unused car-moves.

Ejection Swap

An Ejection Swap LSO swaps a car-move from γ with a car-move from β, associated with different cars. An Ejection Swap differs from an Ejection Replace by swapping cars, while an Ejection Replace only replace a car-move with another car-move for the same car. See Figure 8.10 for an example.

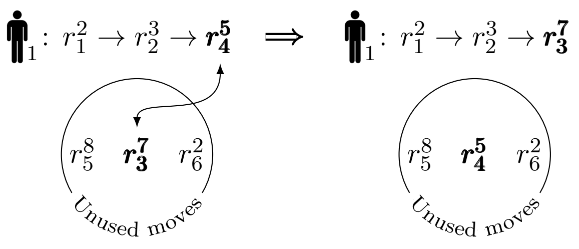

$_1$: $r_1^2 \rightarrow r_2^3 \rightarrow \boldsymbol{r_4^5}$
$\Longrightarrow$
$_1$: $r_1^2 \rightarrow r_2^3 \rightarrow \boldsymbol{r_3^7}$
r_5^8 $\boldsymbol{r_3^7}$ r_6^2
Unused moves
r_5^8 $\boldsymbol{r_4^5}$ r_6^2
Unused moves

Figure 8.10: Ejection Swap example. Two cars, represented by car-moves r_4^5 and r_3^7, are swapped. r_4^5 is taken from γ_1 and r_3^7 is taken from the pool of unused car-moves.

8.5.2 Construction of the Local Neighborhood

There are three different strategies for constructing the local neighborhood $\mathbb{M}$; *Full Enumeration*, Random Generation of one LSO Type (denoted *Random Weighted Enumeration*), and Full Enumeration of one LSO Type (denoted *Full Weighted Enumeration*). The three approaches are described in the following sections.

Full Enumeration

Full enumeration creates a neighborhood that consists of all possible LSOs of all LSO types $q \in Q$. Full enumeration can potentially generate a substantial number of neighboring solutions, depending on the size of the problem. When enumerating all possible LSOs, there is a risk of getting stuck in local optima. This is because the search may get trapped in repeating sequences of LSOs. Local optima can be avoided using a tabu list. However, the length of the tabu list only restricts the minimum size of such a repeating sequence. Section 8.5.3 explains the details of the tabu list.

Random Weighted Enumeration

Random weighted enumeration starts by selecting one LSO type $q \in Q$. The LSO type is selected in a roulette wheel fashion, based on adaptive weights from Section 8.7. The neighborhood is created by generating M^{max} LSOs of the selected LSO type. The LSOs are generated randomly. The random nature of the generation means that not necessarily all possible LSOs of the selected type will be generated. One advantage of the random generation is that it has a low probability of generating the same neighborhood twice. It is therefore unlikely for repeating sequences of neighborhoods to occur. Repeating sequences are often associated with local optima. A drawback with the random neighborhood generation is that the most favorable LSOs of the chosen LSO type may not be generated.

Full Enumeration of one LSO type

Full enumeration of one LSO Type is similar to random weighted enumeration. However, in full enumeration all possible LSOs of the selected LSO type is generated when creating the neighborhood. Compared to full enumeration, there may be a lower chance of getting stuck in local optima, due to the randomness involved when selecting the LSO type.

8.5.3 Tabu List

To further prevent the local search algorithm from getting stuck in local optima, a tabu list is introduced. The tabu list ensures that recently applied LSOs are left out of the neighborhood. The tabu list is adaptive, i.e., its size changes depending on how the search proceeds. If the last I^B iterations have been unsuccessful in finding a local improvement, the length of the tabu list is doubled. Likewise, if the previous I^S iterations have been successful, the length of the tabu list is halved. A lower and upper threshold limits the length of the tabu list.

8.5.4 Neighbor Selection

After the neighborhood $\mathbb{M}$ is constructed, a single neighbor $m_{best} \in \mathbb{M}$ is chosen. There are two strategies for choosing the neighbor. The *best neighbor* criteria picks the neighbor with the greatest objective value improvement. If no neighbor improves the current solution, the neighbor that reduces the objective value the least is chosen. The *first improvement* criteria iterates over the neighborhood and picks the first neighbor that improves the objective value of the current solution. If there does not exist a neighbor which improves the objective value of the current solution, the best neighbor is chosen.

8.6 Large Neighborhood Search

When the ALNS detects that the search is trapped in a local optima, the neighborhood of a solution s is enlarged to explore new areas of the solution space. This happens if no global improvements have been made during the last I^{des} iterations. The large neighborhood consists of combinations of destroy and repair heuristics. The destroy heuristics remove car-moves from γ. Subsequently, a repair heuristic inserts car-moves into γ from the set of unused car-moves β. The degree in which a current solution is destroyed and repaired is denoted as Γ. This means that a proportion Γ of the car-moves present in γ is removed and replaced. One destroy heuristic and one repair heuristic are chosen each time a LNS is executed. A roulette wheel based on adaptive weights is used to choose the heuristics.

8.6.1 Destroy Heuristics

This section presents the various destroy heuristics used in the ALNS heuristic.

Random Removal

Random Removal sequentially removes car-moves randomly and uniformly from γ. As promising parts of the solution may be removed, the effect is increased diversification in the search.

Worst Removal

Worst Removal greedily removes the seemingly worst parts of the current solution s. Combined with a repair heuristic, the intention is that more beneficial car-moves replace the car-moves causing the highest costs. In other words, the car-move reducing the objective function value the most is chosen. Mathematically defined, the following car-move is removed:

$$\arg\max_{r_c^i \in \gamma} f(s^{-r_c^i}) - f(s) \tag{8.11}$$

$f(s^{-r_c^i})$ is defined as the total objective function value of solution s when car-move r_c^i is removed from γ.

Related Removal

As first introduced by Shaw (1997), the Related Removal (also called Shaw Removal) removes *related* car-moves. Shaw (1997) observed that local search heuristics have tendencies to prefer shuffling similar objects in a solution. Hence, he proposed a technique which increases the number of unique objects. A relatedness measure $R(a,b)$ between car-moves a and b is, therefore, introduced. The proposed relatedness measure is based on Shaw (1997), but modified to fit the E-VReP.

$$\begin{aligned} R(a,b) &= \omega_1|o(a)-o(b)| + \omega_2|d(a)-d(b)| + \omega_3|c(a)-c(b)| \\ &\quad + \omega_4|T_a^H - T_b^H| + \omega_5|T_a^S - T_b^S| \end{aligned} \tag{8.12}$$

The first and second terms consider the origin and destination nodes of the car-moves, respectively. Nodes that are geographically closer are more related. The third term checks if both car-moves are either parking- or charging-moves. $c(r)$ returns one if the car-move's destination is a charging node. The fourth term compares the travel T_r^H time of the car-moves. The fifth term compares the start time T_r^S of the car-moves. The lower the values of $R(a,b)$, the more related the two car-moves are. The parameters $\omega_1 \ldots \omega_5$ weight the importance of each of the five measures.

Algorithm 4 shows the steps of Related Removal. Initially, a random car-move is chosen from γ, and inserted into $\mathcal{D}$. While keeping track of the car-moves already removed, a random car-move a from $\mathcal{D}$ is chosen. a's most similar car-move $b \in \gamma$ is ejected and placed in $\mathcal{D}$. The process repeats until a proportion Γ of the car-moves in γ is removed.

Algorithm 4: Related Removal

```
1 Removed car-moves $\mathcal{D} = \{\}$
2 $\mathcal{D} = \mathcal{D} \cup \{$Random car-move from $\gamma\}$
3 while $|\mathcal{D}| < \Gamma \cdot |\gamma|$ do
4     Car-move $a$ = Random car-move from $\mathcal{D}$
5     Car-move $b = \arg\max_{b \in \gamma} R(a,b)$
6     EjectionRemove($b$)
7     $\mathcal{D} = \mathcal{D} \cup \{b\}$
8 end
```

8.6.2 Repair Heuristics

This section presents the repair heuristics used in the ALNS framework. The repair heuristics iteratively insert car-moves from β into γ, repairing the routes dismantled by the destroy heuristic. The car-moves that are chosen are based on an insertion measure. This measure should give an indication of which moves that are beneficial to include in γ. The same number of car-moves that was removed in the destroy heuristic is inserted during the repair heuristic.

Greedy Insertion

The Greedy Insertion heuristic greedily inserts car-moves yielding the greatest improvement to the objective function value. This is done in the same manner as in Equation (8.10).

Regret Insertion

The Regret Insertion heuristic considers the alternative costs of inserting a car-move into γ. The basic 2-Regret considers the difference between the second best insertion and the best insertion of a car-move. The general k-Regret compares the k best alternatives to the best insertion. The car-move with the best 2-Regret value is given below:

$$\arg\max_{r_c^i \in \beta} \sum_{h=2}^{k} (f_h(s^{+r_c^i}) - f_1(s^{+r_c^i})) \tag{8.13}$$

The ALNS utilizes both the 2-Regret and the 3-Regret Insertion heuristics.

8.7 Adaptive Weights Adjustments

Adaptive weights, w, are used to guide both the local and large neighborhood search. Each LSO type, destroy heuristic and repair heuristic have weights associated with them. The weights are updated once in every segment of iterations. A segment for the LSO types consists of I^W consecutive iterations. Similarly, a segment for the destroy and repair heuristics consist of I^{des} iterations, without global improvements. A roulette wheel approach based on the assigned weights is used to decide which LSO types, destroy, and repair heuristics to use.

For simplicity, the roulette wheel and weight update procedures are only described for LSO types. The procedure is equivalent for destroy and repair heuristics. When applying the roulette wheel, the probability of choosing a specific LSO type is given in Equation (8.14).

$$\frac{w_q}{\sum_{\hat{q} \in Q} w_{\hat{q}}} \tag{8.14}$$

w_q is the weight associated with LSO type q. All weights are initialized to 1. A lower threshold is set for all weights in order to ensure that no weight is assigned the value zero. The associated LSO would never get chosen in the roulette wheel if this were to happen. The weight of each LSO type is updated based on its performance. Each LSO type is given a score, μ_q, $q \in Q$, depending on three factors presented in Table 8.1. The scoring criteria are inspired by the ALNS heuristic developed by Ropke and Pisinger (2006).

Table 8.1: Scoring criteria and their associated values for LSO

Score criterion	**LSO score value**
Global improvement	R_Q^G
New solution, local improvement	R_Q^L
New solution	R_Q^N

Successful LSO types achieve higher scores compared to unsuccessful ones. Those LSOs able to satisfy any of the score criteria from Table 8.1 are rewarded based on the *LSO score values*. The inequalities in (8.15) hold for the score values.

$$R_Q^G > R_Q^L > R_Q^N \tag{8.15}$$

After a segment of iterations, the weights are updated according to their performance. The weights w_q for all LSO types $q \in Q$ are updated using Equation (8.16). μ_q is the accumulated score in the current segment. In each iteration of the local search, the LSO type of the selected neighbor is given a score which is added to μ_q. R_Q^N is awarded if the selected neighbor is a solution that has not been found earlier in the search. Similarly R_Q^L is awarded if the neighbor has a better objective function value than the current solution, and the solution has not been found before. Finally, R_Q^G is awarded if the neighbor has a better objective function value than the best solution seen so far. μ_q is reset to zero after each segment of iterations.

$$w_q = w_q(1-\alpha) + \alpha\frac{\mu_q}{\theta_q} \tag{8.16}$$

θ_q is the number of times LSOs of type q have been used in the last segment. w_q, μ_q and θ_q are set to zero at the beginning of each segment. In the special case where μ_q and θ_q are both zero in the last iteration of a segment, the fraction $\frac{\mu_q}{\theta_q}$ is set to zero. α is a factor determining the responsiveness of weight updates. If $\alpha = 0$, no weights are updated. If $\alpha = 1$, the weights equal the last segment's scores.

Chapter 9

Simulation Model

This chapter presents the implementation of the Simulation Model used in the Rolling Horizon framework. Section 9.1 gives a high-level description of the Simulation Model. Section 9.2 describes the assumptions used in the Simulation Model. Section 9.3 introduces notation. Section 9.4 presents the implementation of the Simulation Model in more detail.

9.1 Introduction of the Simulation Model

The Simulation Model simulates the planning horizon of an artificial CSO delivering an electric and free-floating carsharing service. The Simulation Model is able to simulate the effects of relocation done by service employees and customers, imitating a real-world CSO as accurately as possible. To do so, information about all aspects of the system is tracked by the Simulation Model at all times. In a given time step, this information is referred to as the *system state*. The system state mainly consists of information regarding the entities shown in Figure 9.1. This includes the state of all nodes in the system, the state of the service employees, and the state of rental cars. The state of parking nodes consists of the number of cars parked. The state of charging nodes consists of the number of cars charging and the available charging slots. For the service employees, information such as location, their current task and where they are headed constitute the state. The state of a rental car contains information regarding its location, the battery level, and its availability to customers. When solving the E-VReP, information regarding the system state is needed.

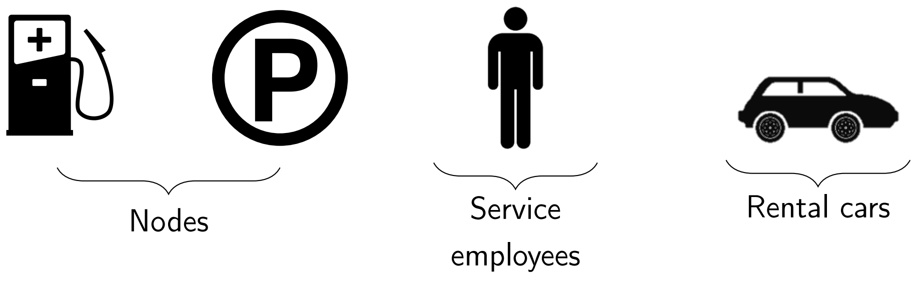


Figure 9.1: Entities in the Simulation Model

9.2 Simulation Model Assumptions

In addition to the assumptions introduced in Chapter 4, additional assumptions have been made in the creation of the Simulation Model. The assumptions are mostly related to the tasks of the service employees.

To naturally reduce the number of cars in the system that are not available for customers, the service employees prioritize rental cars with lower battery levels when choosing between which cars to charge. When relocating cars to satisfy expected demand, the cars are considered homogeneous.

9.3 Notation

This section introduces the parameters shown in Table 9.1. T_{charge} specify the number of minutes it takes to fully charge a rental car with zero battery. T_{range} is the total time a fully charged rental car can drive in normal city traffic.

Cars are divided into two categories; those in need of charging and those sufficiently charged. The cars with battery levels above the threshold ξ_{upper} are considered adequately charged. Only rental cars below the threshold are considered when employees choose cars for recharging. Customers can still rent cars with battery levels between ξ_{upper} and ξ_{lower} as the battery level is sufficient for shorter trips. Rental cars with battery levels below ξ_{lower} are not available to customers.

Table 9.1: Parameters used in the Simulation Model

Parameters	
T_{charge}	Time to charge battery
T_{range}	Car range
ξ_{upper}	Upper battery threshold
ξ_{lower}	Lower battery threshold

9.4 Implementation

The Simulation Model has been implemented in Java 8.0. Section 9.4.1 provides a brief algorithmic model of the Simulation Model implemented. Section 9.4.2 explains the processing of tasks in more detail.

9.4.1 Simulation Model Algorithm

Pseudocode for the Simulation Model is shown in Algorithm 5. The algorithm is run after every decision epoch in the Rolling Horizon framework. Variables such as T_{start} and $T_{increment}$ are fed to the Simulation Model specifying the start and the duration of the period to simulate.

The Simulation Model divides events into departures and arrivals. Departures consists of potential relocations and customer requests, while arrivals consists of soon to be completed relocations and customer rentals. Arrivals contain information about the destination nodes and the arrival times. When relocations and customer requests are allowed to start, they are converted to arrivals. The *event controller* verifies that potential relocations and customer requests do not violate any of the assumptions made. The event controller is detailed in Section 9.4.2.

Lines 4-9 in Algorithm 5 show the simulation of tasks from T_{start} until the end of the simulation period, $T_{start} + T_{increment}$. The variable t is used to track the start time of the previous event. Repeatedly, the Simulation Model finds the next event to happen after the time t. This is a simple process of finding the earliest arrival or departure of service employees and customers. Another event which may occur is that a rental car finishes charging. In this case, the fully charged car is moved from its charging node to the associated parking node. Every time a new task is approved, the battery levels of the cars are updated in line 7.

Algorithm 5: Simulation Model

```
   Input: $T_{start}$, $T_{increment}$, CustomerArrivals, EmployeeArrivals, EmployeeRoutes
   Output: System state
1  CustomerRequests = CustomerDemand.getActualDemand($T_{start}$, $T_{start} + T_{increment}$)
2  NextEvent = findNextEvent()
3  $t \leftarrow$ NextEvent.getTime()
4  while $t < T_{start} + T_{increment}$ do
5      System state, CustomerArrivals, EmployeeArrivals = doEvent(NextEvent)
6      NextEvent $\leftarrow$ findNextEventAfter($t$)
7      updateBatteryLevels($t$, min(NextEvent.getTime(), $T_{start} + T_{increment}$))
8      $t \leftarrow$ NextEvent.getTime()
9  end
```

9.4.2 Event Controller

The event controller performs the actions of the service employees, the customers, and the rental cars. However, due to multiple restrictions regarding each action, the event controller is also responsible for checking the validity of each action.

The actions in the Simulation Model utilize a Discrete Event System (DES) similarly to that of Febbraro et al. (2012) for free-floating carsharing systems. In a DES, the evolution of states depends on the customer demand and the relocations. However, for large-scale carsharing systems, the evolution of states would happen almost continuously. To solve this issue, DES utilizes simpler representation of the events happening in the system; it is sufficient to only track the start and end of actions while ignoring the intermediate states in between. For instance, the state representing the battery level of a car is only updated when the specific rental car is picked up or delivered to a node.

The verification of the actions ensures that the system state remains consistent with the restrictions. When customers request a rental car in a node, the event controller checks if there are any available cars in the node. If so, the car is removed, and a customer arrival with information regarding the rental is added to the list of customer arrivals. When a customer arrives in its destination node, the used car is parked. If the car is in need of charging, the customer sets the car to charging based on some probability measure. When the service employees relocate cars, cars are removed from their current node, and employee arrivals are added to the list of employee arrivals.

Chapter 10

Implementation and Test Instances

The ALNS heuristic introduced in Chapter 8 has been implemented in Java 8.0, together with the Simulation Model described in Chapter 9. The Mixed Integer Program (MIP) models, formulated in Chapter 7 and Appendix B, have been implemented using Xpress IVE version 1.24.18.

This chapter defines the key components used to create test instances. Section 10.1 describes the geographical layout and creation of test instances. This includes a brief description of test instances for the E-VReP and DE-VReP. Section 10.2 describes the initialization of the Customer Demand component in the Rolling Horizon framework. Finally, Section 10.3 discusses problem parameter values regarding travel times and the objective.

10.1 Test Instances

The geographical layout of all test instances is based on the city of Oslo, including the city center and surrounding suburban areas. The nodes are created using an overlying grid, defining each node as a square. Each node represents a parking node as described in Section 4.2. All test instances consists of different subsets of the 225 nodes shown in Figure 10.1. Travel data for the car, bike or public transport are fetched from Google maps.

Test instances are created by a generator developed in Python 3.3.6. Problem parameters, such as the number of nodes, number of charging stations, charging station capacities, and the number of service employees are given as inputs to the generator. The location of each node is drawn at random, with one constraint; To create realistic test instances, the nodes in Figure 10.1 are divided into three separate regions. When selecting nodes for a problem instance, all regions are included equally.

Sections 10.1.1 and 10.1.2 give a short description of the specifics of test instances for the E-VReP and DE-VReP, respectively.

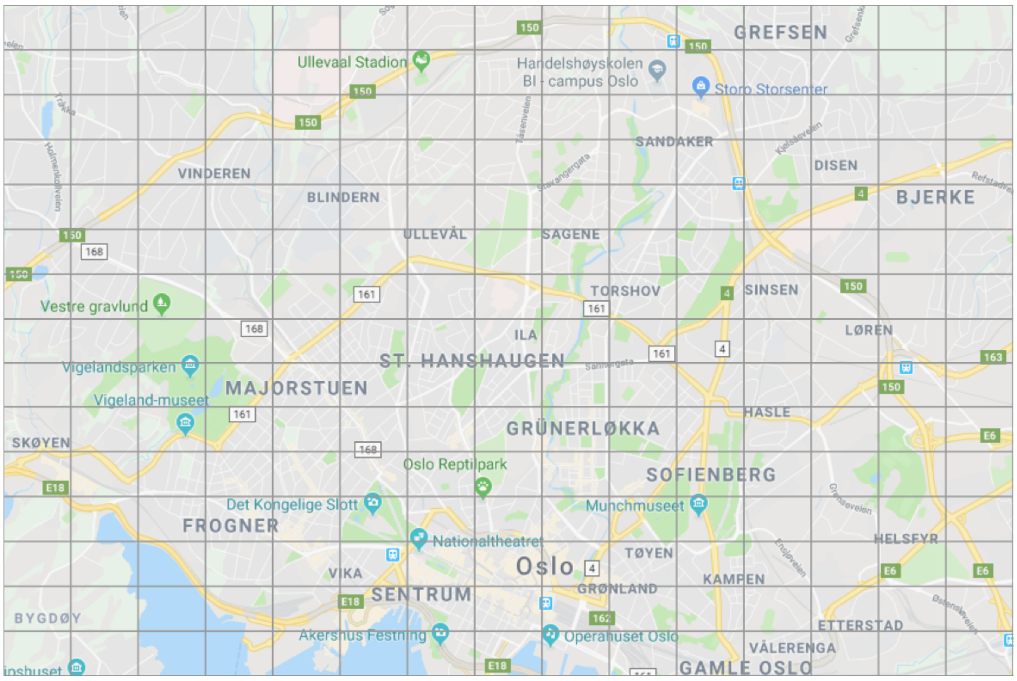


Figure 10.1: Nodes in the city of Oslo. Used as a basis for all test instances created

10.1.1 Test Instances for the E-VReP

The solution methods for the E-VReP are tested on a set of test instances. Testing is done to asses the performance of each of the introduced MIP models and to calibrate the parameters of the ALNS heuristic. Assessing the performance of the heuristic, on varying input parameters, may also provide feedback on how to tune the Rolling Horizon framework to ensure the best performance.

Initial State, Ideal State and Car-Moves

The generator takes the total number of needed relocations as input in order to control the difficulty of each instance it generates. Sequentially, the initial state, demand and ideal state in each node are initiated at random, constrained by the intended difficulty. Finally, car-moves are created based on the state of each node, following the process outlined in Chapter 7. It is not given that the ideal state is achievable within the planning period.

Test Instances

An overview of each test instance and their problem parameters can be seen in Table 10.1. Each test instance is categorized as small, medium, or large according to the size of the instance. All test instances have a planning period $\overline{T}$ of 60 minutes, and no overtime $\overline{T}^L$ is allowed. The letters a, b, and c will be used to distinguish between test instances of equal size.

Table 10.1: Test instances for the E-VReP

Instance	Nodes	Car-Moves	Cars To Relocate	Cars To Charge	Charging Stations	Service Employees	Size
6-3-3	6	12	3	3	2	3	small
8-4-3	8	18	4	3	2	3	
10-7-3	10	34	7	3	2	3	
15-9-3	15	60	9	3	2	3	
15-12-5	15	87	12	5	3	5	medium
20-13-5	20	132	13	5	3	5	
25-15-5	25	165	15	5	3	5	
30-18-5	30	177	18	5	3	5	
50-25-10	50	450	25	10	5	7	large
100-27-10	100	509	27	10	5	7	
125-30-10	125	650	30	10	5	7	
150-33-10	150	908	33	10	5	7	

10.1.2 Test Instances for the DE-VReP

The test instances for the DE-VReP, used in the Rolling Horizon framework, are built similarly to test instances for the E-VReP. However, each test instance is only a starting point, defining the carsharing system that will be simulated. A new problem instance is subsequently generated at every decision epoch, based on the new state of the system.

Three test instances are created by the same Python model from Section 10.1.1. The test instances are shown in Table 10.2. There are six charging stations per charging node for each test instances. The level of expected stress in the test instances is dependent on the relationship between the number of nodes, the number of cars and the available employees to relocate the cars. The three instances have different settings of these parameters. However, the relationship between the settings is similar; there are approximately three times as many cars as nodes.

Table 10.2: Set sizes and constant parameters used when generating instances for testing in the Rolling Horizon framework.

Test Instance	**Nodes**	**Cars**	**Service Employees**	**Charging nodes**
D-20-65-5-6	20	65	5	3
D-50-170-12-12	50	170	12	6
D-120-380-24-24	120	380	24	12

Parameters related to both the dynamic problem instances and the Simulation Model are shown in Table 10.3. At $T_{start} = 6$ AM, it is assumed that the distribution of available rental cars is close to the ideal state. The calculation of the ideal state is explained in Section 10.2.2.

Table 10.3: Set sizes and constant parameters used when generating instances for testing in the Rolling Horizon framework.

	Notation	**Value**
Start time business hours	T_{start}	6 AM
End time business hours	T_{end}	6 PM
Time increments	$T_{increment}$	15 min
Planning period	$\overline{T}$	60 min
Overtime	$\overline{T}^L$	10 min
Charging time	T_{charge}	210 min
Car range	T_{range}	120 min
Upper battery threshold	ξ_{upper}	40%
Lower battery threshold	ξ_{lower}	20%

10.2 The Customer Demand Component

This section describes the implementation of the Customer Demand component from the Rolling Horizon framework, as well as the estimation of the ideal state. Section 10.2.1 explains how the customer demand is generated. Section 10.2.2 presents the calculation of the ideal state.

10.2.1 Generation of Customer Demand

Customer demand is generated based on the traffic flow patterns observed in the city of Oslo using Google Maps. In the morning, traffic flows from the suburban areas into the city center. These flows decrease towards noon. From noon until 3-4 PM, the traffic from the city center to the suburban areas gradually increases with a rush hour peak around 4 PM. These findings led to the simple three-folded categorization of nodes; nodes with morning rush and lower demand in the afternoon, nodes with a steady and moderate level of demand during the entire planning horizon, and nodes with low morning demand but high afternoon demand. Nodes are individually associated with a demand scenario s. The scenario describes the expected number of cars that will be requested during the next hour, as indicated by λ_s. Nodes currently in a scenario of high demand are classified by $s = H$. Equivalently, nodes in scenarios with medium and low demand are associated with scenarios $s = M$ and $s = L$, respectively. Hence, λ_s is defined for all $s \in \{H, M, L\}$.

The demand in each node is assumed to follow a Poisson process where the arrival rate changes during the day. Given the expected number of arrivals, the timings of the arrivals appear approximately at random. The demand rates used, which evolve during the day, are given in Table 10.4. For instance, this means that nodes with morning rush have a rate of λ_H in the morning which linearly decreases towards λ_L in the afternoon.

Table 10.4: Expected number of cars requested for the three scenarios used in the Poisson process.

	Notation	**Number of cars demanded/hour**
High demand	λ_H	4
Medium demand	λ_M	1
Low demand	λ_L	0.3

Customers renting cars are most likely to follow the traffic flow pattern (e.g., travel from a node in outer Oslo to a node in the city center in the morning). Otherwise, the choice of destination node is made uniformly at random. The travel times of customers travels are also stochastic. For simplicity, we have assumed that customers who are renting cars always travel at least for ten minutes. In addition, travel times between departure and destination nodes are added. Since customers may have errands to run, each travel time of customers is adjusted by a factor drawn from a uniform distribution $U \sim \text{unif}(1, 1.4)$.

10.2.2 Calculation of Ideal State

The ideal state is the estimated optimal distribution of cars at the end of the current planning period. This distribution should reflect the distribution of the expected number of cars in each node in the look-ahead period, adjusted by the number of cars currently available for rental in the system. Hence, if this distribution of cars is met at the end of the planning period, the CSO will most likely meet the demand in the next planning period.

10.3 Problem Parameters

The problem parameters presented in this section are used for both dynamic and static test instances.

10.3.1 Time

Travel times between all parking nodes with different means of transportation are fetched from the Google Maps API. The center coordinates of each node are used as the reference point. The relocation time of each car-move, T_r^H is composed of the travel time by car between its origin and destination, plus the associated parking or maintenance time. Travel time between nodes denoted T_{ij}, is the shortest travel time when comparing travel time by bike and travel time by public transport. Charging nodes are assumed to share the same coordinates as their associated parking node.

All car-moves that goes to a charging node has an associated processing time of five minutes, to do necessary maintenance and start the recharging process. To parking nodes, the associated parking time is assumed to be four minutes, due to the time used to locate a suitable parking spot. All relocation times are assumed to be constant.

10.3.2 Objective Parameters in the E-VReP

The objective parameters introduced in Chapter 7 are C^{Ch}, C^{D}, C^{ET}, C^{R} and C^{T}. C^{D} and C^{Ch} represents benefits, but are referred to as cost in this section, for simplicity. Cost parameters are semi-artificial, meaning that they partly reflect their assumed real economic value. However, no study has been conducted to confirm the cost assumptions made. Costs are approximated and determined based on two main principles. First, the relative size of each cost component should reflect the importance of each cost. Secondly, each cost should incorporate its value in a dynamic environment, e.g., the benefit of charging a car is not observable directly, but is beneficial when simulating an entire day. The choice of approximated costs is shown in Table 10.5.

Table 10.5: The value of objective parameters

Cost component	**Value**
C^{PC}	30
C^{D}	10
C^{ET}	0.5
C^{H}	0.2
C^{T}	0.01

If cars are not recharged, two main outcomes may be observed. First of all, the point in time where the car is available to customers again is postponed, which may lead to a loss of profits. Secondly, it may lower customer satisfaction as the customers observe that too few cars are available in the system. Thus the benefit of recharging a rental car C^{Ch} is 30 per car charged. This encourages service employees to prioritize charging cars, which has proven a beneficial strategy in preliminary simulations. The benefit of approaching the ideal state, C^{D}, is 10 per unit improved. This estimate agrees with the estimated profits, found by Folkestad and Hansen (2017), when deviating from the ideal state. It is assumed that this cost is equal for all nodes. The cost per unit of overtime, C^{ET}, is 0.5. With this value, car-moves that are relocated in the last half of the planning period are always profitable to complete, even if overtime is used. The cost of relocation, C^{R}, is 0.2. This semi-artificial cost encourages service employees to relocate cars locally, to reduce the wear of the car park. However, any car-move with relocation time less than 50 minutes, is still profitable to relocate. Finally, the cost of time C^{T} is 0.01. This encourages the service employees to finish their tasks as early as possible.

Chapter 8 introduced three additional objective parameters; C^{I}, C^{L} and C^{ChE}. C^{I} is the cost of breaking the capacity of charging stations, per unit in excess. This cost is 100. C^{L} is the cost of having car-moves in excess in the solution. The cost is 10 per car-move. Finally C^{ChE} is the potential benefit of charging early. This parameter is 0.1.

Chapter 11

Computational Study

This chapter presents the computational study of the proposed solution methods for the DE-VReP and the associated E-VReP. Section 11.1 describes the test environment. Section 11.2 presents the computational study of the Mixed Integer Program (MIP) models for the E-VReP. Similarly, the calibration of the ALNS heuristic for the E-VReP is presented in Section 11.3. Section 11.4 presents and discusses the computational study of the solution method for the DE-VReP. Finally, Section 11.5 presents a selection of practical insights for real-life CSOs.

11.1 Test Environment

The software and hardware used to implement and test the MIP models and the ALNS heuristic, are presented in Table 11.1.

Table 11.1: Hardware and software used in testing

Processor	3,4GHz Intel E5
Memory	512GB RAM
Operating System	CentOS 7.4
Xpress-IVE version	1.24.18
Xpress optimizer version	29.01.10
Mosel version	4.0.3
Java version	9.0.4

Due to testing purposes, two hours of computation time is allowed to solve the E-VReP in the computational study of the MIP models. This is done to create optimal or near optimal solution benchmarks for the calibration of the ALNS heuristic on E-VReP instances. The ALNS runs until termination. Results measured for each test instance include computation time and a *optimality gap*. The optimality gap is defined as the percentage difference between the current objective value and the best-known objective value.

The DE-VReP is solved by iteratively solving a subproblem at each decision epoch. Each subproblem is an instance of the E-VReP. The computation time used to solve each E-VReP is regarded as the most crucial factor if the proposed ALNS heuristic is to be of use in a dynamic real-world scenario. In a dynamic setting, tasks appearing optimal at one moment of time may be aggravated in the next. Hence, the longest acceptable computation time to solve each E-VReP in a dynamic environment is assumed to be three minutes.

11.2 Comparison of the Mixed Integer Programs

Chapter 7 introduced two possible MIP models for the E-VReP. The first and current design denoted the Task-based MIP, is formulated in Chapter 7. This formulation gives each employee a set of task, which they can assign to car-moves. The second, denoted the Flow-based MIP, is an alternative design, formulated in Appendix B. The Flow-based MIP is a classical arc-flow model, and uses variables to indicate that an employee travels between relocating two car-moves. Both models are tested to confirm the strength of the current design. The active constraints for each MIP are presented in Table 11.2.

Table 11.2: Active constraints for the Task-based and Flow-based MIP

Model	**Constraints**
Task-based E-VReP MIP	(7.3) - (7.15)
Flow-based E-VReP MIP	(B.2) - (B.14)

The Task-based and Flow-based MIP models are solved by Xpress and tested on the test instances categorized as small (6-15 nodes) or medium (15-30 nodes) in size, introduced in Section 10.1.1. Large test instances are excluded, as preliminary testing has indicated that Xpress is not able to solve such IP models in reasonable time.

Table 11.3 shows the test results, from solving the small test instances. Tests results for all medium test instances can be found in D.2.

The Task-based MIP model outperforms the Flow-based MIP model. Optimal values are found for all small instances, with less than 15 nodes, within the allowed two hours of computation time. When solving the Task-based model, Xpress finds acceptable solutions for all test instances of medium size, with gaps ranging from 3.9% to 31.5%.

As highlighted in Chapter 7, there are several arguments explaining the performance gap between the two models. The Flow-based has a large number of variables used to indicate travel flow between car-moves. Many of these variables can be considered redundant for two reasons: only a few variables will be non-zero in an optimal solution, and many are mutually exclusive. The Task-based model, on the other hand, was designed with this in mind, containing fewer variables. Hence, the Task-Based MIP is the model of choice.

In terms of real life-value, Xpress is not able to solve the MIP models for medium test instances within what can be considered an acceptable time in a dynamic environment. A key finding in Hellem et al. (2017) was that the difficulty of each static sub-problem, in a Rolling Horizon

Table 11.3: Results from testing the task-based and flow-based MIP on the small test instances. Computational time and % gap is reported by Mosel.

Instance	Task-based MIP		Flow-based MIP	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3_a	0.68	0.0	4.28	0.0
6-3-3_b	0.76	0.0	4.02	0.0
6-3-3_c	1.66	0.0	16.24	0.0
Average	1.03	0.0	8.18	0.0
8-4-3_a	4.23	0.0	53.98	0.0
8-4-3_b	5	0.0	46.43	0.0
8-4-3_c	4.67	0.0	38.60	0.0
Average	4.63	0.0	46.34	0.0
10-7-3_a	250.22	0.0	7200	27.7
10-7-3_b	557.93	0.0	7200	N/A
10-7-3_c	703.68	0.0	7200	N/A
Average	503.94	0.0	7200	N/A
15-9-3_a	3201.99	0.0	7200	N/A
15-9-3_b	5730.11	0.0	7200	1004.6
15-9-3_c	7200	13.9	7200	N/A
Average	5377.40	4.6	7200	N/A

Green cells indicate best average values

framework, may vary between decision epochs. An originally easy test instance could transition into a medium or hard one. In these cases, reasonable solutions might not be found. Thus, the Task-based MIP is only used as a tool to benchmark the performance of the ALNS, on the test instances for the E-VReP.

11.3 Configuration of the ALNS

Chapter 8 introduced several parameters used by the ALNS. It also discussed possible strategies the ALNS could use for neighborhood generation and neighborhood selection. This section determines which strategies to select and describes the calibration of the parameters settings. Section 11.3.1 discusses the test methodology. Section 11.3.2 presents results for the different strategies, and determines the termination criteria. Section 11.3.3 discusses calibration of parameters that depends on the problem size. The remaining parameters, are calibrated in Section 11.3.4. Finally, Section 11.3.5 gives concluding remarks on the ALNS calibration.

11.3.1 ALNS Parameter Calibration Methodology

The ALNS is calibrated on all test instances introduced in Section 10.1.1. An overview of the ALNS parameters is given in Table 11.4. The following strategy is used for calibration: First, an initial value for every parameter is decided. This is done based on the results from Ropke and Pisinger (2006) and an extensive ad hoc, trial-and-error phase denoted preliminary testing. Secondly, calibration is done incrementally for each parameter. Each parameter is given a set of possible values, keeping the other parameters fixed. Once a parameter has been calibrated, it retains that value in further testing. The order of testing is done according to an estimate of each parameter's relative importance and will be presented in that order. Each parameter is calibrated based on objective value, introduced in Section 8.3, indirectly through the gap from the best-known solutions. For the small test instances, the optimal solution is known as found by the Task-based MIP. For problem instances where the MIP was unable to find the optimal solution, the gap is calculated based on the best-known solution found by the ALNS algorithm. Good performance on the larger test instances is credited the most, when calibrating each parameter, due to closer resemblance to real-life scenarios. Note, that all test results tables from the calibration can be found in Appendix D.

Table 11.4: Initial ALNS parameter values. These are the base values, which are to be calibrated.

Parameter	Value	Description
T^{max}	3 600/180	Max running time (seconds)
B^{init}	2	Initial tabu list size
B^{min}	2	Minimal tabu list size
B^{max}	1 024	Maximal tabu list size
I^R	100 000	Max number of iteration without improvement
I^W	100	The number of iterations before the LSO weights are updated
I^{des}	500	Iterations without global improvement before destroy and repair
I^B	4	Iterations without local improvement before increasing the tabu list size
I^S	2	Iterations with local improvements before decreasing the tabu list size
M^{max}	100	Neighborhood size
Γ	0.4	The destroy/repair factor
R_Q^N	1	LSO score for finding a new local solution
R_Q^G	33	LSO score for finding a new global best solution
R_Q^L	13	LSO score for finding a new better local solution
R_U^G	33	Destroy and repair score for finding a better global solution
R_U^L	13	Destroy and repair score for finding a new and better local solution
α	0.1	Update factor for both LSO and destroy and repair weights
$\omega_1 \ldots \omega_5$	0.315, 0.315, 0.315, 0.005, 0.05	Weights for Shaw Removal

The initial value of each parameter can be seen in Table 11.4. B^{init}, B^{min} and B^{max} are parameters that limit the tabu list size. Their values, derived from preliminary testing, are considered appropriate, and small changes to these parameters are considered not to affect the ALNS. Thus, no further calibration is done on their values. This is also the case for $\omega_1 \ldots \omega_5$, parameters which have also been determined based on knowledge of the problem structure. The remaining parameters are all calibrated. I^B, I^W, I^{des}, I^B, I^S, M^{max}, R_Q^N have initial values derived from preliminary testing, while Γ, R_Q^G, R_Q^L, R_U^G, R_U^L are based on values from Ropke and Pisinger (2006). Finally, the T^{max} for all test cases is three minutes, as described in Section 11.1. Additionally, when calibrating for neighborhood generation and neighbor selection, all test instances are solved with T^{max} equal to one hour. This is done to observe how the possible strategies affect performance when the ALNS is run for an equal number of iterations.

11.3.2 Algorithmic Configuration

The three approaches to neighborhood generation introduced in Section 8.5.2 are evaluated. Table 11.5 shows the performance of each strategy on all test instances. Several observations can be made: full enumeration of all LSOs of all types is naturally the most time-consuming approach. Notably, this approach also has the worst performance overall. A reasonable explanation is the increased risk of getting stuck in local optima, due to the to the many LSOs available. The goal of the tabu list is to restrict the search from going back the same path it originated from.

Full neighborhood enumeration may allow the ALNS to counter this restriction, by finding an equivalent set of LSOs, leaving the algorithm stuck locally. The same argument may be applied to explain the difference in performance between the full weighted and random weighted approaches. As described in Section 8.5.2, both strategies generate a neighborhood $\mathbb{M}$ of one LSO type, chosen in a roulette wheel fashion. The results indicate that having a fully enumerated neighborhood of one LSO type in every iteration may increase the risk of getting stuck in local optima. In these cases, randomness may reduce this risk and may assist in further exploration of the search space. Overall, the results show that the randomness embedded in random weighted neighborhood generation is beneficial for solving the E-VReP, and this is the selected approach used in further testing. This same observation is made when the ALNS run both 3600 seconds and 180 seconds.

Table 11.5: Average computational time and gap from best-known objective value for the three approaches to neighborhood generation, when running the ALNS with $T^{max} = 3\,600\text{s}/180\text{s}$ respectively. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value. A negative gap versus MIP, indicates an improvement to the objective value reported in Tables 11.3 and D.2.

Instance	**Full**				**Full Weighted**				**Random Weighted**			
	Comp. time (s)	Gap % (3 600s)	Gap % (180s)	Gap % MIP	Comp. time (s)	Gap % (3 600s)	Gap % (180s)	Gap % MIP	Comp. time (s)	Gap % (3 600s)	Gap % (180s)	Gap % MIP
6-3-3	6	0.0	0.0	0.0	1	0.0	0.0	0.0	3.2	0.0	0.0	0.0
8-4-3	18.8	0.0	0.0	0.0	1.8	0.0	0.0	0.0	3.6	0.0	0.0	0.0
10-7-3	33.6	3.3	3.3	3.3	3.8	2.6	2.6	2.6	5.6	2.0	2.0	2.0
15-9-3	53.2	0.6	0.6	0.6	7.4	0.3	0.3	0.3	9.4	0.0	0.0	0.0
15-12-5	140.6	1.0	1.0	-3.0	19.8	2.1	2.1	-1.8	14.2	0.9	0.9	-3.1
20-13-5	197.6	0.2	1.5	-3.7	25.2	0.5	0.5	-3.5	20.2	0.1	0.1	-3.9
25-15-5	236	1.5	0.9	-16.6	33.8	1.0	1.0	-17.2	23.6	0.4	0.4	-17.9
30-18-5	315.2	0.7	2.5	-12.1	44.2	0.4	0.4	-12.5	30.6	0.3	0.3	-12.6
50-25-10	1 464	2.8	5.9		194.4	2.4	2.4		89.6	1.5	1.5	
100-27-10	2 123.6	4.4	6.5		405	1.7	1.7		161.6	1.5	1.7	
120-30-10	2 719	7.4	9.0		577.4	3.0	3.7		294.8	1.7	2.5	
150-33-10	3 334	6.2	10.2		874	2.2	3.2		497.8	2.4	3.6	
Average	886.8	2.3	3.5	-4.0	182.3	1.3	1.4	-4.0	96.2	0.9	1.1	-4.4

Green cells indicate best average values

When the neighborhood $\mathbb{M}$ is generated, the ALNS picks one LSO from the neighborhood. Section 8.5.4 presented two strategies for neighbor selection. In each iteration, the ALNS may search through the entire neighborhood $\mathbb{M}$, and pick the best LSO. Alternatively, if the algorithm encounters an LSO that increases the current objective value, this operator is picked without evaluating the remaining neighborhood. The last strategy is called first improvement. Both strategies are tested, and the results are shown Appendix D, Table D.4. The results indicate that the best neighbor strategy performs best, and is used in the final implementation of the ALNS.

Finally, Section 8.1 introduced two termination criteria for the ALNS. T^{max} is constant, but I^R is calibrated. Reducing I^R reduces computational time, but may also deteriorate the objective function value, due to the non-deterministic nature of the ALNS. This creates a trade-off between computational time and objective value. Since $T^{max} = 180\text{s}$, this trade-off is more applicable to smaller test instances. I^R is tested for values ranging from 75 000 to 150 000, and test results can be seen in Appendix D, Table D.5. Overall, setting $I^R = 125\,000$ shows the greatest performance in terms of objective value. The increase in computational time is also well within what is acceptable, compared to the relative increase in the objective value. Figure 11.1 shows the development in the gap from best-known solution, for a typical run of the ALNS.

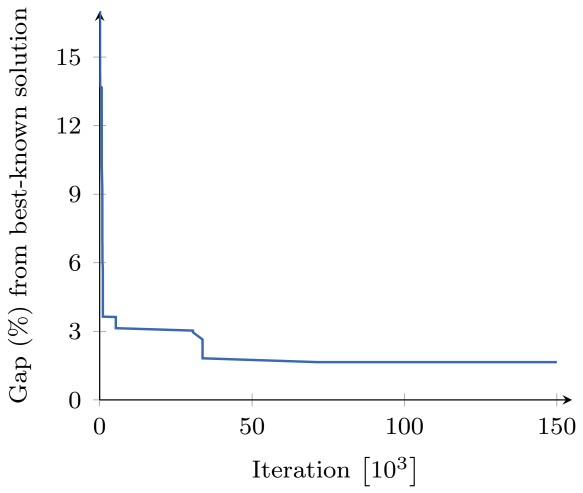


Figure 11.1: Gap % from best-known solution development during a run of 120-30-10.

11.3.3 Scaling Parameters Configuration

Preliminary testing indicates that the ideal values for M^{max}, I^{des} and I^{W} depend on the problem size. This is expected due to the fact that the neighborhood sizes of the LSO types and LNS heuristics increase with the number of car-moves and service employees. Thus, a good indication of the problem size is the number of car-moves in a problem, which increases polynomially with the number of cars to relocate. Interestingly, the number of car-moves present in solution γ does not necessarily increase with the number of car-moves in the instance, since the number of car-moves each employee can relocate is limited. Linearly scaling the parameters with the number of car-moves is observed to negatively impact the computational time and the objective value found. Thus, a scale based upon logarithmic growth is adopted. The scaling of each parameter is presented in Equations (11.1), (11.2) and (11.3). Figures that show the growth rate for different scaling values can be seen in Figures D.1, D.2 and D.3 in Appendix D.

$$M^{S} = \iota \ln(\mathcal{C}) \tag{11.1}$$

$$I^{des} = \upsilon \ln(\mathcal{C}) \tag{11.2}$$

$$I^{W} = \eta \ln(\mathcal{C}) \tag{11.3}$$

The maximal neighborhood size, M^{max}, is a parameter used by the random weighted neighborhood generation. For M^{max} iterations, the ALNS creates another LSO at random. A large M^{max} is beneficial when test instances grow, due to larger neighborhoods. Simultaneously, a larger M^{max} requires more computational resources. If M^{max} is too large, the ALNS could face similar issues as encountered in the full weighted neighborhood generation, since the randomness becomes somewhat limited. Test results for different values of ι are shown in Table D.6. Overall, $\iota = 25$ shows the most promising results.

I^{des} determines how often LNS is used in the ALNS. When no global improvement has been made for the last I^{des} iterations, the LNS is performed. The overall goal is to do LNS when the local neighborhood has been exhausted. The search space of the local neighborhood greatly varies with problem size. A large I^{des} gives the ALNS more iterations to search locally, which may be beneficial when the size of the local neighborhood increases. Additionally, every time LNS is executed, the weight of each destroy and repair heuristics is updated based on performance since the last execution. Larger values of I^{des} would make the algorithm less adaptable, since one heuristic may have time to dominate the others. The results in Appendix D, Table D.7, indicate that the highest values for υ perform better on larger test instances, giving the algorithm more time to perform local search. The final value for υ is 120. Using $\upsilon = 120$, Figure 11.2 shows how the weight of each destroy and repair heuristic develops during 150 000 iterations of the ALNS. The figure indicates that the algorithm adapts to which heuristics to use.

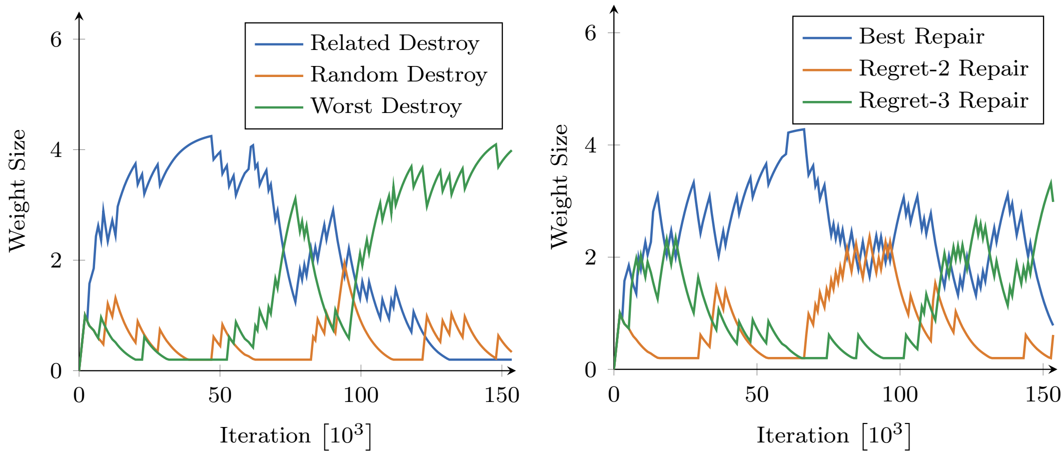


Figure 11.2: Values of the weights for the different LNS heuristics during 150 000 iterations on 120-30-10. Values are sampled at every weight update.

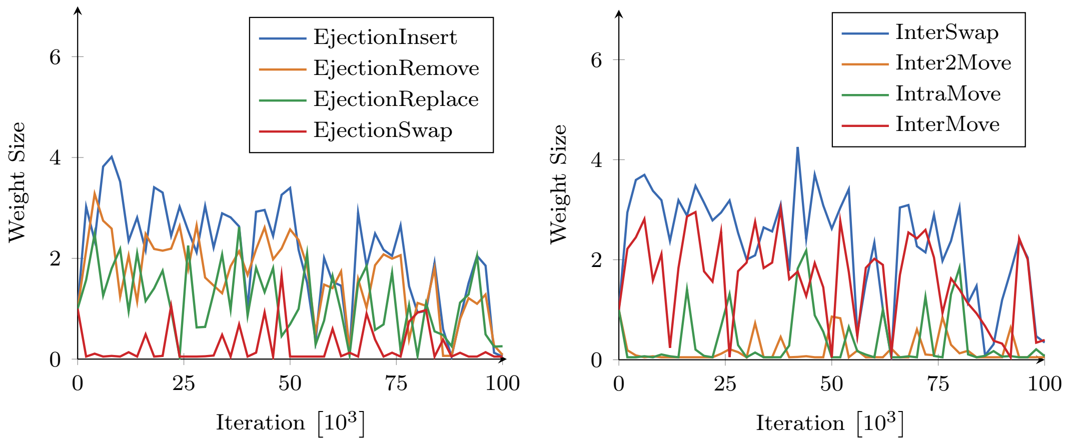


Figure 11.3: Values of the weights for the different LSO types during 100 000 iterations on 10-7-3. Values are sampled every 2 000 iterations, to improve visibility. and the volatility of each weight is higher in reality

Finally, I^W controls how often the weights for the different LSO types are updated. A small value for I^W makes the ALNS adaptable to immediate changes in the scores of the different LSO types. Figure 11.3 shows how the weights vary during a run of the ALNS. Table 11.6 showcases the results when $\iota = 25$ and $\upsilon = 120$ for different values of η. $\eta = 5$ shows the best performance. A reasonable explanation is that the ALNS needs to be very adaptable due to the large number of car-moves and LSO types. The algorithm continuously shuffles car-moves between γ and β, and thus benefits from changing weights values quickly as the solution develops. Figure 11.3

illustrates the volatility of the weights, when $\eta = 5$, moving from large values down to zero several times during a run. Some LSO types dominate others during the search. However, due to quick adaptation the other types are able to increase their weights quickly should new or better solutions be found. Overall, the trendlines for all LSO weights are going downward, as the possible search space becomes exhausted.

Table 11.6: Average computational time and gap from best-known objective value when running the ALNS with scaling values $\eta = 1/5/10/15$ for I^W. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	**$\eta = 1$**		**$\eta = 5$**		**$\eta = 10$**		**$\eta = 15$**	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	5.6	🟩 0.0	2.8	🟩 0.0	2.6	🟩 0.0	2.6	🟩 0.0
8-4-3	5.8	🟩 0.0	3.8	🟩 0.0	3.6	🟩 0.0	3.4	🟩 0.0
10-7-3	10.6	🟩 0.0	7.4	🟩 0.0	6.4	0.7	6.6	0.7
15-9-3	15.8	🟩 0.0	8.4	🟩 0.0	9.0	🟩 0.0	8.8	🟩 0.0
15-12-5	32.7	1.7	18.6	🟩 0.5	16.6	1.3	16.6	1.1
20-13-5	35.8	0.5	24.6	0.3	23.6	🟩 0.1	23.6	0.1
25-15-5	47.2	0.5	25.2	0.5	31.4	🟩 0.2	26.8	0.3
30-18-5	62.2	0.5	39.8	0.2	39.8	0.3	39	🟩 0.1
50-25-10	178.4	1.5	108.0	1.8	122.8	🟩 1.2	111.8	1.7
100-27-10	180	1.5	171.4	🟩 0.6	167.2	1.0	172.8	0.7
120-30-10	180	1.7	180	🟩 1.7	180	1.9	180	1.8
150-33-10	181.2	2.5	180	🟩 2.2	180	2.3	180	2.3
Average	77.9	0.9	64.2	🟩 0.6	65.3	0.7	64.4	0.7

Green cells indicate best average values

11.3.4 General Configuration

The remaining parameters are calibrated in order, using $\iota = 25$, $\upsilon = 120$ and $\eta = 5$. As described in Section 8.5.3, I^B is the number of iterations without local improvement before the tabu list is doubled in size. Accordingly, I^S is the number of iterations with local improvements before the tabu list is halved. Preliminary testing indicated that having I^B twice the size of I^S is a good fit for the ALNS. This allows the algorithm to search extensively away from local optima before the adaptive length of the tabu list restricts the search. The ALNS is tested with $I^B = 2/4/6/8$ and $I^S = 1/2/3/4$. All test results can be seen in Table D.9. The results indicate that having $I^B = 6$ and $I^S = 3$ is a good configuration. These are the final values for I^B and I^S.

Γ indicates the proportion of γ that is destroyed and repaired in a LNS. Unlike the approach in Ropke and Pisinger (2006), the ALNS removes a constant proportion of the current solution every time LNS is performed. Randomness is instead embedded in the size of γ, which greatly fluctuate during the search. However, a too small Γ can cause the ALNS to be stuck in local neighborhoods, and a too large Γ impacts the computational time, due to the resources required to do destroy and repair. The results in Table D.9, indicates that $\Gamma = 0.4$ is a good fit for the ALNS.

Finally, R^N_Q, R^G_Q, R^L_Q, R^G_U, R^L_U and α are parameters that the ALNS uses when updating the weights for the different LSO types, and LNS heuristics. Test results from calibration are shown in Tables D.11 and D.12. Based on the results, the final values are as follows: $R_Q^N = 1$, $R_Q^G = 23$, $R_Q^L = 13$, $R_{\mathcal{U}}^G = 23$, $R_{\mathcal{U}}^L = 13$ and $\alpha = 0.1$.

11.3.5 Final Remarks on the ALNS Calibration

The final values for all parameters can be seen in Table D.13. To showcase the performance of the algorithm in solving the E-VReP, final tests are run to compare the results of the fully calibrated ALNS to the solutions found by the construction heuristic. The results in Table 11.7 shows a performance increase of 45.1 percentage points (pp), on average, compared to a greedy approach.

Table 11.7: Average computational time and gap from the best-known objective value when running the calibrated ALNS against the construction heuristic.

Instance	**Calibrated ALNS**		**Construction Heuristic**	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	2	0.0	0	24.1
8-4-3	3.2	0.0	0	10.0
10-7-3	6.4	0.0	0	43.7
15-9-3	7.8	0.0	0	46.8
15-12-5	17.2	0.5	0	26.5
20-13-5	22.2	0.1	0	28.1
25-15-5	25	0.4	0	58.3
30-18-5	36.2	0.1	0	56.7
50-25-10	93.8	1.6	0	43.4
100-27-10	178.6	0.4	0	58.7
120-30-10	180	0.8	0	83.5
150-33-10	180	2.0	0	66.9
Average	62.7	0.5	0.0	45.6

Green cells indicate best average values

11.4 Configuration of the Solution Method for DE-VReP

This section tests the proposed solution method for the DE-VReP, introduced in Chapter 5. The implementation of the Simulation Model and the Customer Demand used in the solution method are the ones presented in Chapters 9 and 10, respectively. The E-VReP Solver in the Rolling Horizon framework is the fully calibrated ALNS assessed in Section 11.3.

The evaluation of the solution method is based on the objectives of the CSO presented in Chapter 4. The degree of demand served during the business hours is the most important key performance indicators of this section. The percentage demand served is referred to as DS. The number of rental cars charged by the service employees during the business hours is also presented. Section 11.4.1 describes the test instances used in more detail.

To calibrate the solution method, two tests are considered; Section 11.4.2 tests the length of the planning period, while Section 11.4.3 tests the re-planning frequency. Each test is run over ten days with different realizations of customer requests. The average scores over all days are used as a basis for comparison. To reduce the variance of the results, all models are run on the same set of realized customer requests.

11.4.1 Test Instance Considerations

The tests generated for the DE-VReP are designed to be stressful for the CSO. With the given number of cars in the system for each test instance and the expected demand in each node, the service employees are exposed to large workloads. On average, 22.5 cars are requested in each node during a twelve hour period. For each of the three test instances, this implies that there are 6-7 times more customer requests than cars in the system. Hence, even a 5 pp improvement in DS is approximately 135 more customers served in the case of 120 nodes.

Most cars are fully charged at the beginning of the planning horizon. However, the large number of customer rentals causes many cars to become in need of charging almost simultaneously after a given number of decision epochs. This wave of cars in need of charging is shown in Figure 11.4. In addition, there is a considerable amount of uncertainty to when and where rental cars are delivered by the customers. Hence, even though the service employees would perform optimal tasks at all times, achieving large degrees of DS in the DE-VReP is not likely.

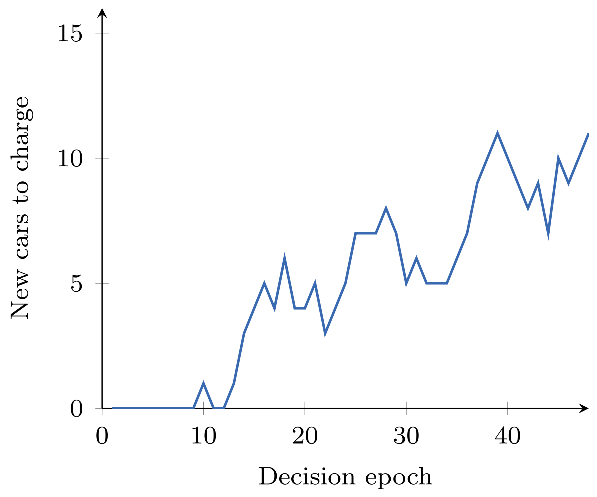


Figure 11.4: New cars in need of charging in every decision epoch for D-50-170-12-12

11.4.2 Planning Period

This test explores the effects of changing the length of the planning period $\overline{T}$. The length of the planning period restricts the number of relocations that the ALNS outputs for the service employees. Ideally, the static model would consider the whole planning horizon. However, there are three main arguments against such long planning periods. First, using longer planning periods increases the search space due to the increased number of possible routes for the service employees. A larger search space may increase the computation time needed for the ALNS to find good solutions. Second, the future states of the system are stochastic due to varying customer demand and travel times. Hence, longer planning periods come at the cost of more uncertainty. This implies that a solution looking optimal at the moment, may not even be feasible after the next couple of minutes due to unforeseen events. Finally, since the solution method for the DE-VReP re-plans sequentially, the actions done by the service employees are usually only the first couple of actions provided by the ALNS. Hence, the use of longer planning periods involves more calculations of needless actions that may are not likely to be performed. Table 11.8 substantiates these arguments.

Table 11.8: Demand served and cars charged for different planning periods

Instance	$\overline{T} = 40\,min$		$\overline{T} = 60\,min$		$\overline{T} = 80\,min$		$\overline{T} = 100\,min$		$\overline{T} = 120\,min$	
	DS %	Cars charged	DS %	Cars charged	DS %	Cars charged	DS %	Cars charged	DS %	Cars charged
D-20-65-5-6	58.35	53	64.10	58	60.48	53	59.48	60	58.98	53
D-50-170-12-12	60.36	139	63.74	136	62.81	131	61.31	129	61.77	124
D-120-380-24-24	57.51	279	58.31	285	57.37	266	58.41	255	56.19	251
Average	58.74	157	62.05	160	60.22	150	59.30	146	58.98	144

Green cells indicate best values for each test instance

Using a planning period of 60 minutes slightly outperforms the alternatives. There are three significant findings. First, the demand served reduces with longer planning periods. This is most likely due to increased search space making it more difficult for the ALNS to find good solutions within the three minutes. Second, if the planning period is too short, the solutions provided by the ALNS becomes more similar to a greedy approach. This explains the low demand served when using a planning period of 40 minutes. Third, fewer cars are charged when using longer planning periods. In these tests, there are no incentives for service employees to recharge rental cars early in the planning period. Thus, charging-moves are prone to the risk of being postponed iteratively, due to re-planning. However, reducing the planning period, charging-moves are more likely to be among the first and, therefore, more likely to be performed.

11.4.3 Frequency of re-planning

This test explores the effects of changing the re-planning frequency $T_{increment}$. The re-planning frequency determines the rate at which an E-VReP is solved in the Rolling Horizon framework. Given the result from Section 11.4.2, all tests use a planning period of 60 minutes. The results of three different re-planning frequencies are presented in Table 11.9.

From Table 11.9, it is evident that a re-planning frequency of 15 minutes slightly outperforms re-planning frequencies of 10 and 20 minutes. Re-planning more often should, intuitively, do no worse than re-planning more seldom. This explains why re-planning every 20 minutes has the worst performance. However, it is noteworthy that optimizing too often may have negative effects. One possible explanation is the current implementation of the ALNS. Since the ALNS is non-deterministic, the quality of the solutions, i.e., routes, between decision epochs may vary. Hence, when re-planning more frequently, the probability of finding less effective routes increases. If high-quality sub-routes were preserved between decision epochs, less effective solutions could be easier to avoid. Additionally, when the re-planning frequency is high, there are only small changes to the system's state in between decision epochs. Hence, re-planning too often might diminish the possible long-term benefit of the routes.

Table 11.9: Demand served and cars charged for different re-planning frequencies

Instance	$T_{increment} = 10\,min$		$T_{increment} = 15\,min$		$T_{increment} = 20\,min$	
	DS %	Cars charged	DS %	Cars charged	DS %	Cars charged
D-20-65-5-6	59.96	59	64.10	58	63.06	58
D-50-170-12-12	63.47	144	63.74	136	63.52	140
D-120-380-24-24	56.61	279	58.31	285	58.40	279
Average	60.01	161	62.05	160	61.66	159

Green cells indicate best values for each test instance

Another finding is that re-planning more often tends to increase the number of cars charged. The most likely explanation is that information regarding the battery levels is updated earlier in the Simulation Model. In addition, frequent re-planning increases the probability of charging rental cars. The probability increases because the ALNS has the option to charge cars early more frequently.

11.4.4 Comparison to Greedy Construction Heuristic

Section 11.3.5 showed that the ALNS had a performance increase of 45.1 pp compared to the construction heuristic. Table 11.10 shows the results from testing the construction heuristic in the Rolling Horizon framework. Interestingly, when solving the DE-VReP, the difference in DS is only 7.86 pp. This implies that the uncertainty faced when solving the DE-VReP reduces the performance gap between the two approaches. However, a difference of 7.86 pp in DS equals an additional 175 customers served throughout a twelve hour period. 175 additional customers represent a significant revenue increase, supporting the value of the proposed solution method.

Table 11.10: Comparing the calibrated solution method to the Construction Heuristic

Instance	**Construction Heuristic**	
	DS %	Cars charged
D-20-65-5-6	54.73	47
D-50-170-12-12	55.5	112
D-120-380-24-24	52.33	229
Average	54.19	129
Δ to ALNS	-7.86 pp	-31

When comparing the number of cars charged, the construction heuristic charges fewer cars. The likely explanation is that the ALNS provides routes with more efficient relocations of cars. Hence, in total, the service employees relocate more cars during the same time period, both to charging and parking nodes.

11.5 Practical Insights

This section discusses the practical use of the solution method for a real-life CSO. As introduced in Chapter 4, strategic and tactical decisions greatly affect the ability of the CSO to serve demand. Hence, in addition to the operational level, the proposed solution method may also be used as a decision support tool for both the strategic and the tactical levels. Hence, this section is divided into operational, tactical and strategic insights. The tests in this section are run similarly to those in Section 11.4, but only on test instance D-50-170-12-12. The fully calibrated solution method from Section 11.4 is referred to as the *standard solution method*.

11.5.1 Operational Insights

This section presents three operational insights; benefits of charging cars early, which destinations to consider for relocation, and the use of a uniform ideal state.

Benefits of Charging Cars Early

One objective of the proposed solution method is to charge cars in need of charging. Hence, charging-moves in the solution is rewarded. However, there is no guarantee that these relocations are ever done by the service employees if they are not among the first relocations in the solutions. Naturally, cars need to be charged in order to meet future demand. Hence, prioritizing early charging of cars seems beneficial. This test explores to what extent early charging should be prioritized. The reward for early charging is set to 0.1 per time unit, i.e., including and adjusting the C^{ChE} from Chapter 7.

Over a twelve hour period, rewards for early charging result in lower demand served. With $C^{ChE} = 0.1$ the standard solution method slightly outperforms early charging by approximately 1 pp. However, the number of cars charged when rewarding early charging is significantly greater. It is, therefore, interesting to test how these alternatives perform over a twenty hour period. In twenty hour periods, rewarding early charging improves the demand served by approximately 4 pp. The reason for this improvement can be deduced from Figure 11.5. When rewarding early charging of cars, the number of cars in need of charging is kept at a more steady level than when using the standard solution method. With the standard solution method, the system over time tends to become overloaded with cars in need of charging. The overload of unavailable cars forces the service employees to give less priority to relocations intended to satisfy the short-term demand of customers.

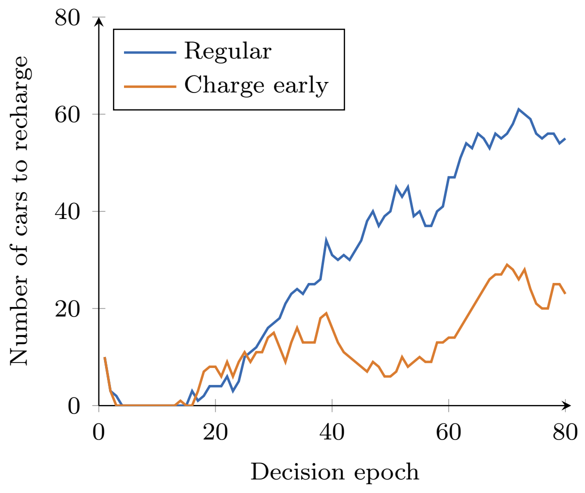


Figure 11.5: Development of cars in need of charging for D-50-170-12-12

Figure 11.5 shows the importance of a proper balance between the short-term and long-term considerations of satisfying customer demand. When charging cars early, the short-term demand served is slightly decreased. However, it is evident that the long-term costs of not meeting future demand are most likely higher than the short-term losses.

The benefit of charging early boils down to the preferences and opening hours of the CSO. For instance, if a CSO only allows car rentals during the daytime, it seems beneficial to prioritize serving demand short-term and do most of the recharging of cars during the night. However, charging all cars during the night requires a sufficient number of service employees to work the night shift. For instance, simple calculations show that for test instance D-50-170-12-12, charging all cars would require 12 hours of work. To prevent too much work at night, it is advisable to use strategies like early charging.

Destinations to Consider for Relocation

Chapter 7 introduced the possible destinations each car can be relocated to, identifying the set of car-moves. The search space of the ALNS solving the E-VReP consists of all these car-moves. Similar to the approach in Kirchler and Calvo (2013) for the Dial-a-Ride problem, it is possible to reduce the search space by removing car-moves not likely to be part of good solutions. This would simplify the solving of the E-VReP. Also, with a smaller search space, the runtime of the ALNS improves. However, this speed improvement may come at the cost of removing good solutions. This test explores the removal of car-moves from the search space.

The DE-VReP is solved in the Rolling Horizon framework over a day. In each run of the ALNS heuristic, the entire set of car-moves and the car-moves present in the best solution are saved. Figure 11.6 shows the distributions of all saved car-moves. The distribution is calculated by comparing the car-moves to the longest available travel time present in the test instance, i.e., travel times of car-moves are divided by the longest available travel time. Parking-moves and charging-moves are those present in the best solutions found by the ALNS.

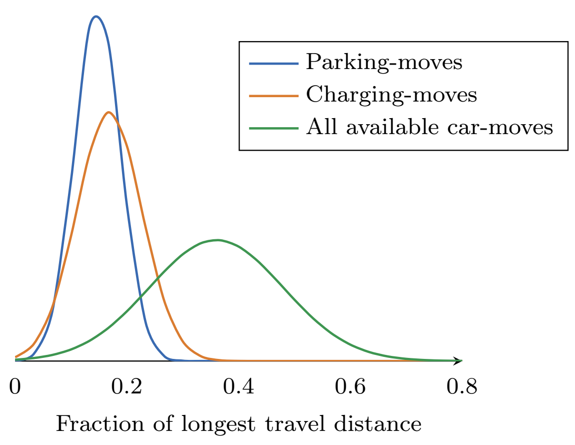


Figure 11.6: Distributions of car-moves for test instance D-50-170-12-12. Parking-moves and charging-moves are the distributions for car-moves present in best-found solutions. All available car-moves are the distribution for all car-moves identified.

Figure 11.6 shows that the car-moves used in the solutions of ALNS are among the shorter ones. The mean of parking-moves used by the ALNS is 0.15, while the mean of charging-moves is slightly higher at 0.17. Charging-moves may have a larger average simply because there are fewer charging nodes than parking nodes in a test instance. The mean of all available car-moves is 0.36. The average variance of parking and charging-moves is 0.05, while the variance of all car-moves is 0.12. These findings indicate that it may be possible to at least half the search space without degrading the quality of the solutions found by the ALNS, with the current configuration. This implies that cars should in most cases be relocated locally, a finding which considerably simplifies the operational problem faced by a CSO. Testing indicates that these findings also hold for instances D-20-65-5-6 and D-120-380-24-24.

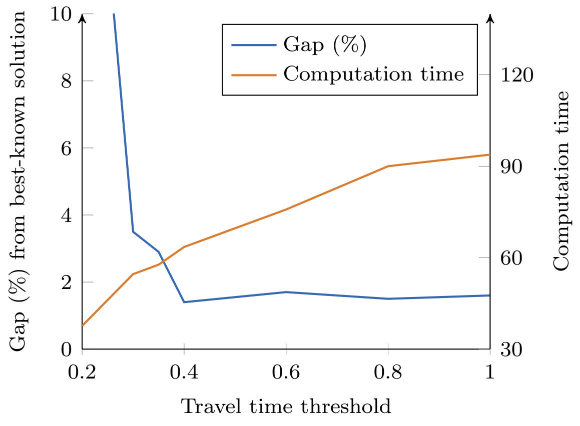


Figure 11.7: Gap (%) from best-known solution and computation time used when solving test instance 50-25-10 for the E-VReP. Different cutoffs for car-moves are used to indicate how the ALNS performs given the different sets of car-moves.

To test this finding, Figure 11.7 shows the result from removing car-moves above certain travel time thresholds for test instance 50-25-10. Interestingly, the ALNS manages to find the best-known solutions within the time limit with a threshold of 0.4. With lower thresholds than 0.4, car-moves that are part of the best-known solution are removed. Finding the optimal threshold for a CSO can impact both the speed and the quality of the solutions found. Hence, reducing the search space seems highly beneficial for real-life CSOs.

Using a Uniform Ideal State

The solutions provided by the ALNS are very much dependent on the given ideal state. In the Rolling Horizon framework, the calculation of the ideal state is the one described in Chapter 10. The ideal state reflects the expected number of cars demanded by customers, adjusted for the number of sufficiently charged cars in the system. Some nodes may have an ideal state of zero cars. However, even though the ideal state is zero, the uncertainty in the system may result in a demand requests in the empty node.

One approach to make the solution method more robust with regards to unforeseen demand requests is to strive for a uniform distribution of cars in all nodes. The new ideal state is calculated as follows; start by distributing an equal number of cars to all nodes. The remaining cars after this initial distribution are greedily distributed among the nodes with the greatest expected demand. The intention behind this strategy is to prevent nodes from being empty, while at the same time incorporating some of the forecasted demand. An example of the difference between the ideal states is shown in Figure 11.8.

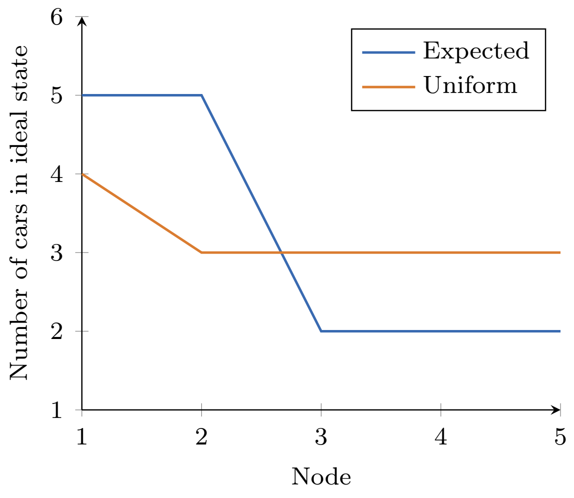


Figure 11.8: Ideal state of cars based on expected and uniform distribution. The two first nodes are exposed to morning rush. There are 16 cars available in the system. The uniform ideal state assigns three cars to each node. The last car is assigned to either node 1 or 2, in this case, node 1.

Testing shows that the uniform state on average performs 3 pp worse than the original ideal state. This result supports the importance of making ideal states that coincide with the demand patterns of customers. Hence, given that the CSO has accurate predictions of the expected demand, it seems beneficial to utilize these forecasts in the ideal state. If the demand predictions are inaccurate, a uniform ideal state is recommended.

11.5.2 Tactical and Strategic Insights

This section presents one tactical and one strategic insight; how the number of service employees and charging stations affects the demand served. Decisions regarding these numbers greatly affect the demand served by the system, as discussed in Chapter 4. The proposed solution method can provide decision support for other tactical and strategic aspects such as the number of cars, the cars' range and charging time, and the locations of the parking and charging zones. However, these tests are not included.

Number of Service Employees

The optimal number of service employees used in a carsharing system is dependent on the problem instance as well as the CSO's preference regarding the trade-off between costs and customer satisfaction. Intuitively, increasing the number of service employees strictly improves the performance of the system and vice versa. However, there is a trade-off between the marginal revenue gained by adding one more service employee and the marginal cost of employment. To illustrate this trade-off, tests where only the number of employees vary are performed. The results are shown in Figure 11.9.

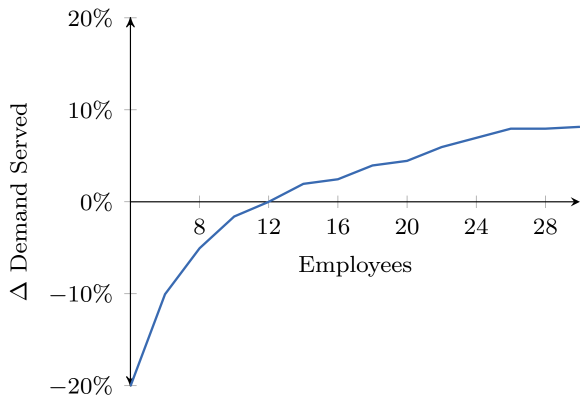


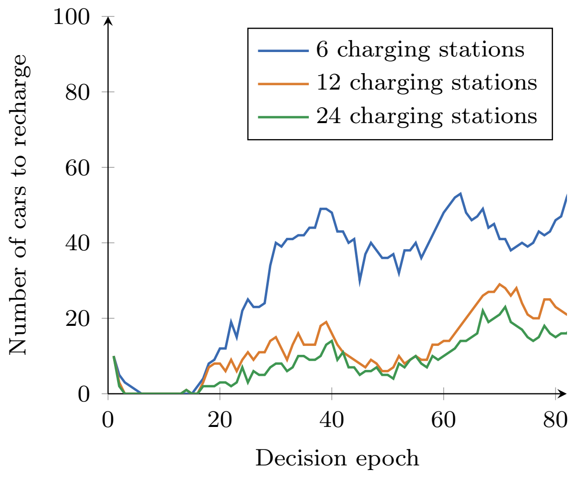
Figure 11.9: Percentage point difference in demand served when varying the number of service employees. The difference is compared to using 12 employees from the original test instance D-50-170-12-12.

It is evident from Figure 11.9 that the marginal value of additional service employees is diminishing. Too few service employees are punished by low degrees of demand served, while too many service employees yield no significant improvement in demand served. Based on the specific cost and revenue values of the CSOs, it is possible to optimize the number of service employees given the operating system. The optimal number of employees should be chosen where the marginal revenue of an additional employee is most similar to the marginal cost of employment, preferably equal.

Number of Charging Stations

For the rental cars to be available for customers during the operating hours, charging of cars is crucial. However, the number of cars that can be charged is restricted by the number of available charging stations. Due to capital costs associated with charging stations, this test explores the importance of a sufficient number of charging stations in the carsharing system. Based on the test instance D-50-170-12-12, two additional test instances are generated; one with 6 charging stations and one with 24 charging stations. For the three test instances, the charging stations are spread uniformly in the operating area. For more significant changes in the result, a planning horizon of twenty hours is used. In addition, charging of cars is prioritized as in Section 11.5.1.

Figure 11.10 shows the development in the number of cars in need of charging over the planning horizon for each of the three test instances. Naturally, keeping the number of cars in need of charging at lower levels results in more cars available for customers in future periods, which affects the demand served. Using this fact, Figure 11.10 shows that for the test instance used, halving the number of charging stations results in fewer cars available for customers. Figure 11.10 shows that DS reduces by 3.76 pp. When doubling the number of charging stations, the demand served increases by 1.87 pp. However, the increase in demand served diminishes when doubling the number of charging stations. Similar to the case of service employees, the number of charging stations should be chosen such that the marginal revenue from adding a charging station equals the marginal cost.



Instance	Δ DS%	Δ Cars Charged
6 charging stations	-3.76	-27
24 charging stations	+1.87	+39

Figure 11.10: Development of cars in need of charging over the planning horizon for instances with a different number of charging stations. The percentage point differences in demand served and difference in cars charged compared to 12 charging nodes are also presented.

Chapter 12

Concluding Remarks

This chapter concludes this thesis and outlines future research opportunities. Section 12.1 presents the conclusion, and Section 12.2 discusses the research opportunities.

12.1 Conclusion

This thesis presents a solution method for the Dynamic Electric Vehicle Relocation Problem (DE-VReP). The DE-VReP is concerned with routing of service employees and relocation of electrical cars in a free-floating carsharing system, solved in a dynamic environment during the opening hours of a carsharing organisation (CSO). The goal of the DE-VReP is to maximize the customer demand served, cost-effectively. Relocation includes recharging and transportation of rental cars to improve the distributions of cars in the system. Studies show that free-floating carsharing systems are prone to unbalanced distributions, and relocating vehicles can improve the their economic viability. Relocation is performed by service employees. When not relocating cars, the service employees can travel using folding bikes or public transportation. The solution method adopts a Rolling Horizon framework, solving static subproblems (E-VRePs) of the DE-VReP at different decision epochs. Between decision epochs, the state of the carsharing system is updated to incorporate newly available information.

Two solution methods to solve the E-VReP are proposed; a Mixed Integer Program (MIP) model, and an Adaptive Large Neighborhood Search (ALNS) heuristic based on Ropke and Pisinger (2006). The objective of each method is to maximize the number of cars charged and minimize deviation from an expected ideal distribution of rental vehicles. Since relocations are associated with a cost, the relocations should be done in a cost-effective way. A solution consists of routes for each service employee, which cars to relocate, and where to relocate them. The solution methods allow solutions where service employees originate and end at all locations in the operating area. Thus, the methods are capable of solving the E-VReP that arises in free-floating carsharing systems with electric vehicles, at any point in time. This feature makes them appropriate in a Rolling Horizon framework.

The E-VReP is a variation of classical vehicle routing and pickup and delivery problems. The problem structure of the E-VReP allows identification of the minimal set of possible relocation destinations for each car. Each element in this set is denoted a *car-move*. The proposed solution methods for solving the E-VReP utilize car-moves. The use of car-moves is a novelty which significantly simplifies the pickup and delivery aspect of the E-VReP, reducing the complexity of the problem.

The MIP model can be solved for small instances of the E-VReP. However, when solving realistic instances of the problem in the Rolling Horizon framework, the computation time required to solve the MIP model is not acceptable. The ALNS heuristic, on the other hand, can achieve satisfactory results for realistic instances of the E-VReP in less than 180 seconds. The heuristic finds the optimal solution for all test instances solvable by the MIP within less than 10 seconds. Additionally, the ALNS shows great stability when solving realistic problem instances, with an average gap of 0.5% to the best-known solution. The quality and efficiency of the ALNS heuristic enables it to solve realistic instances of E-VReP subproblems that arises in the solution of the DE-VReP. Hence, the ALNS heuristic is proposed as the solution method in the Rolling Horizon framework for the DE-VReP.

A simulation model is developed to test the proposed solution method on problem instances for the DE-VReP. The simulation model mimics the work day of an artificial CSO. The solution method is able to provide efficient solutions for test instances of at least 120 nodes and 380 rental cars. When stress-testing the solution method, it serves 62% of customers on average during a period of 12 hours. This equals 1 674 customer rentals served. Compared to a greedy approach, an additional 200 customers are served using the proposed solution method.

There are several benefits for CSOs using the proposed solution method. First, the solution method is capable of producing effective routes for service employees during a work day. The routes include relocations for all service employees to better serve demand while charging cars with low battery levels. This procedure is considered a vital task in maintaining a functional carsharing service. Second, the proposed solution method is flexible in terms of how each objective is weighted. This flexibility provides a CSO an opportunity to align the proposed solution method with their priorities. Finally, test results indicate that the solution method using the ALNS can provide insights to decision-making on other planning levels than the operational, e.g., determining the number and the location of charging stations, or the number of service employees and rental cars.

In conclusion, solving the DE-VReP with the proposed solution method provides high-quality solutions in reasonable computation time for realistic problem instances. Novel search methods has been introduced that effectively deal with the large search space of the problem. The solution method is versatile and adaptive to fit the preferences of CSOs. In total, we consider the proposed solution method a significant contribution to the creation of efficient, and lasting carsharing systems.

12.2 Future Research Opportunities

Future research opportunities are discussed in the following sections. Section 12.2.1 presents potential extensions on the strategic and tactical levels. Section 12.2.2 addresses a possible transition from the current solution method to a commercial product. Section 12.2.3 discusses how more realistic test data can be used to validate the value of the proposed solution method further. Finally, Section 12.2.4 discusses how autonomous vehicles can impact the carsharing systems of the future.

12.2.1 Decision Support on the Strategic and Tactical Levels

The proposed solution method, in combination with the simulation model, can provide decision support on both the strategic and tactical levels of carsharing. On the strategic level, further studies may be conducted on how CSOs should organize charging stations in their systems. The total number, locations, and capacity are key questions to consider. Decision support can be provided by iteratively simulating different charging station configurations, observing its impact on the operational level. On the tactical level, the booking and incentive schemes are of particular interest. Tuning the simulation model allows direct observation on how different schemes affect the operational performance. Booking schemes where the customer must book cars in advance would allow more predictability and is hypothesized to improve the performance of the proposed solution method.

12.2.2 Extensions to the Solution Method

The current simulation model is designed for solving the DE-VReP over a single day. The relocations carried out in the final hours of the working day, does not necessarily reflect the optimal distribution for the day to come. Neither does the trade-offs between different objectives while the system is in operation, as discussed in Section 11.5. Support for multi-day simulation should be implemented to further increase the potential commercial value of the proposed solution method. A multi-day simulation would involve further study into a realistic demand component, which takes into account day-to-day operations. Key considerations are optimal distributions at the end of each working day, and how to organize the service employees for potential work at night.

The weighting of each objective in the implemented solution method is assumed to be constant is this thesis. This assumption may not hold at any point in time in a dynamic environment. There are several possibilities for research concerning short and long-term considerations. In the short-term, each objective may be weighted, given the current state of the system. This may make the proposed solution method more adaptable. In the long-term, the objectives may be weighted based on trade-offs between the current state, a steady-state or a future state that a CSO strives to achieve. This could make the proposed solution method more stable.

Finally, there are possibilities to test alternative implementations of the large neighborhood search heuristics. Heuristics to explore would be the most time-consuming ones in the current implementation. In large test instances for the DE-VReP, the number of car-moves in the resulting E-VRePs has been observed to be high; up to 5 000 has been observed. The increased search space, especially for the k-regret and best-insertion repair heuristics, has a significant impact on the computation time required. An alternative approach is to consider a subset of the total available car-moves, chosen at random. Further testing needs to be done on the viability of this alternative.

12.2.3 Validation of the Solution Method

The current test data is generated based on the city of Oslo. However, there does currently not exist any electrical vehicle carsharing systems in Oslo. The DE-VReP solution method should be tested in a real-world carsharing system to verify its potential. If real-world test data were to become available, it would be easy to incorporate it into the current solution method.

12.2.4 Autonomous Vehicles

Many car companies and on-demand car companies, such as Uber and Lyft, are developing autonomous vehicles. Driverless cars are expected to be available for the public sometime between 2020 and 2030 (Autonomous cars, 2018). Driverless cars are going to impact the carsharing industry in significant ways. With autonomous vehicles, the DE-VReP will be altered, and most likely become a simpler problem due to fewer employees needed for the relocations. The solution method developed in this thesis would have to be slightly altered to the new problem. If the new problem turns out to be a simpler one, a modified version of the developed solution method should be able to solve this problem at least as well as the current version of the DE-VReP.

Bibliography

Ait-Ouahmed, A., Josselin, D., and Zhou, F. (2018). 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.

Autonomous cars (2018). Autonomous car forecasts. `http://www.driverless-future.com/?page_id=384`. Accessed: 2018-22-5.

Barth, M. and Todd, M. (1999). Simulation model performance analysis of a multiple station shared vehicle system. *Transportation Research Part C*, 7:237–259.

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

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

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

Bruglieri, M., Pezzella, F., and Pisacane, O. (2018). An Adaptive Large Neighborhood Search for relocating vehicles in electric carsharing services. *Discrete Applied Mathematics.*

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

Chen, S., Chen, R., Wang, G.-G., Gao, J., and Sangaiah, A. K. (2018). An adaptive large neighborhood search heuristic for dynamic vehicle routing problems. *Computers & Electrical Engineering.*

Chen, Z.-L. and Xu, H. (2006). Dynamic Column Generation for Dynamic Vehicle Routing with Time Windows. *Transportation Science*, 40:74–88.

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

Correia, G., J. D. and Atunes, D. (2014). The added value of accounting for users' flexibility and information on the potential of a station-based one-way car-sharing system. *Journal of Intelligent Transportation Systems*, 18(3):299–308.

Febbraro, A., Sacco, N., and Saeednia, M. (2012). One-Way Carsharing. *Transportation Research Record: Journal of the Transportation Research Board*, 2319:113–120.

Folkestad, C. A. and Hansen, N. Å. (2017). An optimization-simulation framework for vehicle and personnel relocations of one-way electric car-sharing systems with reservations. Master's thesis, Norwegian University of Science and Technology.

Frost and Sullivan (2014). Strategic Insight of the Global Carsharing Market. Technical report, Frost and Sullivan Research Service, San Antonio, Texas.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New Insertion and Postoptimization Procedures for the Traveling Salesman Problem. *Operations Research*, 40:1086–1094.

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

Hellem, S. S., Julsvoll, C. A., and Moan, M. (2017). A Rolling Horizon Solution Approach to the Dynamic Repositioning Problem in Free-Floating Carsharing Systems. Tech report, Norwegian University of Science and Technology.

Herbawi, W., Knoll, M., Kaiser, M., and Gruel, W. (2016). An evolutionary algorithm for the vehicle relocation problem in free floating carsharing. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 2873–2879.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* The University of Michigan Press.

Jorge, D., Correia, G. H. A., and Barnhart, C. (2014). Comparing Optimal Relocation Operations With Simulated Relocation Policies in One-Way Carsharing Systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1667–1675.

Kaspi, M., R. T. and Tzur, M. (2014). Parking reservation policies in one-way vehicle sharing systems. *Transportation Research Part B: Methodological*, 62:35–50.

Kathrin S. Kühne, T. A. R. and Breitner, M. H. (2016). An Optimization Model and a Decision Support System to Optimize Car Sharing Stations with Electric Vehicles. *Operations Research Proceedings*, 2014:313–320.

Kek, A., Cheu, R., and Chor, M. (2006). Relocation simulation model for multiple-station shared-use vehicle systems. *Transportation Research Record: Journal of the Transportation Research Board*, 1986:81–88.

Kek, A. G., Cheu, R. L., Meng, Q., and Fung, C. H. (2009). A decision support system for vehicle relocation operations in carsharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):149–158.

Kilby, P., Prosser, P., and Shaw, P. (1998). Dynamic VRPs: A Study of Scenarios. Technical report, University of Strathclyde: Glasgow, UK.

Kirchler, D. and Calvo, R. W. (2013). A Granular Tabu Search algorithm for the Dial-a-Ride Problem. *Transportation Research Part B: Methodological*, 56:120–135.

Li, X., Ma, J., Cui, J., Ghiasi, A., and Zhou, F. (2016). Design framework of large-scale one-way electric vehicle sharing systems: A continuum approximation model. *Transportation Research Part B: Methodological*, 88:21–45.

Loose, W. (2009). The environmental impacts of Car-Sharing use. https://ec.europa.eu/energy/intelligent/projects/sites/iee-projects/files/projects/documents/momo_car-sharing_f03_environmental_impacts_en.pdf. Accessed: 2017-12-6.

Martin, E. and Shaheen, S. (2016). 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.*

Martinez, L. M., Caetano, L., Eiró, T., and Cruz, F. (2012). An Optimisation Algorithm to Establish the Location of Stations of a Mixed Fleet Biking System: An Application to the City of Lisbon. *Procedia - Social and Behavioral Sciences*, 54:513–524.

Mukai, N. and Watanabe, T. (2005). Dynamic Location Management for On-Demand Car Sharing System. In *Knowledge-Based Intelligent Information and Engineering Systems: 9th International Conference, KES 2005, Part I*, pages 768–774. Springer Berlin Heidelberg.

Nagata, Y. and Bräysy, O. (2009). A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*, 37(5):333–338.

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

Penna, P. H. V., Subramanian, A., Ochi, L. S., Vidal, T., and Prins, C. (2018). A Hybrid Heuristic for a Broad Class of Vehicle Routing Problems with Heterogeneous Fleet. *Annals of Operations Research*, pages 1–70.

Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. (2011). A Review of Dynamic Vehicle Routing Problems. *European Journal of Operational Research*, 225(1):1–11.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002.

Psaraftis, H. N., Wen, M., and Kontovas, C. A. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31.

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

Secomandi, N. and Margot, F. (2009). Reoptimization Approaches for the Vehicle-Routing Problem with Stochastic Demands. *Operations Research*, 57:214–230.

Sethi, S. and Sorger, G. (1991). A Theory of Rolling Horizon Decision Making. *Annals of Operations Research*, 29(1):387–415.

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

Shaheen, S., Sperling, D., and Wagner, C. (1998). Carsharing in Europe and North America: Past, Present, and Future. *Transportation Quarterly*, 52(3):35–52.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, Department of Computer Science, University of Strathclyde, Glasgow, United Kingdom.

Simone Weikl, K. B. (2012). Relocation Strategies and Algorithms for free-floating Car Sharing Systems. *IEEE Intelligent Transportation Systems Magazine*, 5:100–111.

Thomas, B. W. (2007). Waiting Strategies for Anticipating Service Requests from Known Customer Locations. *Transportation Science*, 41(3):319–331.

Ulmer, M., Goodson, J., Mattfeld, D., and Thomas, B. (2017). Route-Based Markov Decision Processes for Dynamic Vehicle Routing Problems. Working paper.

UN (2017). World population prospects: The 2017 revision, key findings and advance tables. Tech Report, UN.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.

Weikl, S. and Bogenberger, K. (2015). A practice-ready relocation model for free-floating carsharing systems with electric vehicles – Mesoscopic approach and field trial results. *Transportation Research Part C: Emerging Technologies*, 57:206–223.

Wilson, N. and Colvin, N. (1977). *Computer Control of the Rochester Dial-A-Ride System.* CTS report. Massachusetts Institute of Technology, Center for Transportation Studies.

Yang, J., Jaillet, P., and Mahmassani, H. (2002). Real-Time Multi-Vehicle Truckload Pick-Up and Delivery Problems. *Transportation Science*, 38:135–148.

Appendix A

Task-Based Formulation

Table A.1: Sets used in the mathematical formulation

Sets	
$\mathcal{N}$	Set of nodes
$\mathcal{N}^C$	Set of charging nodes, $\mathcal{N}^C \subset \mathcal{N}$
$\mathcal{N}^P$	Set of parking nodes, $\mathcal{N}^P \subset \mathcal{N}$
$\mathcal{N}^{P-}$	Set of deficit parking nodes, $\mathcal{N}^{P-} \subset \mathcal{N}^P$
$\mathcal{N}^{PC}$	Set of parking nodes that has cars for charging, $\mathcal{N}^{PC} \subset \mathcal{N}^P$
$\mathcal{R}$	Set of car-moves
$\mathcal{R}_c$	Set of car-moves for car $c \in \mathcal{C}$
$\mathcal{R}_i^D$	Set of car-moves that ends in parking node $i \in \mathcal{N}^P$
$\mathcal{R}_i^C$	Set of car-moves that ends in a charging station and begins in node $i \in \mathcal{N}^P$
$\mathcal{R}_i^{DC}$	Set of car-moves that ends in charging node $i \in \mathcal{N}^C$
$\mathcal{P}$	Set of service operators
$\mathcal{M}$	Set of possible tasks
$\mathcal{C}$	Set of cars

Table A.2: Indices used in the mathematical formulation

Indices	
i, j	Node $i, j \in \mathcal{N}$
k	Operator $k \in \mathcal{K}$
m	Task number $m \in \mathcal{M}$
c	Cars $c \in \mathcal{C}$
r, v	Car-move $r, v \in \mathcal{R}$
$o(k)$	Origin node $o(k) \in \mathcal{N}$ for operator $k \in \mathcal{K}$
$o(r), d(r)$	Origin node $o(r) \in \mathcal{N}^P$ and destination node $d(r) \in \mathcal{N}$ for move $r \in \mathcal{R}$

Table A.3: Parameters used in the mathematical formulation

Parameters	
C^D	Benefit of reducing the deviation from ideal state by 1
C^{Ch}	Benefit of charging a car
C^{ET}	Cost of additional time used outside of planning period
C^R	Cost of handling
C^T	Cost of time use
T_r^H	Handling time for car-move $r \in \mathcal{R}$
T_r^S	Earliest start time for car-move $r \in \mathcal{R}$
T_k^S	Start time for operator $k \in \mathcal{K}$
T_{ij}	Travel time between node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$
$\overline{T}$	Length of planning period
$\overline{T}^L$	Maximum additional time allowed for last visit
S_i^{0-}	Initial deficit of rental cars in parking node $i \in \mathcal{N}^{P-}$
S_i^C	Initial number of insufficiently charged rental cars in parking node $i \in \mathcal{N}^P$
N_i^{CS}	Number of charging slots available in charging node $i \in \mathcal{N}^C$

Table A.4: Variables used in the mathematical formulation

Variables	
x_{krm}	1 if service operator $k \in \mathcal{K}$ performs move $r \in \mathcal{R}$ as task $m \in \mathcal{M}$
t_{km}	Start time for operator $k \in \mathcal{K}$ on task $m \in \mathcal{M}$
t_k^+	Positive deviation from time limit for operator $k \in \mathcal{K}$
t_k^-	Negative deviation from time limit for operator $k \in \mathcal{K}$ at the end of planning period
s_i^-	Negative deviation from ideal state in parking node $i \in \mathcal{N}^P$ at the end of planning period
s_i^C	Number of insufficiently charged cars in parking node $i \in \mathcal{N}^P$ at the end of planning period

A.1 Objective Function

$$\max z = \sum_{i \in \mathcal{N}^{P-}} C^D (S_i^{0-} - s_i^-) + \sum_{i \in \mathcal{N}^{PC}} C^{PC} (S_i^C - s_i^C) - \sum_{k \in \mathcal{K}} C^T t_{k|M|}$$
$$- \sum_{k \in \mathcal{K}} C^{ET} t_k^+ - \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} \sum_{m \in \mathcal{M}} C^H T_r^H x_{krm} \tag{A.1}$$

A.2 Constraints

A.2.1 Relocation of Rental Vehicles

$$\sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_c} \sum_{m \in \mathcal{M}} x_{krm} \leq 1 \qquad c \in \mathcal{C} \tag{A.2}$$

$$\sum_{r \in \mathcal{R}} x_{krm} \leq 1 \qquad k \in \mathcal{K}, m \in \mathcal{M} \tag{A.3}$$

$$\sum_{r \in \mathcal{R}} x_{kr(m+1)} \leq \sum_{r \in \mathcal{R}} x_{krm} \qquad k \in \mathcal{K}, m \in \mathcal{M} \setminus \{|\mathcal{M}|\} \tag{A.4}$$

A.2.2 Node State Balancing

$$\sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_i^D} \sum_{m \in \mathcal{M}} x_{krm} + s_i^- = S_i^{0-} \qquad i \in \mathcal{N}^{P-} \tag{A.5}$$

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_i^C}\sum_{m\in\mathcal{M}} x_{krm} + s_i^C = S_i^C \qquad i\in\mathcal{N}^{PC} \tag{A.6}$$

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_i^{DC}}\sum_{m\in\mathcal{M}} x_{krm} \le N_i^{CS} \qquad i\in\mathcal{N}^{C} \tag{A.7}$$

A.2.3 Time Tracking of Node Visits

$$t_{km} + T_r^H x_{krm} + \sum_{v\in\mathcal{R}} T_{d(r)o(v)} x_{kv(m+1)}$$

$$-M_r(1-x_{krm}) \le t_{k(m+1)} \qquad k\in\mathcal{K}, r\in\mathcal{R}, m\in\mathcal{M}\setminus\{|\mathcal{M}|\} \tag{A.8}$$

$$t_{km} \le t_{k(m+1)} \qquad k\in\mathcal{K}, m\in\mathcal{M}\setminus\{|\mathcal{M}|\} \tag{A.9}$$

$$T_r^S x_{krm} \le t_{km} \qquad k\in\mathcal{K}, r\in\mathcal{R}, m\in\mathcal{M} \tag{A.10}$$

$$(T_p^S + T_{o(k)o(r)})x_{kr1} \le t_{k1} \qquad k\in\mathcal{K}, r\in\mathcal{R} \tag{A.11}$$

$$t_{k|\mathcal{M}|} + \sum_{r\in\mathcal{R}} T_r^H x_{kr|\mathcal{M}|} + t_p^- - t_k^+ = \overline{T} \qquad k\in\mathcal{K} \tag{A.12}$$

$$t_{k|\mathcal{M}|} + \sum_{r\in\mathcal{R}} T_r^H x_{kr|\mathcal{M}|} \le \overline{T} + \overline{T}^L \qquad k\in\mathcal{K} \tag{A.13}$$

A.2.4 Binary, Non-negativity and Integer Definitions

$$x_{krm} \in \{0,1\} \qquad k\in\mathcal{K}, r\in\mathcal{R}, m\in\mathcal{M} \tag{A.14}$$

$$t_{km} \ge 0 \qquad k\in\mathcal{K}, m\in\mathcal{M} \tag{A.15}$$

$$t_k^+ \ge 0 \qquad k\in\mathcal{K} \tag{A.16}$$

$$t_k^- \ge 0 \qquad k\in\mathcal{K} \tag{A.17}$$

$$s_i^- \in \mathbb{Z}^+ \qquad i\in\mathcal{N}^{P-} \tag{A.18}$$

$$s_i^C \in \mathbb{Z}^+ \qquad i\in\mathcal{N}^{PC} \tag{A.19}$$

A.2.5 Big-M Calculation

$$M_r = \max_{v \in \mathcal{R},\, i \in \mathcal{N}^P \setminus \mathcal{N}^{P-}} T_{d(r)i} - (T_v^H + T_{d(v)i}) \qquad r \in \mathcal{R} \tag{A.20}$$

$$\tag{A.21}$$

Appendix B

Flow-Based Model Formulation

B.1 Formulation

Table B.1: Sets used in the mathematical formulation

Sets	
$\mathcal{N}$	Set of nodes
$\mathcal{N}^C$	Set of charging nodes, $\mathcal{N}^C \subset \mathcal{N}$
$\mathcal{N}^P$	Set of parking nodes, $\mathcal{N}^P \subset \mathcal{N}$
$\mathcal{N}^{P-}$	Set of deficit parking nodes, $\mathcal{N}^{P-} \subset \mathcal{N}^P$
$\mathcal{K}$	Set of operators
$\mathcal{C}$	Set of cars
$\mathcal{R}$	Set of car-moves
$\mathcal{R}_c$	Set of car-moves for car $c \in \mathcal{C}$, $\mathcal{R}_c \subset \mathcal{R}$
$\mathcal{R}^A$	Set of car-moves that are origin and final destination for $k \in \mathcal{K}$, $\mathcal{R}^A \subset \mathcal{R}$
$\mathcal{R}_i^D$	Set of car-moves that ends in parking node $i \in \mathcal{N}^P$, $\mathcal{R}_i^D \subset \mathcal{R}$
$\mathcal{R}_i^C$	Set of car-moves that ends in $j \in \mathcal{N}^C$ and begins in node $i \in \mathcal{N}^P$, $\mathcal{R}_i^C \subset \mathcal{R}$
$\mathcal{R}_i^{DC}$	Set of car-moves that ends in charging node $i \in \mathcal{N}^C$, $\mathcal{R}_i^{DC} \subset \mathcal{R}$

Table B.2: Indices used in the mathematical formulation

Indices	
i, j	Node $i, j \in \mathcal{N}$
k	Operator $k \in \mathcal{K}$
c	Cars $c \in \mathcal{C}$
r, v	Car-move $r, v \in \mathcal{R}$
$o(k), d(k)$	Origin, and destination car-move $o(k), d(k) \in \mathcal{R}$ for operator $k \in \mathcal{K}$
$o(r), d(r),$	Origin node $o(r) \in \mathcal{N}^P$ and destination node $d(r) \in \mathcal{N}$ for car-move $r \in \mathcal{R}$

Table B.3: Parameters used in the mathematical formulation

Parameters	
C^D	Benefit of reducing the deviation from ideal state
C^{Ch}	Benefit of charging a car
C^{ET}	Cost of additional time used outside of planning period
C^R	Cost of handling
C^T	Cost of time use
T_k^S	Start time for operator $k \in \mathcal{K}$
T_{ij}	Travel time between node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$
T_r^S	Earliest start time for car-move $r \in \mathcal{R}$
T_r^H	Handling time for car-move $r \in \mathcal{R}$
$\overline{T}$	Length of planning period
$\overline{T}^L$	Maximum additional time allowed for last visit
S_i^{0-}	Initial deficit of rental cars in parking node $i \in \mathcal{N}^{P-}$
S_i^C	Initial number of insufficiently charged rental cars in parking node $i \in \mathcal{N}^P$
N_i^{CS}	Number of charging slots available in charging node $i \in \mathcal{N}^C$

Table B.4: Variables used in the mathematical formulation

Variables	
x_{rvk}	1 if service operator $k \in \mathcal{K}$ travels directly from car-move $r \in \mathcal{R}$ to $v \in \mathcal{R}$
t_r	Start time of car-move $r \in \mathcal{R}$
t_k^+	Positive deviation from time limit for operator $k \in \mathcal{K}$
t_k^-	Negative deviation from time limit for operator $k \in \mathcal{K}$ at the end of planning period
s_i^-	Negative deviation from ideal state in parking node $i \in \mathcal{N}^P$ at the end of planning period
s_i^C	Number of insufficiently charged cars in parking node $i \in \mathcal{N}^P$ at the end of planning period

B.1.1 Objective Function

$$\max z = \sum_{i \in \mathcal{N}^{P-}} C^D (S_i^{0-} - s_i^-) + \sum_{i \in \mathcal{N}^{PC}} C^{PC} (S_i^C - s_i^C) - \sum_{k \in \mathcal{K}} C^T t_{d(k)} - \sum_{k \in \mathcal{K}} C^{ET} t_k^+ - \sum_{r \in \mathcal{R}} \sum_{v \in \mathcal{R}} \sum_{k \in \mathcal{K}} C^H T_r^H x_{rvk} \tag{B.1}$$

B.1.2 Relocation of Rental Vehicles

$$\sum_{r \in \mathcal{R}_c} \sum_{v \in \mathcal{R}} \sum_{k \in \mathcal{K}} x_{rvk} \leq 1 \qquad c \in \mathcal{C} \tag{B.2}$$

$$\sum_{v \in \mathcal{R}} x_{rvk} - \sum_{v \in \mathcal{R}} x_{vrk} = 0 \qquad r \in \mathcal{R} \setminus \mathcal{R}^A, k \in \mathcal{K} \tag{B.3}$$

$$\sum_{v \in \mathcal{R}} \sum_{k \in \mathcal{K}} x_{rvk} \leq 1 \qquad r \in \mathcal{R} \tag{B.4}$$

$$\sum_{r \in \mathcal{R}} x_{rd(k)k} = 1 \qquad k \in \mathcal{K} \tag{B.5}$$

$$\sum_{r \in \mathcal{R}} x_{o(k)rk} = 1 \qquad k \in \mathcal{K} \tag{B.6}$$

B.1.3 Node State Balancing

State Balance in Parking Nodes for Sufficiently Charged Rental Cars

$$\sum_{r \in \mathcal{R}_i^D} \sum_{v \in \mathcal{R}} \sum_{k \in \mathcal{K}} x_{rvk} + s_i^- = S_i^{0-} \qquad i \in \mathcal{N}^P \tag{B.7}$$

State Balance in Parking Nodes for Cars in Need of Charging

$$\sum_{r \in \mathcal{R}_i^C} \sum_{v \in \mathcal{R}} \sum_{k \in \mathcal{K}} x_{rvk} + s_i^C = S_i^C \qquad i \in \mathcal{N}^P \tag{B.8}$$

State Balance in Charging Nodes

$$\sum_{r \in \mathcal{R}_i^{DC}} \sum_{v \in \mathcal{R}} \sum_{k \in \mathcal{K}} x_{rvk} \leq N_i^{CS} \qquad i \in \mathcal{N}^C \tag{B.9}$$

B.1.4 Time Tracking of Node Visits

Time Usage for Routing of Service Operators

$$t_r + (T_r^H + T_{d(r)o(v)})x_{rvk} \leq t_v + (\overline{T} + \overline{T}^L)(1 - x_{rvk}) \qquad r \in \mathcal{R}, v \in \mathcal{R}, k \in \mathcal{K} \tag{B.10}$$

$$T_r^S \leq t_r \qquad r \in \mathcal{R} \tag{B.11}$$

$$T_k^S \leq t_{o(k)} \qquad k \in \mathcal{K} \tag{B.12}$$

Time Usage Outside Planning Period

$$t_{d(k)} + t_k^- - t_k^+ = \overline{T} + \overline{T}^L \qquad k \in \mathcal{K} \tag{B.13}$$

$$t_{d(k)} \leq \overline{T} \qquad k \in \mathcal{K} \tag{B.14}$$

(B.15)

B.1.5 Binary, Non-negativity and Integer Definitions

$$x_{rvk} \in \{0,1\} \qquad r \in \mathcal{R}, v \in \mathcal{R}, k \in \mathcal{K} \tag{B.16}$$

$$t_r \geq 0 \qquad r \in \mathcal{R} \tag{B.17}$$

$$t_k^+ \geq 0 \qquad k \in \mathcal{K} \tag{B.18}$$

$$t_k^- \geq 0 \qquad k \in \mathcal{K} \tag{B.19}$$

$$s_i^- \in \mathbb{Z}^+ \qquad i \in \mathcal{N}^{P-} \tag{B.20}$$

$$s_i^C \in \mathbb{Z}^+ \qquad i \in \mathcal{N}^{PC} \tag{B.21}$$

Appendix C

Solution Example to the E-VReP

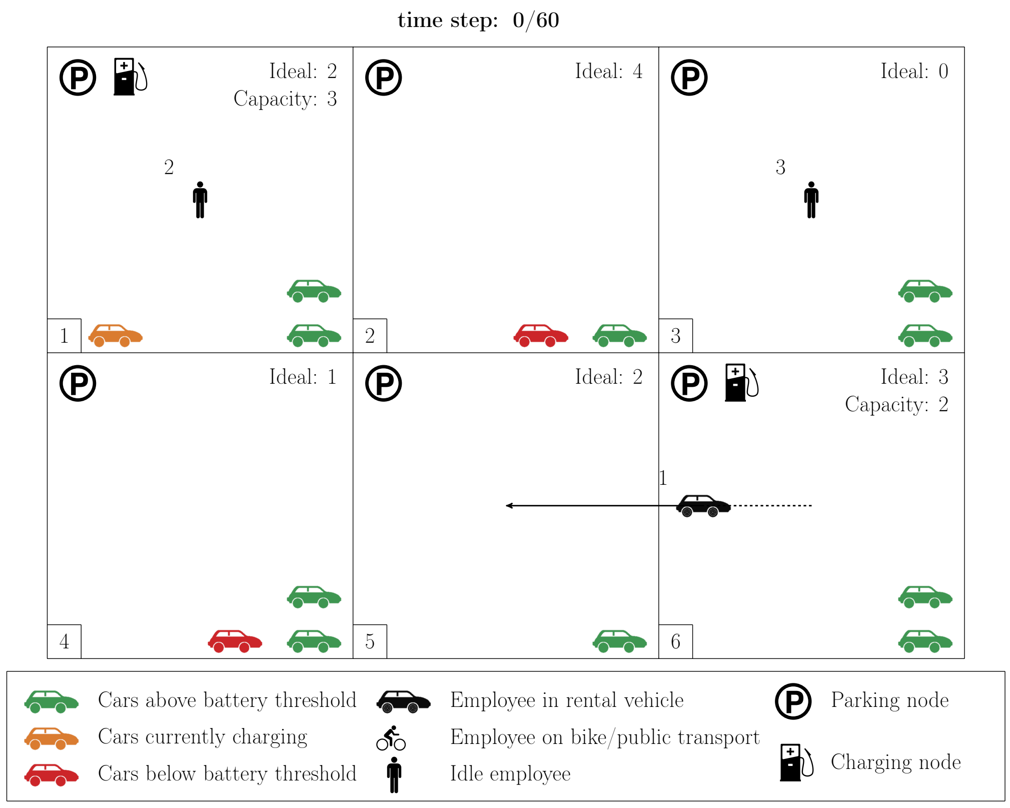


Figure C.1: Initial state

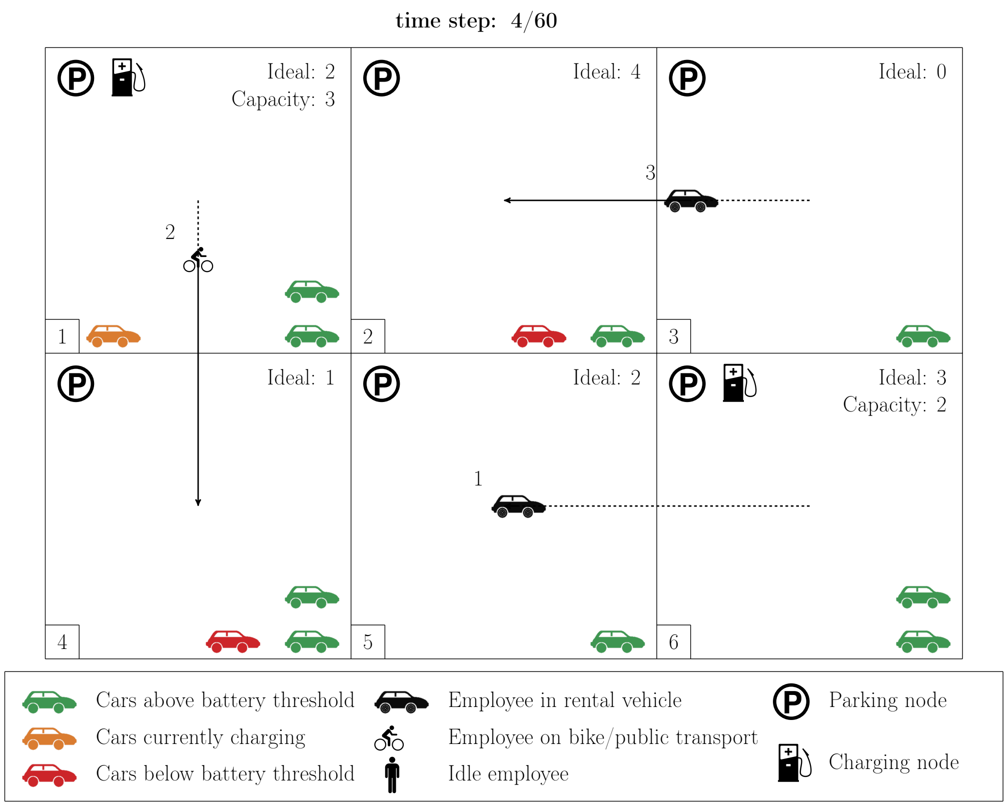


Figure C.2: Intermediate state 1

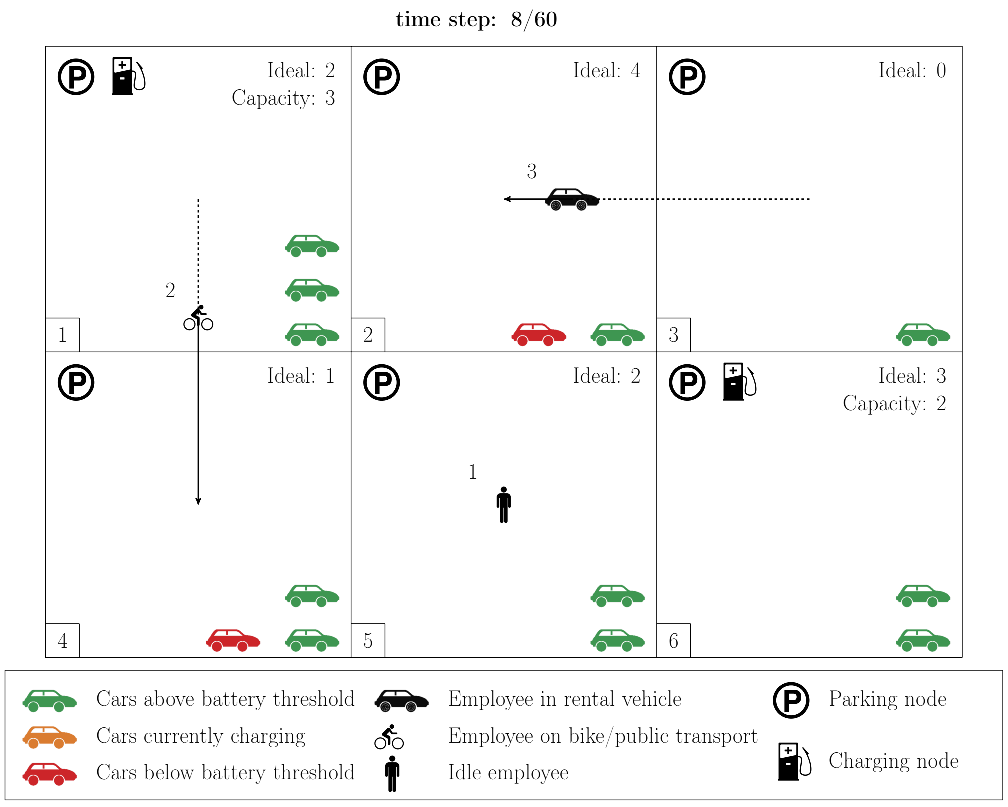


Figure C.3: Intermediate state 2

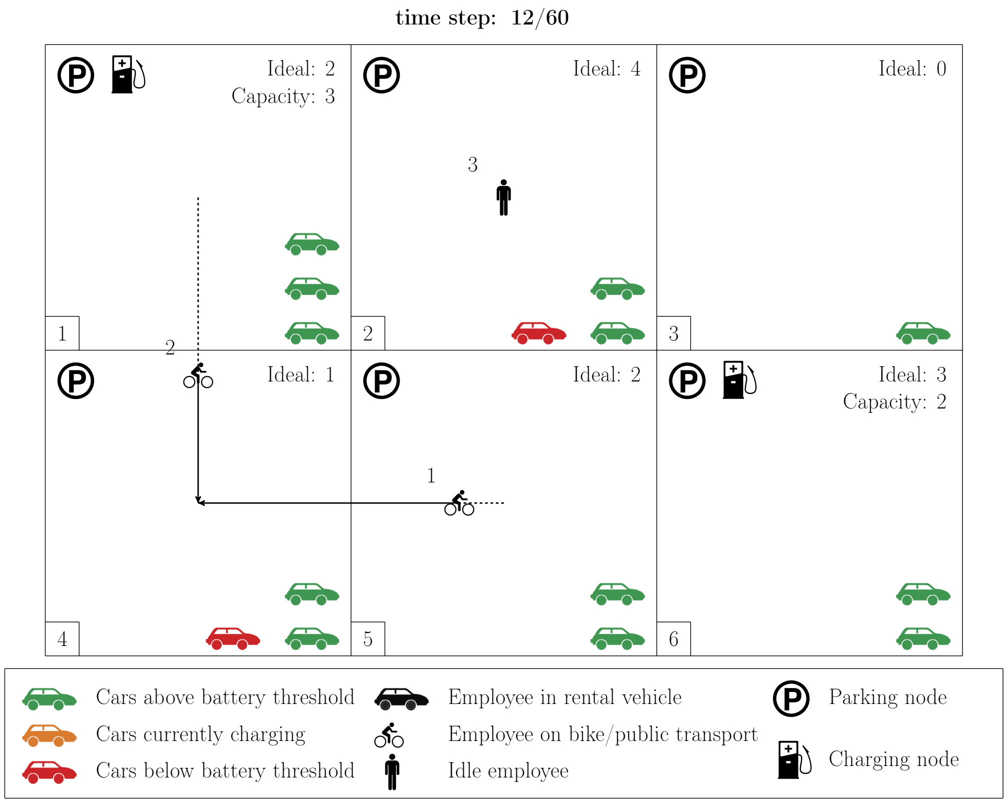


Figure C.4: Intermediate state 3

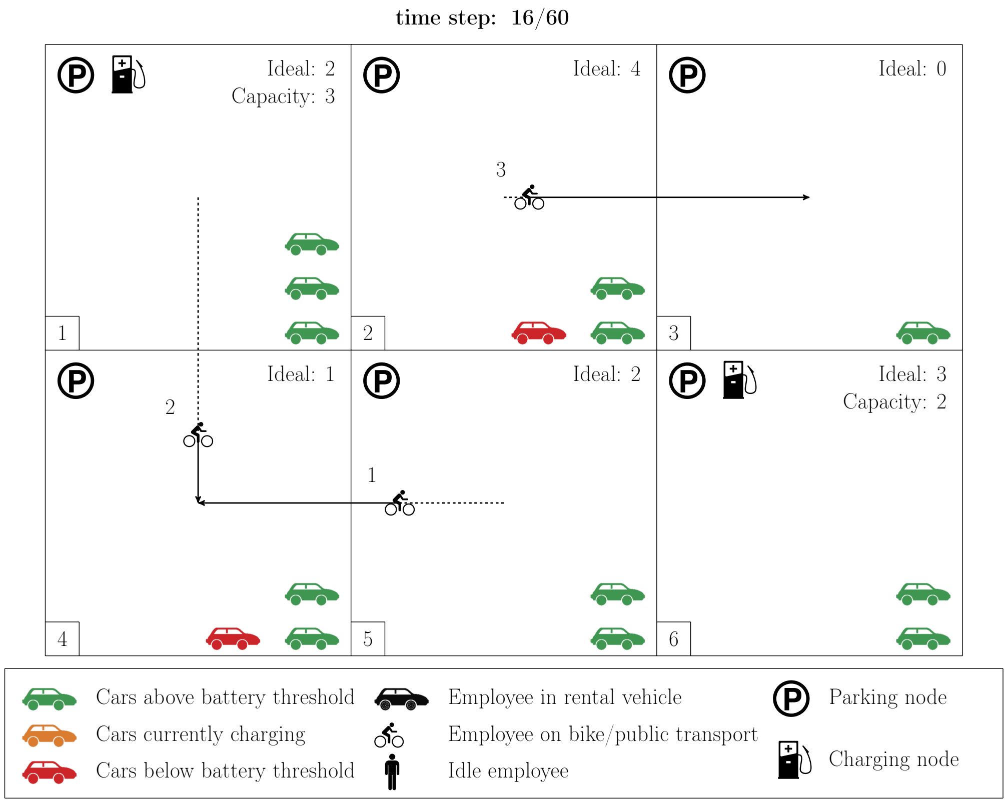


Figure C.5: Intermediate state 4

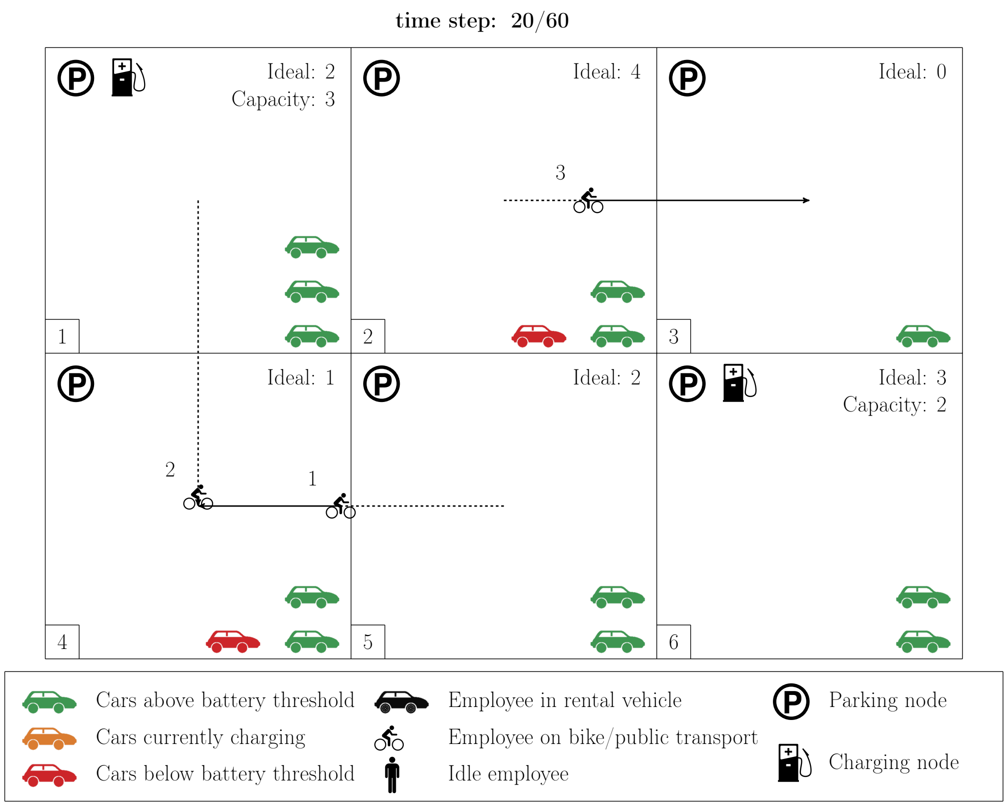


Figure C.6: Intermediate state 5

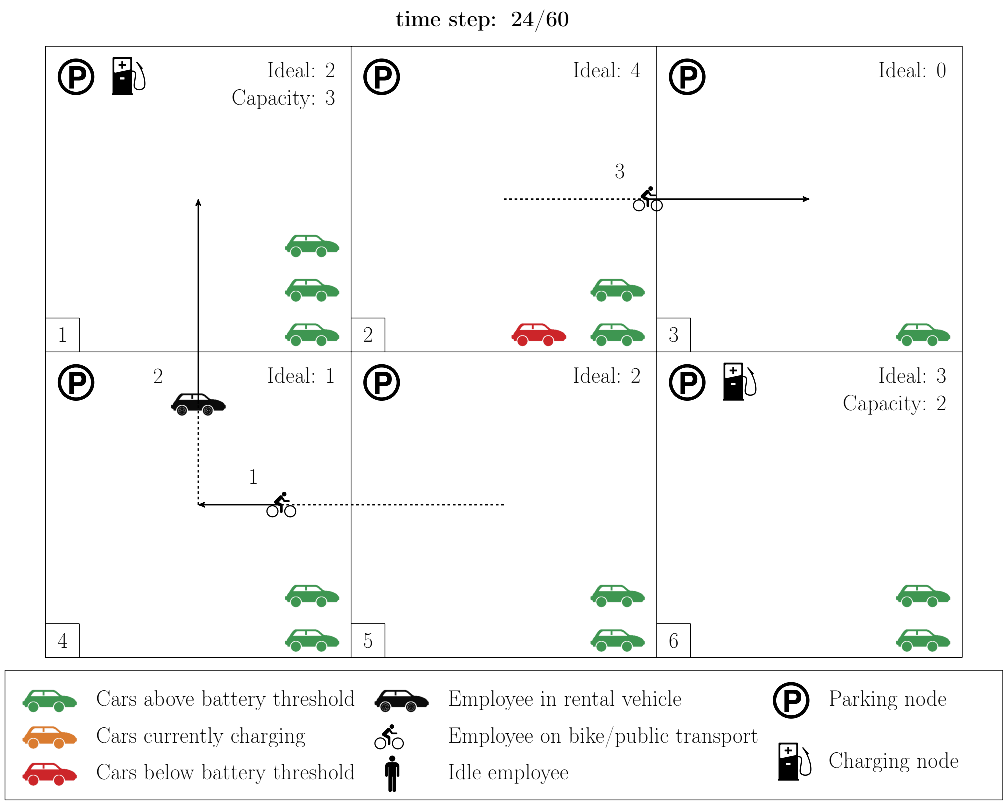


Figure C.7: Intermediate state 6

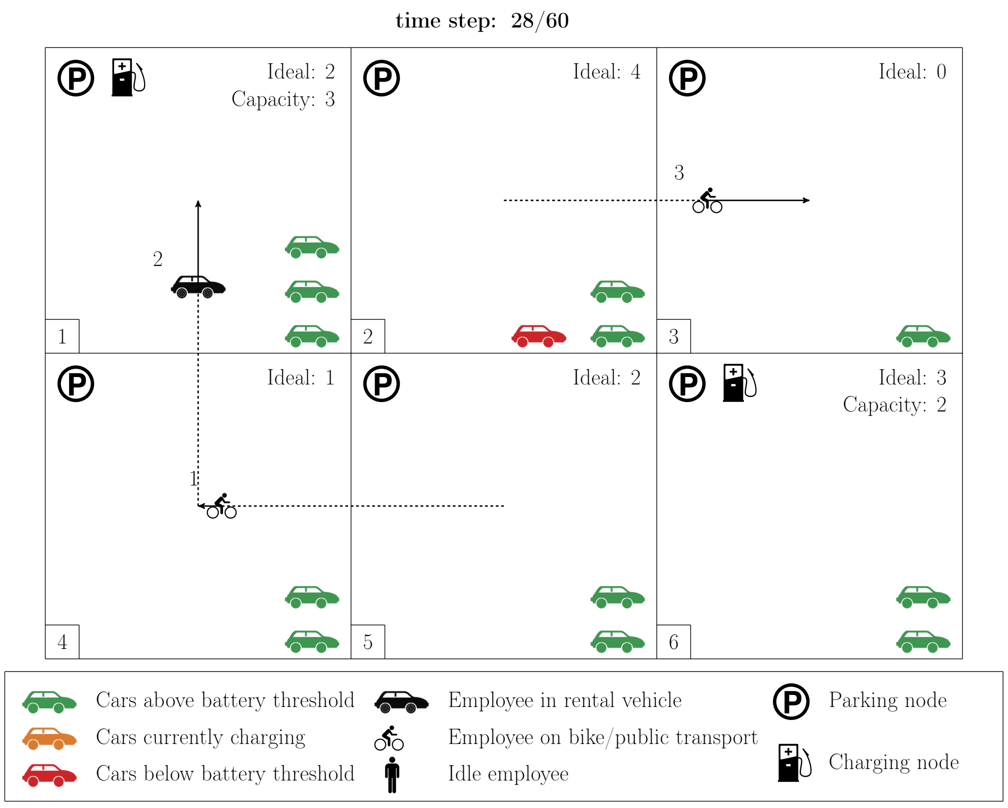


Figure C.8: Intermediate state 7

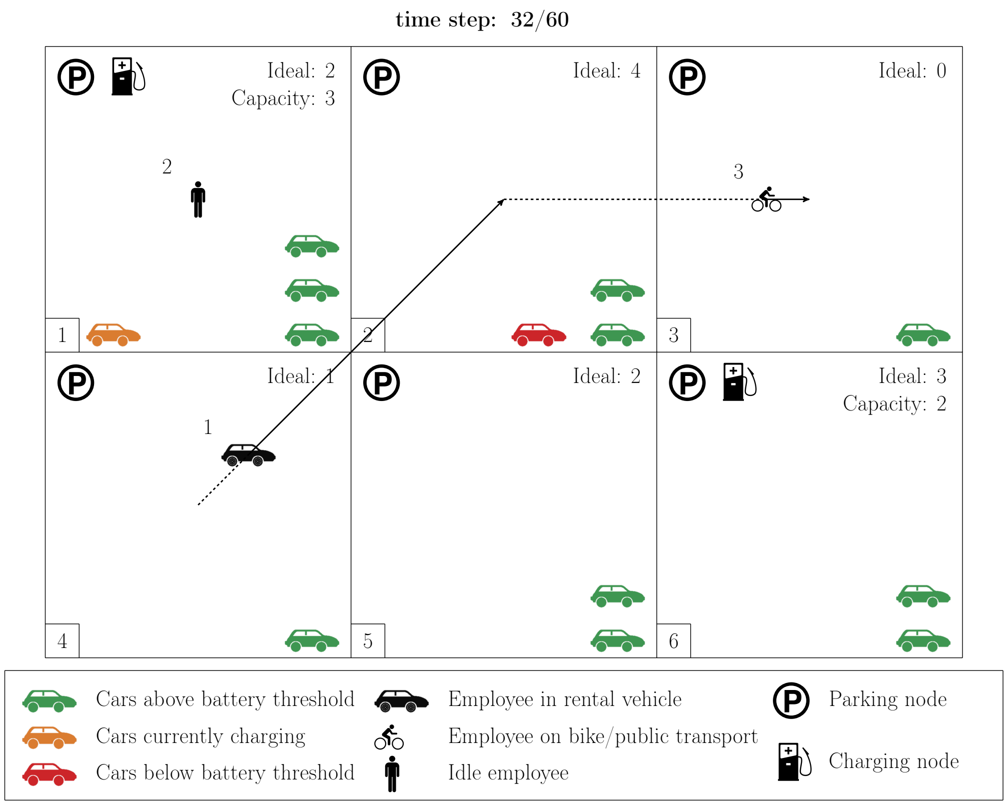


Figure C.9: Intermediate state 8

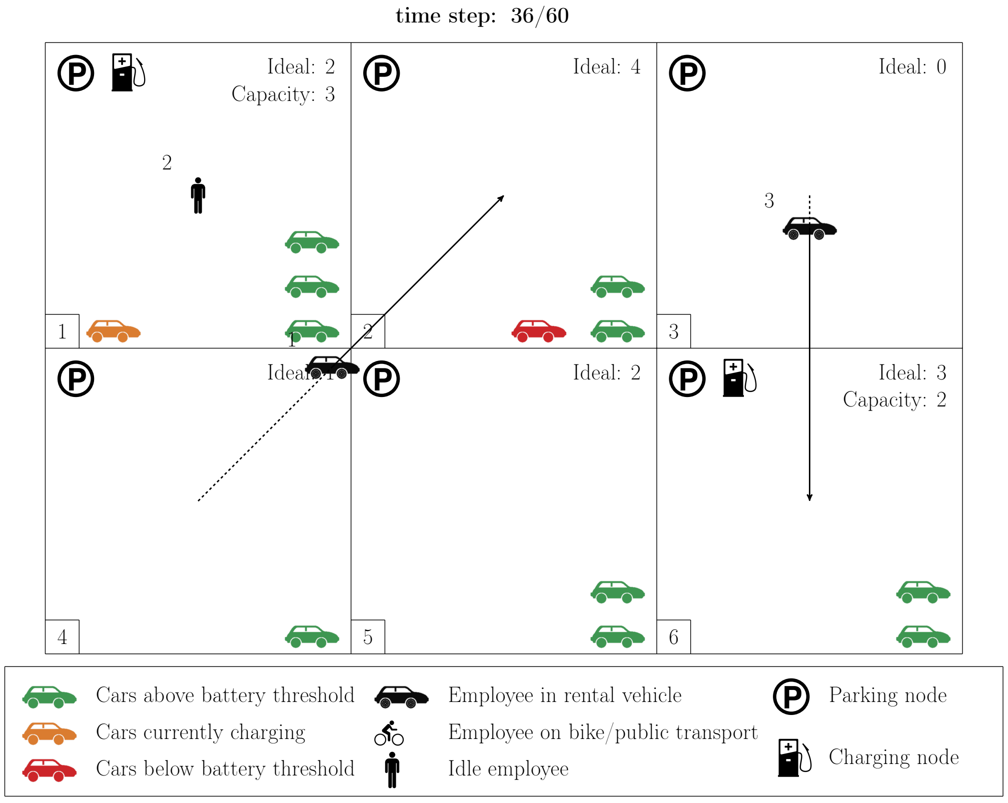


Figure C.10: Intermediate state 9

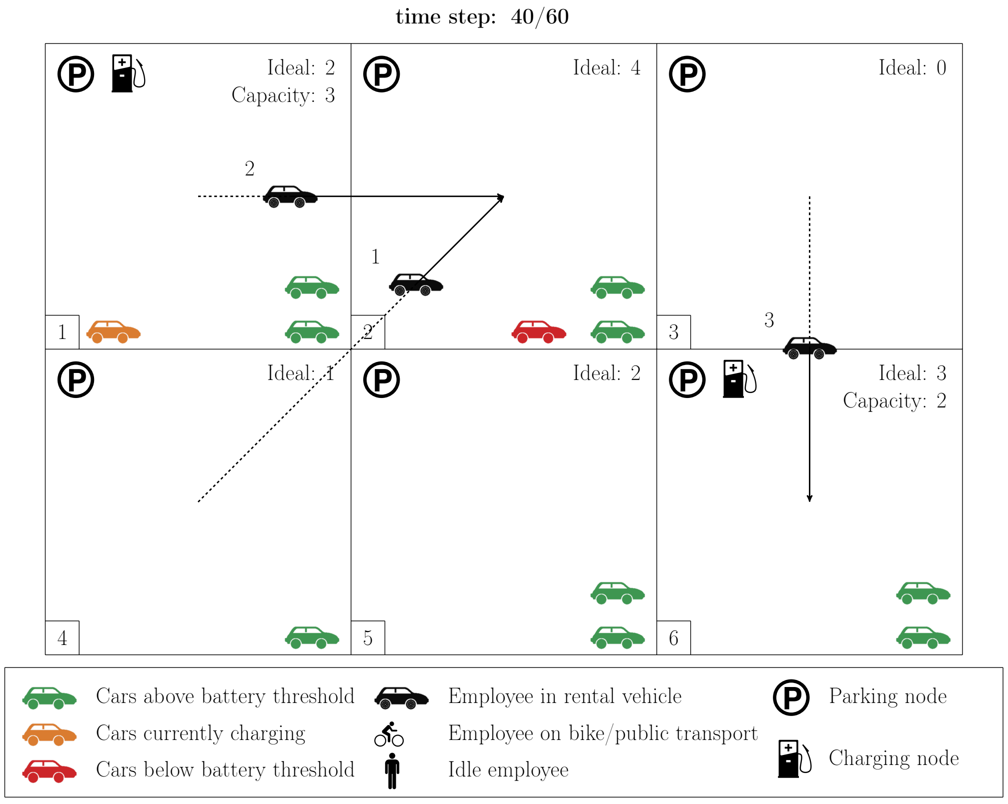


Figure C.11: Intermediate state 10

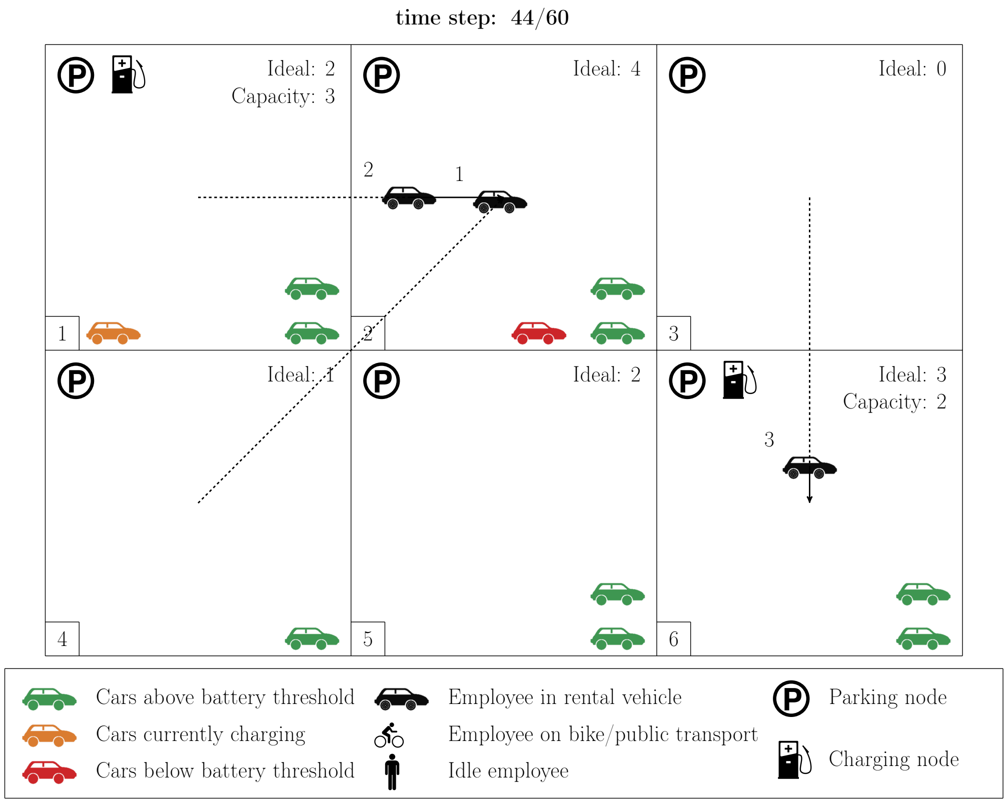


Figure C.12: Intermediate state 11

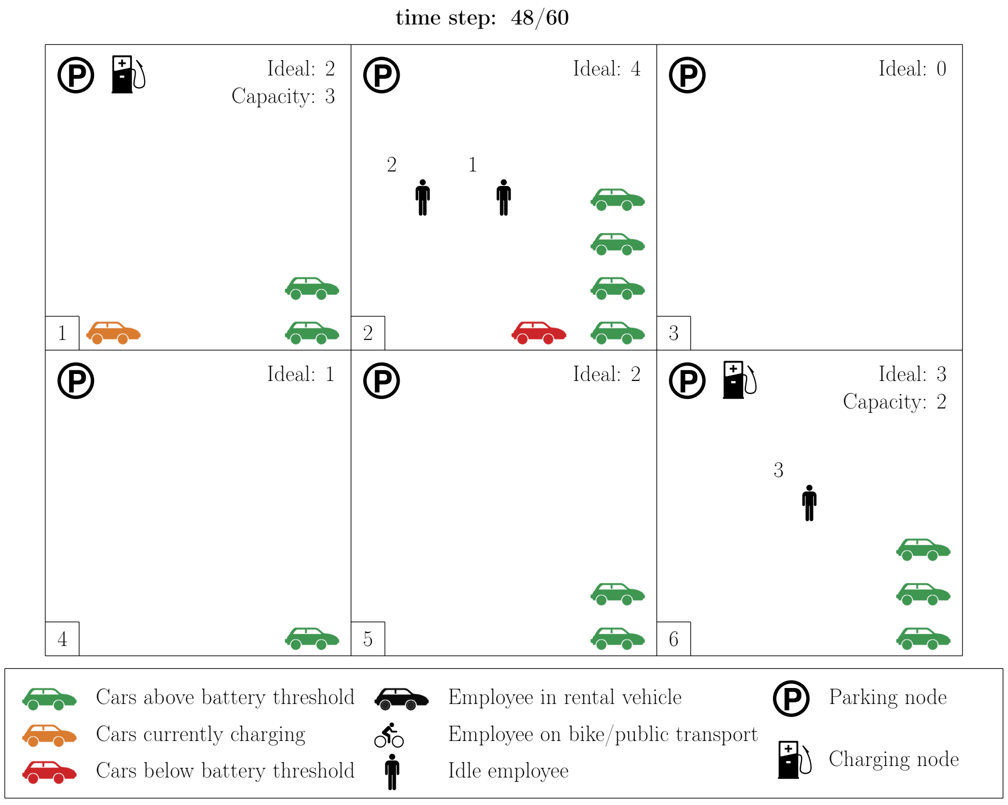


Figure C.13: Intermediate state 12

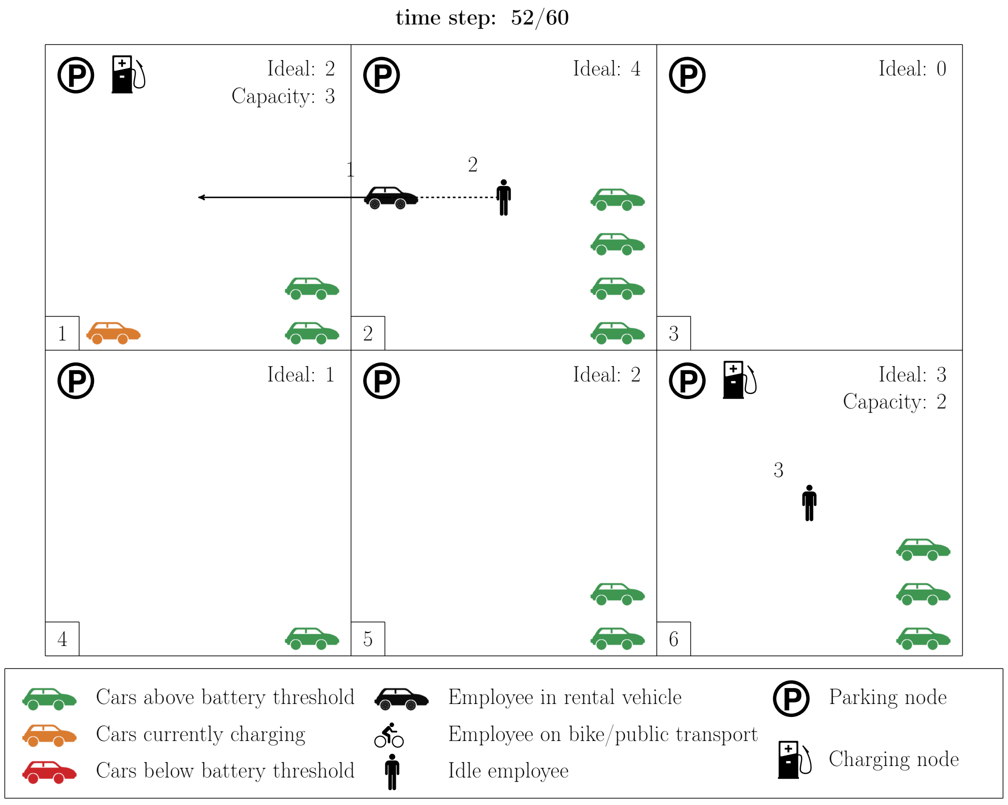


Figure C.14: Intermediate state 13

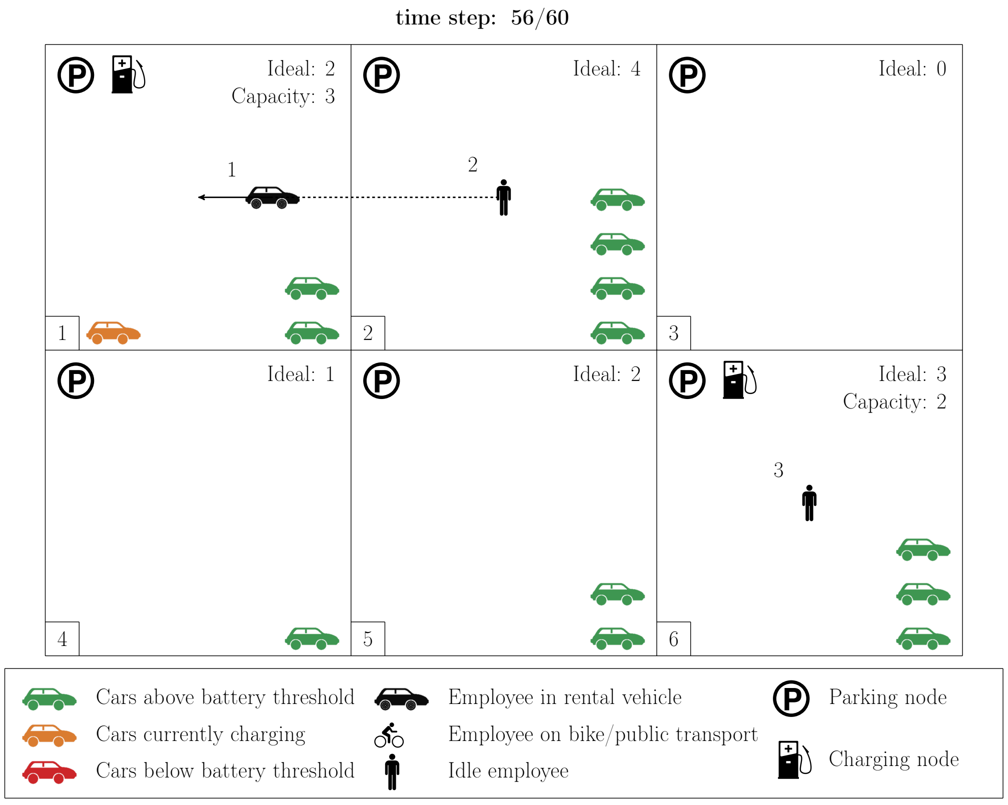


Figure C.15: Intermediate state 14

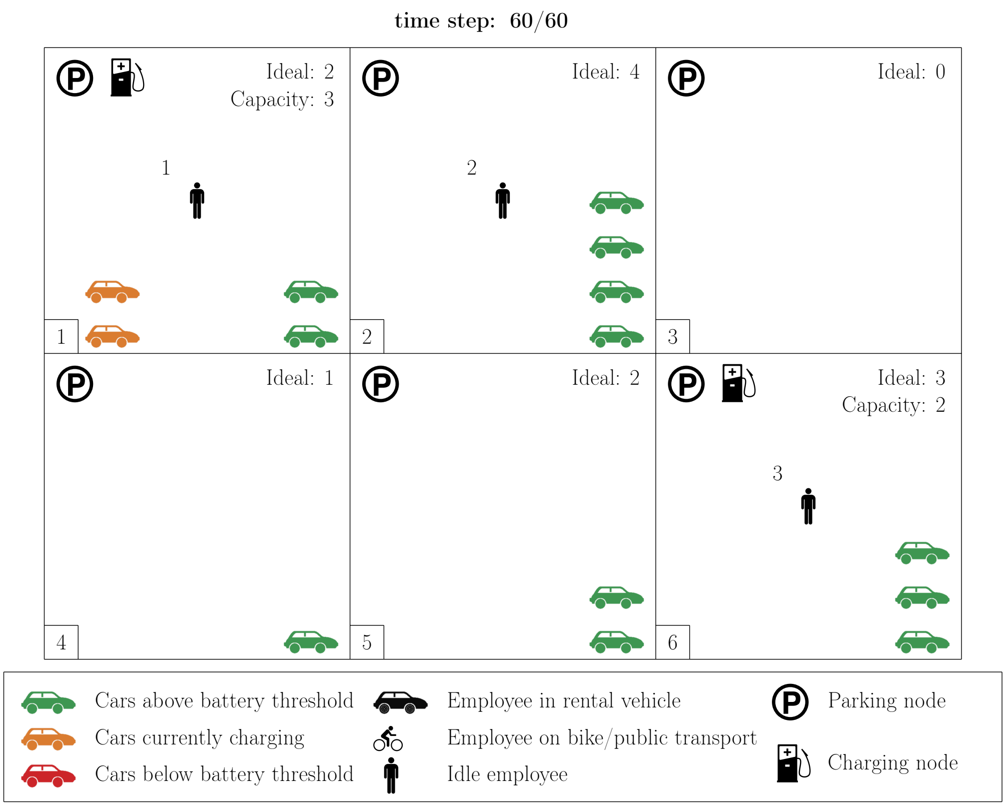


Figure C.16: Final state

Appendix D

Computational Study

D.1 Comparison of the Mathematical Models

Table D.1: Results of testing the task-based and flow-based MIP on the small test instances. Computational time and % gap is reported by Mosel.

Instance	**Task-based MIP**		**Flow-based MIP**	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3_a	0.68	0.0	4.28	0.0
6-3-3_b	0.76	0.0	4.02	0.0
6-3-3_c	1.66	0.0	16.24	0.0
Average	1.03	0.0	8.18	0.0
8-4-3_a	4.23	0.0	53.98	0.0
8-4-3_b	5	0.0	46.43	0.0
8-4-3_c	4.67	0.0	38.60	0.0
Average	4.63	0.0	46.34	0.0
10-7-3_a	250.22	0.0	7200	27.7
10-7-3_b	557.93	0.0	7200	N/A
10-7-3_c	703.68	0.0	7200	N/A
Average	503.94	0.0	7200	N/A
15-9-3_a	3201.99	0.0	7200	N/A
15-9-3_b	5730.11	0.0	7200	1004.6
15-9-3_c	7200	13.9	7200	N/A
Average	5377.40	4.6	7200	N/A

Green cells indicate best average values

Table D.2: Results of testing the task-based and flow-based MIP on the medium test instances. Computational time and % gap is reported by Mosel.

Instance	**Main Model**		**Alternative Model**	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %
15-12-5_a	7200	3.9	7200	N/A
15-12-5_b	7200	13.9	7200	120.0
15-12-5_c	7200	8.0	7200	157.1
Average	7200	8.6	7200	N/A
20-13-5_a	7200	19.3	7200	N/A
20-13-5_b	7200	13.7	7200	126.4
20-13-5_c	7200	10.2	7200	1004.6
Average	7200	14.4	7200	N/A
25-15-5_a	7200	18.5	7200	N/A
25-15-5_b	7200	20.3	7200	N/A
25-15-5_c	7200	39.3	7200	404.7
Average	7200	26.0	7200	N/A
30-18-5_a	7200	28.5	7200	695.8
30-18-5_b	7200	31.5	7200	N/A
30-18-5_c	7200	20.1	7200	N/A
Average	7200	26.7	7200	N/A

Green cells indicate best average values

D.2 Calibration of the ALNS

D.2.1 Calibrating the LST Generation

Table D.3: Average computational time and gap from best-known objective value for the three approaches to neighborhood generation, when running the ALNS with T^{MAX} = 3600s/180s respectively. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value. A negative gap versus MIP, indicates an improvement to the objective value reported in Tables D.1 and D.2.

Instance	**Full**				**Full Weighted**				**Random Weighted**			
	Comp. time (s)	Gap % (3600s)	Gap % (180s)	Gap % MIP	Comp. time (s)	Gap % (3600s)	Gap % (180s)	Gap % MIP	Comp. time (s)	Gap % (3600s)	Gap % (180s)	Gap % MIP
6-3-3	6	0.0	0.0	0.0	1	0.0	0.0	0.0	3.2	0.0	0.0	0.0
8-4-3	18.8	0.0	0.0	0.0	1.8	0.0	0.0	0.0	3.6	0.0	0.0	0.0
10-7-3	33.6	3.3	3.3	3.3	3.8	2.6	2.6	2.6	5.6	2.0	2.0	2.0
15-9-3	53.2	0.6	0.6	0.6	7.4	0.3	0.3	0.3	9.4	0.0	0.0	0.0
15-12-5	140.6	1.0	1.0	-3.0	19.8	2.1	2.1	-1.8	14.2	0.9	0.9	-3.1
20-13-5	197.6	0.2	1.5	-3.7	25.2	0.5	0.5	-3.5	20.2	0.1	0.1	-3.9
25-15-5	236	1.5	0.9	-16.6	33.8	1.0	1.0	-17.2	23.6	0.4	0.4	-17.9
30-18-5	315.2	0.7	2.5	-12.1	44.2	0.4	0.4	-12.5	30.6	0.3	0.3	-12.6
50-25-10	1464	2.8	5.9		194.4	2.4	2.4		89.6	1.5	1.5	
100-27-10	2123.6	4.4	6.5		405	1.7	1.7		161.6	1.5	1.7	
120-30-10	2719	7.4	9.0		577.4	3.0	3.7		294.8	1.7	2.5	
150-33-10	3334	6.2	10.2		874	2.2	3.2		497.8	2.4	3.6	
Average	886.8	2.3	3.5	-4.0	182.3	1.3	1.4	-4.0	96.2	0.9	1.1	-4.4

Green cells indicate best average values

D.2.2 Calibrating the LST Selection Criteria

Table D.4: Average computational time and gap from best-known objective value for the first improvement and best neighbor strategies, when running the ALNS with $T^{MAX} = 3600\text{s}/180\text{s}$ respectively. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	**First improvement**			**Best neighbor**		
	Comp. time (s)	Gap % (3600s)	Gap % (180s)	Comp. time (s)	Gap % (3600s)	Gap % (180s)
6-3-3	2.6	0.0	0.0	3.2	0.0	0.0
8-4-3	3.6	0.0	0.0	3.6	0.0	0.0
10-7-3	5.2	1.3	1.3	5.6	2.0	2.0
15-9-3	6.4	0.0	0.0	9.4	0.0	0.0
15-12-5	9.6	1.9	1.9	14.2	0.9	0.9
20-13-5	15.2	0.4	0.4	20.2	0.1	0.1
25-15-5	17.8	0.5	0.5	23.6	0.4	0.4
30-18-5	25.6	0.6	0.6	30.6	0.3	0.3
50-25-10	102.8	2.0	2.0	89.6	1.5	1.5
100-27-10	131.8	2.7	2.7	161.6	1.5	1.7
120-30-10	185.6	2.7	3.6	294.8	1.7	2.5
150-33-10	433.4	3.5	3.5	497.8	2.4	3.6
Average	78.3	1.3	1.4	96.2	0.9	1.1

Green cells indicate best average values

D.2.3 Calibrating the Termination Criteria

Table D.5: Average computational time and gap from best-known objective value when running the ALNS with $I^R = 75/100/125/150(000)$. All results are averages over five runs, and all runs are independent from one another. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	$I^R = 75000$		$I^R = 100000$		$I^R = 125000$		$I^R = 150000$	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	2.6	🟩 0.0	3.2	🟩 0.0	3.4	🟩 0.0	4.2	🟩 0.0
8-4-3	3.6	🟩 0.0	3.6	🟩 0.0	4.8	🟩 0.0	5.6	🟩 0.0
10-7-3	5.8	2.0	5.6	2.0	8.2	🟩 0.7	8.6	1.3
15-9-3	10.2	🟩 0.0	9.4	🟩 0.0	10.6	🟩 0.0	12.2	🟩 0.0
15-12-5	13	1.7	14.2	🟩 0.9	15.6	1.3	19.2	1.1
20-13-5	14.4	0.2	20.2	0.1	20.4	0.1	23.6	🟩 0.1
25-15-5	19.2	0.4	23.6	0.4	20.8	🟩 0.3	28.4	0.4
30-18-5	27	0.9	30.6	🟩 0.3	33.4	0.4	36	0.7
50-25-10	94	1.9	89.6	🟩 1.5	106.6	1.9	104.4	1.9
100-27-10	150	1.9	164.2	1.7	180	1.5	176.8	🟩 1.5
120-30-10	163.8	2.2	173.2	2.5	173.4	🟩 2.1	180	2.1
150-33-10	171.6	3.7	180	3.6	180	🟩 3.4	180	3.4
Average	56.3	1.2	59.8	1.1	63.1	🟩 1.0	65	1.0

Green cells indicate best average values

D.2.4 Calibrating the Neighborhood Size

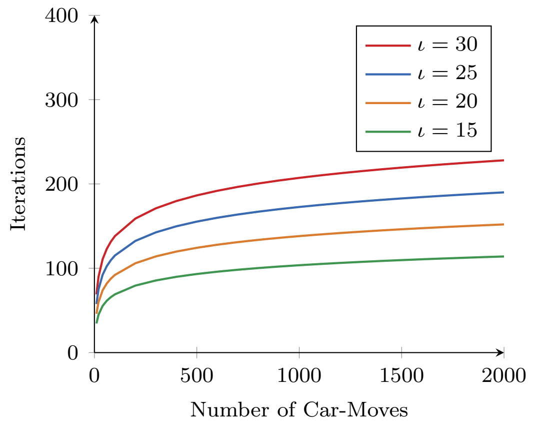


Figure D.1: Calibration of M^{MAX}: An illustration on how M^{MAX} increases with the problem size for different values of ι.

Table D.6: Average computational time and gap from best-known objective value when running the ALNS with scaling values $\iota = 15/20/25/30$ for M^{MAX}. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	**ι = 15**		**ι = 20**		**ι = 25**		**ι = 30**	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	2.2	🟩 0.0	3.2	🟩 0.0	3.2	🟩 0.0	4	🟩 0.0
8-4-3	3.2	🟩 0.0	4.2	🟩 0.0	4.4	🟩 0.0	5	🟩 0.0
10-7-3	5.4	2.0	7.4	2.6	6.6	🟩 0.7	8.8	2.0
15-9-3	9	🟩 0.0	10.8	🟩 0.0	9.8	🟩 0.0	11.4	🟩 0.0
15-12-5	14	1.3	19	🟩 0.5	19	0.9	26.4	1.0
20-13-5	20.8	0.2	26.6	0.2	23.4	🟩 0.1	32.8	0.1
25-15-5	26	0.3	30	0.5	28.6	0.4	42.8	🟩 0.2
30-18-5	29.8	0.6	39.8	🟩 0.2	37.4	0.4	53.4	0.3
50-25-10	99.4	1.9	143	1.7	130	1.8	165.2	🟩 1.5
100-27-10	179.2	1.4	180	1.0	180	🟩 1.0	180	1.1
120-30-10	180	2.6	180	2.8	180	🟩 2.1	180	2.1
150-33-10	180	3.1	180	🟩 2.9	180	3.1	180	3.3
Average	62.4	1.1	68.7	1.0	66.9	🟩 0.9	74.2	1.0

Green cells indicate best average values

D.2.5 Calibrating the LNS Criteria

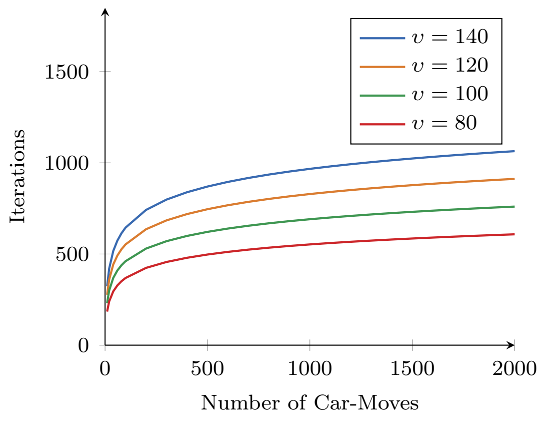


Figure D.2: An illustration on how I^{Des} may increase with the problem size based on different value of υ.

Table D.7: Average computational time and gap from best-known objective value when running the ALNS with scaling values $\upsilon = 80/100/120/140$ for I^R. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	υ = **80**		υ = **100**		υ = **120**		υ = **140**	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	3.4	0.0	3.6	0.0	2.8	0.0	3	0.0
8-4-3	4.4	0.0	5	0.0	4	0.0	4.6	0.0
10-7-3	6.8	1.3	7.8	1.3	6.6	0.7	7.8	2.0
15-9-3	10.4	0.0	10.6	0.0	10.2	0.0	11	0.0
15-12-5	16.6	1.3	20.8	1.0	19.2	0.8	21	0.9
20-13-5	26.4	0.1	29	0.1	25	0.1	29.4	0.2
25-15-5	28.4	0.4	35.8	0.4	26.8	0.4	33.4	0.5
30-18-5	36.4	0.1	44	0.3	36.6	0.3	40.2	0.3
50-25-10	152.8	1.7	148.8	1.8	128.4	1.8	152.2	1.8
100-27-10	180	1.5	180	1.8	179.8	1.3	180	0.9
120-30-10	180	2.0	180.2	1.9	179.2	1.8	180	2.2
150-33-10	180.6	2.2	180.2	2.6	180.4	2.2	180	2.0
Average	68.9	0.9	70.5	0.9	66.6	0.8	70.2	0.9

Green cells indicate best average values

D.2.6 Calibrating the Weight Update Criteria

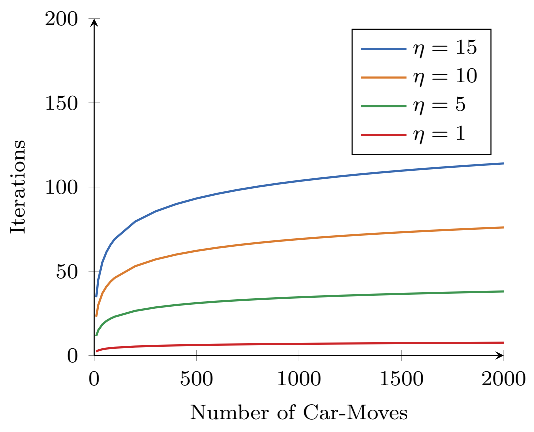


Figure D.3: Calibration of the Weight Update Criteria: An illustration on how I^W may increase with the problem size based on different values of η.

Table D.8: Average computational time and gap from best-known objective value when running the ALNS with scaling values $\eta = 1/5/10/15$ for I^W. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	$\eta = 1$		$\eta = 5$		$\eta = 10$		$\eta = 15$	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	5.6	0.0	2.8	0.0	2.6	0.0	2.6	0.0
8-4-3	5.8	0.0	3.8	0.0	3.6	0.0	3.4	0.0
10-7-3	10.6	0.0	7.4	0.0	6.4	0.7	6.6	0.7
15-9-3	15.8	0.0	8.4	0.0	9.0	0.0	8.8	0.0
15-12-5	32.7	1.7	18.6	0.5	16.6	1.3	16.6	1.1
20-13-5	35.8	0.5	24.6	0.3	23.6	0.1	23.6	0.1
25-15-5	47.2	0.5	25.2	0.5	31.4	0.2	26.8	0.3
30-18-5	62.2	0.5	39.8	0.2	39.8	0.3	39	0.1
50-25-10	178.4	1.5	108.0	1.8	122.8	1.2	111.8	1.7
100-27-10	180	1.5	171.4	0.6	167.2	1.0	172.8	0.7
120-30-10	180	1.7	180	1.7	180	1.9	180	1.8
150-33-10	181.2	2.5	180	2.2	180	2.3	180	2.3
Average	77.9	0.9	64.2	0.6	65.3	0.7	64.4	0.7

Green cells indicate best average values

D.2.7 Calibrating the Tabu List

Table D.9: Average computational time and gap from best-known objective value when running the ALNS with different values for I^B and I^S. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	$I^S = 1$, $I^B = 2$		$I^S = 2$, $I^B = 4$		$I^S = 3$, $I^B = 6$		$I^S = 4$, $I^B = 8$	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	2.4	0.0	2.8	0.0	2.6	0.0	2.4	0.0
8-4-3	3.2	0.0	3.8	0.0	4	0.0	3.8	0.0
10-7-3	6.4	0.7	7.4	0.0	8.6	0.0	6.6	1.3
15-9-3	9.6	0.0	8.4	0.0	9.6	0.0	9.6	0.0
15-12-5	17.6	1.8	18.6	0.5	20.2	0.9	17	1.0
20-13-5	24	0.1	24.6	0.3	29	0.1	22.4	0.1
25-15-5	27.4	0.3	25.2	0.5	29.4	0.3	30.8	0.2
30-18-5	39.6	0.2	39.8	0.2	43.4	0.2	38.6	0.2
50-25-10	131.4	1.8	108	1.8	117.6	1.0	110.6	1.7
100-27-10	168.4	0.9	171.4	0.6	175.6	0.9	161.8	1.0
120-30-10	180	1.9	180	1.7	180	1.4	180	1.8
150-33-10	180	2.2	180	2.2	180	2.1	180	2.2
Average	65.9	0.8	64.2	0.6	66.6	0.6	63.7	0.8

Green cells indicate best average values

D.2.8 Calibrating the LNS Size

Table D.10: LNS factor calibration. Average computational time and gap from best-known objective value when running the ALNS with different values for Γ. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	$\Gamma = \mathbf{0.2}$		$\Gamma = \mathbf{0.4}$		$\Gamma = \mathbf{0.6}$	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	2.8	0.0	2.6	0.0	2.6	0.0
8-4-3	4	0.0	4.4	0.0	3.4	0.0
10-7-3	6.6	1.3	8.6	0.0	8	0.7
15-9-3	11.2	0.0	9.6	0.0	9.2	0.0
15-12-5	18.2	0.8	20.2	0.9	18.6	1.1
20-13-5	24.4	0.1	29	0.1	25.8	0.1
25-15-5	27.4	0.3	29.4	0.3	30	0.3
30-18-5	36.2	0.1	43.4	0.2	44.4	0.1
50-25-10	111.8	1.4	117.6	1.0	137.6	1.7
100-27-10	156.8	1.1	175.6	0.9	180	1.9
120-30-10	173.6	1.7	180	1.4	180	1.7
150-33-10	180	2.7	180	2.1	180	2.7
Average	62.8	0.8	66.6	0.6	68.3	0.9

Green cells indicate best average values

D.2.9 Calibrating the Weight Update Parameters

Table D.11: Weights update factors calibration. Average computational time and gap from best-known objective value when running the ALNS with different configurations for α, R_Q^G and R_U^G. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	α = **0.1**						α = **0.2**					
	R_Q^G, R_U^G = **13**		R_Q^G, R_U^G = **23**		R_Q^G, R_U^G = **33**		R_Q^G, R_U^G = **13**		R_Q^G, R_U^G = **23**		R_Q^G, R_U^G = **33**	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	3	0.0	2	0.0	2.6	0.0	3	0.0	3.8	0.0	3.4	0.0
8-4-3	4	0.0	3.2	0.0	4	0.0	4	0.0	4.6	0.0	4.6	0.0
10-7-3	6.6	1.3	6.4	0.0	8.6	0.0	6.6	0.7	8.4	2.0	7.8	2.0
15-9-3	8.8	0.0	7.8	0.0	9.6	0.0	8.8	0.0	10.8	0.0	9.8	0.0
15-12-5	21.4	1.0	17.2	0.5	20.2	0.9	21.4	1.0	23	0.6	24	0.8
20-13-5	29.4	0.1	22.2	0.1	29	0.1	29.4	0.1	29.2	0.1	33.6	0.3
25-15-5	30.2	0.3	25	0.4	29.4	0.3	30.2	0.3	39.2	0.4	38.2	0.3
30-18-5	39.8	0.2	36.2	0.1	43.4	0.2	39.8	0.2	49.6	0.2	46.8	0.3
50-25-10	122.2	1.4	93.8	1.6	117.6	1.0	122.2	1.4	144.4	1.7	128	1.5
100-27-10	159.4	1.1	178.6	0.4	175.6	0.9	159.4	1.1	180	0.7	177.2	1.3
120-30-10	180	1.8	180	0.8	180	1.4	180	1.8	180	1.8	180	1.6
150-33-10	180	2.9	180	2.0	180	2.1	180	2.9	180	2.3	180	2.3
Average	65.2	0.9	62.7	0.5	66.7	0.6	65.4	0.8	71.1	0.8	69.5	0.9

Green cells indicate best average values

Table D.12: Weights update factors calibration. Average computational time and gap from best-known objective value when running the ALNS with different configurations for R_Q^N, R_Q^L and R_U^L. All results are averages over five runs. Gap is calculated as the difference between the best-known objective value, and the average objective value.

Instance	R_Q^L, R_U^L = **8**				R_Q^L, R_U^L = **13**				R_Q^L, R_U^L = **18**			
	R_Q^N = **1**		R_Q^N = **2**		R_Q^N = **1**		R_Q^N = **2**		R_Q^N = **1**		R_Q^N = **2**	
	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %	Comp. time (s)	Gap %
6-3-3	3	0.0	3	0.0	2	0.0	3	0.0	3	0.0	3	0.0
8-4-3	3.4	0.0	4.2	0.0	3.2	0.0	4.4	0.0	4	0.0	4.2	0.0
10-7-3	7.4	0.7	8.8	0.7	6.4	0.0	8.8	0.7	7.4	1.3	7.2	0.7
15-9-3	8.8	0.0	9.8	0.0	7.8	0.0	11	0.0	8.6	0.0	9.4	0.0
15-12-5	19.6	1.5	23	1.8	17.2	0.5	25	0.7	20.6	0.8	23.6	1.0
20-13-5	27.8	0.1	25.4	0.2	22.2	0.1	31.6	0.3	27	0.3	28.8	0.1
25-15-5	33.8	0.4	34.2	0.4	25	0.4	38.2	0.4	30.8	0.5	30.6	0.3
30-18-5	41.2	0.2	46.6	0.3	36.2	0.1	52.6	0.1	41.8	0.1	41.4	0.3
50-25-10	102.8	1.7	130.8	1.6	93.8	1.6	133.4	1.5	116.2	1.4	129.8	1.8
100-27-10	168	0.7	180	1.2	178.6	0.4	180	0.9	171.8	0.5	180	0.6
120-30-10	180	1.4	180	1.8	180	0.8	180	1.6	176.6	1.7	180	1.3
150-33-10	180.2	2.3	180.2	2.4	180	2.0	180	2.8	180.2	1.9	180	2.3
Average	64.7	0.7	68.8	0.9	62.7	0.5	70.7	0.7	65.7	0.7	68.2	0.7

Green cells indicate best average values

D.2.10 Final Remarks

Table D.13: ALNS: Final Parameter Values

Parameter	Value	Description
T^{MAX}	180	Max running time (seconds)
B^{INIT}	2	Initial tabu list size
B^{MIN}	2	Minimal tabu list size
B^{MAX}	1024	Maximal tabu list size
I^{R}	125000	Max number of iteration without improvement
I^{W}	$5 \ln \|C\|$	The number of iterations before the LSO weights are updated
I^{DES}	$120 \ln \|C\|$	Iterations without global improvement before destroy and repair
I^{B}	6	Iterations without local improvement before increasing the tabu list size
I^{S}	3	Iterations with local improvements before decreasing the tabu list size
M^{MAX}	$25 \ln \|C\|$	Neighborhood size
Γ	0.4	The destroy/repair factor
R_Q^N	1	LSO score for finding a new local solution
R_Q^G	23	LSO score for finding a new global best solution
R_Q^L	13	LSO score for finding a new better local solution
R_U^G	23	Destroy and repair score for finding a better global solution
R_U^L	13	Destroy and repair score for finding a new and better local solution
α	0.1	Update factor for both LSO and repair and destroy weights
$\omega_1 \ldots \omega_5$	0.315, 0.315, 0.315, 0.005, 0.05	Weights for Shaw Removal