UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

A DECISION-MAKING SYSTEM FOR MANAGING ELECTRIC VEHICLE FLEETS SUBJECT TO MULTIPLE OPERATIONAL CONSTRAINTS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

JUAN PABLO FUTALEF GALLARDO

PROFESOR GUÍA:
MARCOS ORCHARD CONCHA

PROFESOR CO-GUÍA:
DIEGO MUÑOZ CARPINTERO

MIEMBROS DE LA COMISIÓN:
DORIS SÁEZ HUEICHAPAN
FERNANDO AUAT CHEEIN

SANTIAGO DE CHILE
2021

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA, Y AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: JUAN PABLO FUTALEF GALLARDO
FECHA: 2021
PROF. GUÍA: MARCOS ORCHARD CONCHA
PROF. CO-GUÍA: DIEGO MUÑOZ CARPINTERO

A DECISION-MAKING SYSTEM FOR MANAGING ELECTRIC VEHICLE FLEETS SUBJECT TO MULTIPLE OPERATIONAL CONSTRAINTS

Electric Vehicles (EVs) are attractive candidates to reduce transportation's environmental impact. Nonetheless, their low driving ranges, high recharging times, and the poor recharging infrastructure prevent their entire deployment. In this thesis work, we develop a strategy to manage EV fleets for delivery purposes efficiently. The main objective is to find and update the least-cost routes, charging plans, and departure times that allow an EV fleet to visit all destinations, subject to several operational constraints and real-world conditions, a problem known as the Electric Vehicle Routing Problem (E-VRP). The strategy consists of splitting the operation into two stages: pre-operation and online operation. We calculate initial routes in the pre-operation by solving an offline E-VRP (Off-EVRP). In the online stage, the dispatcher updates the routes based on traffic state realizations and EVs' state measurements by solving an online E-VRP (On-E-VRP). We solve both E-VRP variants with Genetic Algorithms (GA) using a novel encoding for each case. The overall strategy is tested with two experiments. Results show that solving the Off-E-VRP provides good initial route candidates, whereas solving the On-E-VRP can improve the operation and service quality. Finally, the developed decision-making system enables the fleet to fulfill the delivery purpose efficiently.

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA, Y AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: JUAN PABLO FUTALEF GALLARDO
FECHA: 2021
PROF. GUÍA: MARCOS ORCHARD CONCHA
PROF. CO-GUÍA: DIEGO MUÑOZ CARPINTERO

A DECISION-MAKING SYSTEM FOR MANAGING ELECTRIC VEHICLE FLEETS SUBJECT TO MULTIPLE OPERATIONAL CONSTRAINTS

Los vehículos eléctricos (EVs) son candidatos atractivos para reducir el impacto ambiental del transporte. Sin embargo, sus reducidos rangos de conducción, altos tiempos de recarga, y la pobre infraestructura de recarga hace difícil adoptarlos. En esta tesis, se desarrolla una estrategia para operar EVs en despacho eficientemente. El objetivo es encontrar y actualizar sus rutas de mínimo costo, planes de recarga, y tiempos de partida para visitar todos los destinos cumpliendo restricciones de operación en condiciones del mundo real. Tal problema se llama Electric Vehicle Routing Problem (E-VRP). La estrategia consiste en dividir la operación en pre-operación y operación online. En pre-operación, se calculan las rutas iniciales resolviendo un E-VRP offline (Off-EVRP). En operación online, se actualizan las rutas según realizaciones del tráfico y mediciones del estado de los EV resolviendo un E-VRP online (On-E-VRP). Ambas variantes del E-VRP son resueltas con algoritmos genéticos (GA) usando una codificación novedosa para cada problema. La estrategia completa es probada con dos experimentos. Los resultados muestran que resolver el Off-E-VRP permite obtener buenas rutas iniciales, mientras que resolver el On-E-VRP mejora la operación y calidad del servicio. Finalmente, el sistema de decisión permite satisfacer los requerimientos eficientemente.

Agradecimientos

El camino que he recorrido hasta aquí ha sido bastante exigente, diverso y entretenido, todo al mismo tiempo. No habría podido llegar hasta aquí sin el apoyo y confianza que tantas personas me han entregado a lo largo de estos años.

Al profesor Diego Muñoz, por ser un guía atento y perseverante, y por siempre estar dispuesto a resolver hasta las dudas más pequeñas.

Al profesor Marcos Orchard, por ser un guía motivador y su increíble apoyo en múltiples ocasiones. Junto a Marcos Orchard, también agradezco a Doris Sáez y Jorge Silva, pues gracias a sus recomendaciones fui aceptado en el programa de magíster.

Por toda mi vida, Ximena, mi madre, y Juan Carlos, mi padre, han sido fundamentales para mi educación. Mi madre me enseñó que, incluso en los momentos más difíciles, siempre se puede encontrar la solución. Mi padre me enseñó que no existen límites para la curiosidad y la motivación.

A mis grandes amigos, el Nacho y el Jano, presentes desde el primer día de clases en la Universidad. Gracias a ustedes tengo recuerdos muy felices e historias (algunas extravagentes) que contar de mi vida universitaria.

A Vale Ríos, Pablo Montero, Rodrigo Pérez, Mati García, Ale Olguin, Joaquín Merino, y Fran Herrera. Las "ñañas", con quien compartí mi día a día en eléctrica y que me entregaron su apoyo en muchas ocasiones.

Al Mati, mi compañero de departamento, por estar y soportarme en los momentos más duros de la tesis. También por los festines veganos improvisados que armamos, que tanto me ayudaron cuando más falta me hacían.

Al Koto y a la Vale Flores, pues han estado por muchos años y en los momentos más difíciles. Gracias a ustedes nunca olvidaré que no solo amo la ingeniería, si no que también la música.

A Silvia, pues gracias a ella descubrí que soy un aventurero y redescubrí lo mucho que me encanta lo que estudié. Tu motivación fue la última que necesité para entrar al magíster y decidir lanzarme al mundo.

A todos ustedes, muchísimas gracias por dejarme ser parte de sus vidas y creer en mí. Este viaje universitario ha sido infinitamente mejor por ustedes.¡Un abrazo!

Contents

List of Tables

List of Figures

Nomenclature

BEV	Battery Electric Vehicle
CS	Charging Station
CVRP	Capacitated Vehicle Routing Problem
D-SPP	Dynamic Shortest Path Problem
DS-SPP	Dynamic and Stochastic Shortest Path Problem
E-VRP	Electric Vehicle Routing Problem
EV	Electric Vehicle
FCEV	Fuel Cell Electric Vehicle
GA	Genetic Algorithm
GHG	Greenhouse Gas
ICEV	Internal Combustion Engine Vehicle
Off-E-VRP	Offline Electric Vehicle Routing Problem
On-E-VRP	Online Electric Vehicle Routing Problem
OR	Operations Research
PHEV	Plug-in Hybrid Electric Vehicle
S-SPP	Stochastic Shortest Path Problem
SOC	State of Charge
SPP	Shortest Path Problem
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

Chapter 1

Introduction

1.1 Context and motivation

In the last decade, Electric Vehicles (EVs) have experienced enormous sales growth. The main reason is that EVs are one of the most promising alternatives to replace Internal Combustion Engine Vehicles (ICEVs), which highly depend on the oil market and significantly contribute to Greenhouse Gases (GHG) emissions [1]. Furthermore, the transportation sector is among the most contaminant sectors: it is responsible for nearly 28% of the total GHG emissions in developed countries [2]. As a result, numerous governments have implemented strategies to penalize excessive pollution due to misuse of conventional means of transport and benefit sustainable projects. In that context, fleet owners are considering more and more the adoption of EV fleets.

Different types of EVs include Plug-in Hybrid Electric Vehicles (PHEVs), Fuel Cell Electric Vehicles (FCEV), and Battery Electric Vehicles (BEVs) [3]. All of them possess multiple benefits to lower GHG emissions but add several drawbacks that cannot be ignored. For instance, let us briefly examine the benefits of BEVs, which are the EV type considered in this thesis work. BEVs propel themselves using an electric motor. As a result, they do not emit GHG locally [3]. The motor is powered by a large battery, which can be recharged using renewable energies. The latter is a desirable feature because a complete free-of-emission power grid could allow fleet owners to lower their GHG emissions substantially. Besides, BEVs for delivery purposes usually come in small sizes and produce an almost noiseless operation [4]. That is of great interest in urban areas, where the space usage and social impact of transportation is significant.

On the other hand, an entire adoption of BEV fleets is still not possible due to their high purchasing costs and the lack of efficient management systems. The battery is the main responsible for high EV prices [4][5]. Although battery prices have lowered considerably in the last decade, they have not reached the costs to make EVs fully competitive. Besides, their capacity is still not enough to make EVs achieve similar driving ranges to ICEVs [2][6]. Therefore, EVs must frequently restore the driving range by recharging their battery. The latter implies that EVs regularly divert to Charging Stations (CS) to make a recharging operation. This operation can take a substantial time, whereas the relationship between this time and the recharging amount is nonlinear [7]. Such different behavior makes most conventional routing techniques very difficult to adapt, if not impossible, in an EV fleet context [3][8].

Vehicle routing is a well-researched subject in the Operations Research (OR) field. This

subject has its foundations on the well-known Vehicle Routing Problem (VRP). This problem aims to find the least cost routes each vehicle in a fleet can follow to serve a finite number of customers, considering several operational constraints [9]. Due to the vast diversity of operational scenarios, a great variety of VRP variants exists. For example, a widely studied variant is the Capacitated VRP (CVRP) [10], where vehicles can carry up to a maximum weight. Another widely studied variant is the VRP with Time Windows (VRPTW) [11], where vehicles can only serve customers at certain times of the day. Of course, in some cases, one may require addressing both variants simultaneously: the Capacitated VRP with Time Windows (CVRPTW) [12].

The VRP plays a crucial role in the last mile of the supply chain. The distribution process is one of the most determinant elements of the whole delivery process: direct contact with customers. Enabling vehicles to provide a high-quality operation offers an immediate improvement in the business marketing and the attraction of new customers [13]. Besides, an efficient handling of the fleet during the operational stage leads to lower operational costs and externalities vastly. Fleet owners should address these last two elements because transportation is the most expensive component in a logistics system [13].

Nonetheless, solving the VRP is a challenging task. The VRP belongs to the family of NP-Hard problems, which implies that it cannot be solved in polynomial time [14]. As a result, exact algorithms, i.e., algorithms that ensure finding the optimum, increment their computational time exponentially as the problem size grows. Such exponential growth of computational time introduces a significant obstacle in practical applications. Fleet owners frequently require to solve the VRP in short periods considering large fleets. Therefore, several exact methods become impractical in real-world scenarios.

To solve large VRP instances in short periods, researchers develop heuristics. A heuristic is a practical method that approximates the optimum in much less time than an exact algorithm [15][16]. In most cases, researchers tackle a particular VRP variant and develop a specialized heuristic to solve it. However, these heuristics could be impossible to apply in other VRP variants that introduce new elements. The latter reason motivates the use of metaheuristics.

A metaheuristic is a special kind of heuristic that intends to solve difficult optimization problems [16]. Metaheuristics are not designed for a particular optimization problem, making them easy to adapt to new VRP variants. Besides, metaheuristics are commonly designed to explore the search space and prevent them from getting stuck in local solutions. The latter makes them suitable for problems with high nonlinear behavior. Among the most famous metaheuristics to solve the VRP we find Evolutionary Algorithms (EA) [17], Tabu Search (TS) [18], Variable Neighborhood Search (VNS) [19], Simulated Annealing (SA) [20], and Ant Colony Optimization (ACO) [21].

The significant research devoted to solve the VRP has provided, without doubt, excellent solution methods for it. Nonetheless, standard VRP formulations only consider ICEVs, which have long driving ranges. Restoring this driving range is easy because refueling is very fast, and refueling stations are available almost everywhere [2]. That is not the case for EVs. As mentioned before, these vehicles have a short driving range, which makes them require restoring their driving range more regularly than ICEVs. The latter is achieved by recharging the battery at a CS, equivalent to refueling an ICEV. However, the battery recharging process

is nonlinear and can take a long time. [2][4].

When addressing EV fleets, the problem is called the Electric Vehicle Routing Problem (E-VRP) [22][23]. This VRP variant addresses typical VRP constraints, but it incorporates the new constraints EVs introduce. The main difference between the classical VRP and the E-VRP is that the E-VRP should allow EVs to detour to CSs to recharge their battery if not enough energy is available. This behavior is similar to a refueling process, with the difference that the battery recharging process heavily affects the operation due to its long duration. As a result, most classical VRP variants and their corresponding solution methods are not suitable in an EV context [23][8].

In most recent E-VRP works, authors agree that considering nonlinear charging functions [7][24], realistic EV energy consumption models [25], and traffic networks' realistic behavior [26] provide better solutions for efficiency and robustness in real-world applications. However, most E-VRP works only address a few limitations. For example, back in 2014, [22] considers time windows and partial recharging, but assuming a linear charging function and constant energy consumption. One year later, [27] extends the previous work considering a realistic energy consumption model but maintaining the linear charging function. Later, in 2017, [7] addresses the nonlinear behavior of the charging function. However, the authors dropped several constraints, such as time windows, EVs capacity, and realistic energy consumption models. Just recently, in 2019, [28] couples the nonlinear charging function with time windows.

Despite the recent E-VRP research achievements, it is still necessary to explore new solutions methods and improve formulations to address more realistic elements. According to the survey conducted in [23], most E-VRP works only tackle a small set of limitations. Furthermore, there is still required to investigate more case studies, dynamic traffic conditions, uncertainty in travel times, integration of nonlinear charging functions, accurate estimation of energy consumption, adaptive routing techniques, and CS capacities.

This thesis work attempts to design, implement, and test with a simulation environment a strategy to manage EV fleets efficiently for delivery purposes. The approach consists of splitting the operation into two stages: pre-operation and online. In the pre-operation stage, the system calculates the initial routes of EVs. Then, in the online stage, the system recalculates the routes according to measurements from the traffic network and EV's states. For each stage, we develop an E-VRP variant and solve it with GA. These E-VRP variants incorporate several real-world elements, realistic EV features, and constrained recharging infrastructure in the same problem.

The first variant is an offline version of the E-VRP (Off-E-VRP), which intends to assign customers and calculate the initial routes of each EV efficiently. We design two GAs to solve this problem: αGA and βGA. αGA determines the customers each EV will serve and calculates the initial routes, whereas βGA only works when customers have already been assigned. Therefore, both GAs can be used together to solve the Off-E-VRP. First, use αGA, and then improve the solution with βGA.

The second variant is an online version of the E-VRP (On-E-VRP). This variant aims to update the routes of EVs during operation according to their state and the traffic network

state. As a result, the On-E-VRP provides a closed-loop method to manage the fleet, where control actions are the new routes of EVs. To solve the problem, we develop onGA, a GA capable of updating the routes based on measurements from the traffic network and EVs' state.

To test the Off-E-VRP case, we generate several artificial instances of the problem and solve them with αGA and βGA. The instances vary in size; thus, they allow us to know how well GAs perform in different complexity scenarios. Using measurements from one of Santiago de Chile's most congested areas, we develop a real-world instance to test both Off-E-VRP and On-E-VRP over realistic scenarios. In our case, we only use the real-world instances to test the On-E-VRP because they provide statistics of travel times and energy consumption between nodes. The latter allows us to simulate a dynamic and stochastic traffic network, one of the key reasons to adopt an online methodology.

1.2 Hypotheses

According to the previous discussion, the following hypotheses will be tested:

1. **Solving a properly formulated Offline Electric Vehicle Routing Problem (Off-E-VRP) enables the fleet to serve all customers efficiently**. A proper formulation of the Off-E-VRP must formally define the operation of EVs, i.e., the sequences of nodes, the charging plan, and the departure time from the depot. We can use those decision variables to calculate an operational cost and check constraints achievement. Due to the high complexity of the problem, we assume the optimal operation is hard to find. Therefore, we will say that properly solving the Off-E-VRP implies finding a solution such that its cost is low compared to other solution candidates, and it satisfies all operational constraints.

2. **An adequately designed Genetic Algorithm can solve the Off-E-VRP**. The GA design includes developing an encoding that stores the operation of all EVs in the fleet, genetic operations, and a proper GA workflow. Genetic operations must be suitable to explore for new solution using the proposed encoding. Finally, the GA will provide the following decision variables: the sequence of nodes, the charging plan, and the departure time of all EVs. These decision variables are such that they properly solve the Off-E-VRP.

3. **It is possible to develop an online method that updates routes in-operation to improve the operational performance and service quality of the fleet operation.** In this hypothesis, we assume that EVs operate in a dynamic and stochastic traffic network. Therefore, the routes obtained after solving the Off-E-VRP may end up having a different performance according to traffic realizations. Developing an online method to update routes consists of properly formulating an Online Electric Vehicle Routing Problem (On-E-VRP) that considers these new traffic realizations. The On-E-VRP will use measurements from the traffic network and EVs to continuously calculate new routes with the same criteria used by the Off-E-VRP. If EVs follow those routes, we will obtain lower costs (performance improvement) and satisfy a higher number of constraints (service quality improvement).

4. **An adequately designed GA can solve the On-E-VRP.** In this case, EVs are operating; thus, fleet size may vary as some EVs finish their operation. Therefore, the GA design must account for a proper encoding that stores the operation of remaining EVs, their genetic operations, and the GA workflow. The decoding mechanism will receive the initial positions and states of EVs, and it will use them to produce the new routes. The result the GA gives are new routes with equal or better performance than the original routes.

1.3 Objectives

1.3.1 Main Objective

The main objective of this thesis work is to design, implement, and test a strategy to manage EV fleets efficiently. This strategy will split the operation into pre-operation and online operation. In the pre-operation stage, the system will generate initial routes for each EV. In the online stage, the system will update the routes and provide a closed-loop control of the fleet.

1.3.2 Specific Objectives

To achieve the main objective, the following specific objectives are considered:

1. To formulate a new offline variant of the E-VRP (Off-E-VRP) that meets the delivery requirements while incorporating decisive real-world elements and EV limitations.

2. To design, develop, and test a solution method to solve the Off-E-VRP.

3. To formulate an online variant of the E-VRP (On-E-VRP) that allows the dispatcher to update the routes according to real-time measurements while accomplishing the same delivery requirements as the Off-E-VRP.

4. To design, develop, and test a solution method that solves the On-E-VRP.

5. To test the performance of the decision-making system in both operational stages.

1.4 Thesis outline

The thesis structure is as follows. Chapter 2 provides an insight into EVs and the VRP. Both lead to a proper definition of the E-VRP. Chapter 3 reviews GAs and their applications to the VRP. Chapter 4 states both the offline and online E-VRP addressed in this work. Several other critical elements are defined, such as the methodology to prevent exceeding CS capacities, the EV energy consumption model, and the EV state-space model used by the optimization model. Chapter 5 introduces the solution framework. That is, the main GA algorithms in charge of solving both offline and online problems. Chapter 6 describes the online system. Chapter 7 conducts a simulation test in two kinds of instances: a real-world instance and artificially-generated instances. Finally, Chapter 8 concludes about the work and prospective research.

Chapter 2

Electric Vehicles and the Vehicle Routing Problem

2.1 Introduction

This chapter provides a review of key elements regarding EVs and the VRP. For EVs, we review several aspects that make them attractive for fleet owners, and the important drawbacks that are still blocking their complete adoption. As batteries are presumably the components that impose these limitations, an entire section is dedicated to review them. We also provide a review of the Shortest Path Problem (SPP) and the classical VRP. These two problems are highly related. Finally, a review of several E-VRP variants and solution methods is provided, which intends to guide the reader in the comprehension of the problem.

2.2 Electric Vehicles

According to [29], it is common to consider three EV categories: Fuel Cell Electric Vehicles (FCEVs), Plug-in Hybrid Electric Vehicles (PHEVs), and Battery Electric Vehicles (BEVs). A brief review of these three categories is presented in the following subsections. More emphasis is put into BEVs, as they are the EV type addressed in this thesis work.

2.2.1 Fuel Cell Electric Vehicles

A FCEV uses a fuel cell to generate electricity with hydrogen. This energy is used to either propel the vehicle, or recharge a battery. These vehicles have a battery which can store energy from regenerative braking or assist the fuel cell when it cannot handle great load variations. In comparison with BEVs and PHEVs, they can be recharged very fast. However, they are less efficient than BEVs and PHEVs, and have high purchasing costs[29].

2.2.2 Plug-in Hybrid Electric Vehicles

A PHEV have both an electric motor and a internal combustion engine motor. They can choose which motor to use to propel themselves and, in comparison to BEV, the driving range a PHEV can reach with the electric battery is much less. However, they accomplish longer driving ranges thanks to the combustion engine motor. PHEVs are considered transition vehicles, as battery technology development is still ongoing[30][29].

2.2.3 Battery Electric Vehicles

A BEV obtains the energy from a large battery. This battery provides energy to power an electric motor, which propels the vehicle. In addition, the battery supplies energy to all external components of the vehicle. BEVs possess plenty of attractive features. For example, BEVs are highly power efficient, they do not produce local Greenhouse Gasses (GHG) emissions, and their battery can be recharged from renewable sources. Nonetheless, BEVs face several challenges which prevent them to become a commercial standard.

Several works agree that the short driving range of BEVs is their largest entry barrier [4][2][29]. Their driving range oscillates among 160 to 240 km, which is almost a quarter the driving range of Internal Combustion Engine Vehicles (ICEVs) [23]. To achieve higher distances, BEVs can recharge their battery during operation in a Charging Station (CS). However, this process introduces several new challenges. A significant issue is the time it takes to recharge the BEV battery. According to [2], recharging a BEV battery could take from 20 min up to several hours. This high time requirements lead to less time to accomplish operational requirements.

Another issue arises from the available recharging infrastructure. Currently, several highly populated areas around the world do not have a proper public recharging infrastructure. Therefore, fleet owners must decide wether or not to invest in new recharging infrastructure.

Batteries are a critical to enable the adoption of EVs. In recent years, there has been a substantial increment on the production of Li-Ion batteries thanks to a reduction in their price, and an increment of their capacity. Nevertheless, Li-Ion-based EVs are not capable of reaching ICEVs driving ranges.

2.3 Batteries: Basic Concepts and Definitions

2.3.1 Techonology Types and Applications

The most common battery technology for passenger and delivery purposes is Lithium Ion (Li-Ion) [4]. Compared to lead acid and nickel metal hydride batteries, Li-Ion batteries posses higher energy density, higher power density, longer lifespan, and low memory effect [29]. Figure 2.1 shows a comparison among several battery technologies. Regarding current battery developments, it is expected that new battery technologies such as Lithium-Sulfure (Li-S) and Lithium-Air (Li-Air) increase the current energy density of Li-Ion batteries up to three times. Nonetheless, there are still major improvements to overcome in order to make these technologies commercially feasible.

2.3.2 State of Charge (SOC)

The State of Charge (SOC) is a percentage representation of the current available energy in the battery [31]. Let us consider that the total EV battery capacity is Q, and that the EV battery can deliver energy up to $E(t)$ at instant t. Thus, the SOC is defined as

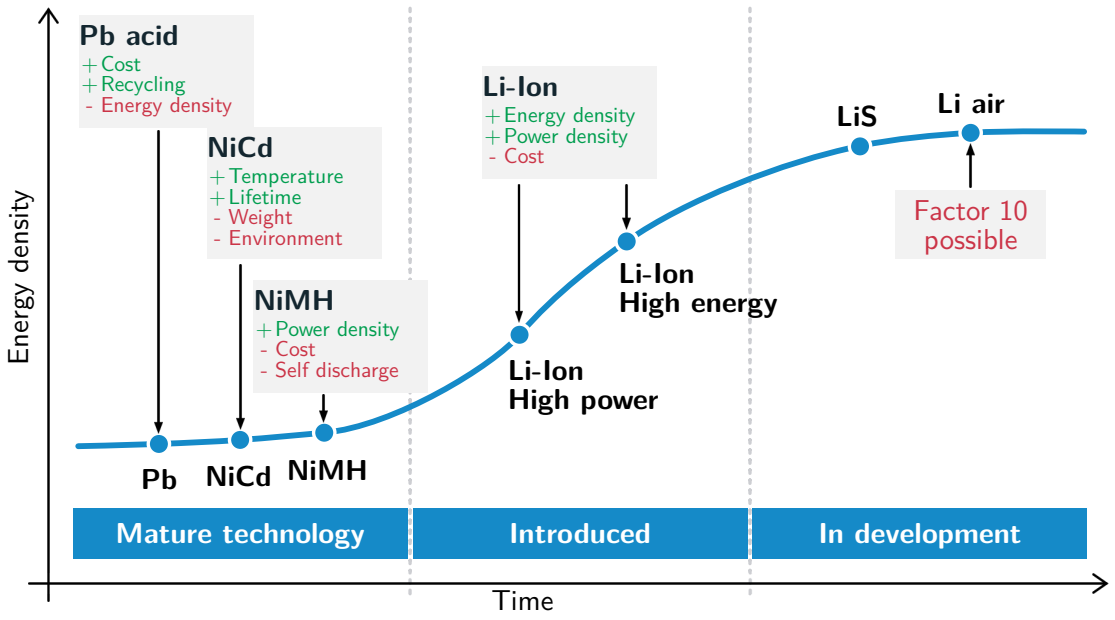


Figure 2.1: Expected developments in battery technology (adapted from [4] ©2013 CE Delft)

$$SOC(t) = \frac{E(t)}{Q} \tag{2.1}$$

Good estimation and forecasting of the SOC is crucial to enable high-quality decision-making in path planning of EVs[29]. Nonetheless, the SOC is not directly observable. Thus, it can be inferred from other observable measurements. However, because many sensing systems are in the presence of noise, and there is a non-linear relationship among the battery parameters, real-time filtering is usually used to estimate the SOC [31].

2.3.3 Battery Degradation: Factors and Operation Recommendations

The battery is the most expensive single component of an EV. Therefore, replacing it should be studied carefully. The replacement of the battery is a direct result of its degradation. The latter represents a reduction of the total energy the battery can store, which occurs by the formation of a Solid Electrolyte Interface (SEI) on the anode's surface and the consumption of lithium ions through the reaction. These effects cause power and capacity fade, respectively [3].

To lower battery degradation, one can apply policies that diminish the impact of the sources that produce degradation. Table 2.1, extracted from [3], presents a summary of cause and effect sources of battery aging. There, it can be noticed that some actions enhance the effect of degradation. In particular, high and low SOC levels are factors that influence battery degradation in many cases.

Several studies have shown that operating batteries in levels near empty or near full leads to quicker battery aging. In [32], the authors study degradation considering different SOC policies in Multi-Service Portfolios. A SOC policy consists of constraining the SOC to operate

Table 2.1: Overview of Li-Ion battery anode aging mechanisms. Reprinted from Journal of Power Sources, Vol. 147, Vetter et al., *Ageing mechanisms in lithium-ion batteries*, 269-281, ©(2005), with permission from Elsevier.

Cause	Effect	Leads to	Reduced by	Enhanced by
Electrolyte decomposition (→ SEI) (Continuous side reaction at low rate)	Loss of lithium Impedance rise	Capacity fade Power fade	Stable SEI (additives) Rate decreases with time	High temperatures High SOC
Solvent co-intercalation, gas evolution and subsequent cracking formation in particles	Loss of active material (graphite exfoliation) Loss of lithium	Capacity fade	Stable SEI (additives) Carbon pre-treatment	Overcharge
Decrease of accessible surface area due to continuous SEI growth	Impedance rise	Power fade	Stable SEI (additives)	High temperatures High SOC
Changes in porosity due to volume changes, SEI formation and growth	Impedance rise Overpotentials	Power fade	External pressure Stable SEI (additives)	High cycling rate High SOC
Contact loss of active material particles due to volume changes during cycling	Loss of active material	Capacity fade	External pressure	High cycling rate High DOD
Decomposition of binder	Loss of lithium Loss of mechanical stability	Capacity fade	Proper binder choice	High SOC High temperatures
Current collector corrosion	Overpotentials Impedance rise Inhomogeneous distribution of current and potential	Power fade Enhances other aging mechanisms	Current collector pre-treatment	Overdischarge Low SOC
Metallic lithium plating and subsequent electrolyte decomposition by metallic lithium	Loss of lithium (loss of electrolyte)	Capacity fade (power fade)	Narrow potential window	Low temperature High cycling rate Poor cell balance Geometric misfits

between ranges narrower than zero to one, e.g., from 0.2 to 0.8. Their results, shown in Figure 2.2, show that some policies last longer than others. Besides, the policy 0 to 1 is not the best policy because the battery degrades faster than other policies and, according to the economic study they conduct, long-term costs are higher than other policies.

[30] provides several recommendations to assess battery management in the context of EVs. Among these recommendations, they define the ideal SOC working range in Figure 2.3, where they conclude that 20% to 90% is an ideal policy. A similar result is obtained by [3], where they use a battery degradation model to study the impact of several variables.

2.4 The Shortest Path Problem

Navigation systems attempt at determining a path to travel from one location to another location[33]. In real life, there are numerous alternatives to do that. Yet, it is quite straightforward that some of these alternatives are better than others. The cost of a path is a measure of the effort one must put to travel from the start to the destination. For example, the path cost might refer to the travel time, energy consumption, distance, or any other cost indicator between start and destination.

The problem of finding a path that minimizes its cost is called the Shortest Path Problem (SPP). There are several SPP variants; however, this section focuses on the SPP from a specified vertex to another vertex. To provide a more rigorous understanding of this problem,

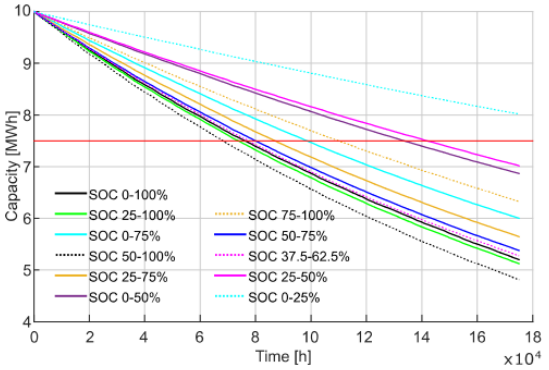


Figure 2.2: Energy capacity degradation for different operational policies. Red line indicated 75% of nominal capacity (extracted from [32] ©2016 IEEE)

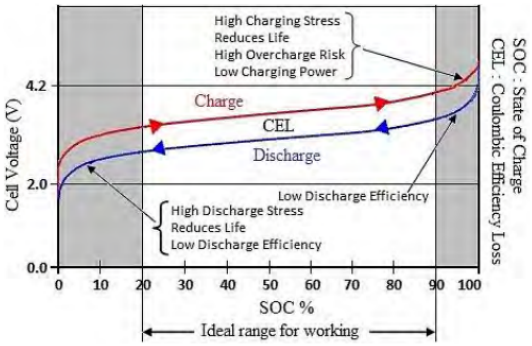


Figure 2.3: Ideal SOC range (adapted from [30] ©2018 IEEE)

we present a classical formulation extracted from [34] (Chap. 11). Consider the weighted directed graph (digraph) $D = (V, A)$, where V is the vertex set, and A is the arc set. In a road network, vertices denote road intersections, whereas arcs denote the links among those intersections.

The digraph is described by an n by n cost matrix $C = [c_{ij}]$, where each c_{ij} is the cost of traveling from vertex i to vertex j. If $i = j$, then $c_{ij} = 0$. And, if there is no link between i and j, then $c_{ij} = +\infty$. A path P is a sequence of adjacent arcs; thus, the total cost of P, C_P, is the sum of all arc costs in P. Consider the start vertex s and the terminal vertex t. The problem is determining a path P^* such that its cost C_{P^*} is minimum.

In most cases, the arc costs do not satisfy the triangle inequality. That is, $c_{ik} + c_{kj}$ is not necessarily greater than c_{ij} for any $i, j, k \in V$. This assumption implies that the optimal path among two adjacent vertices may not be the direct link between them. In real road networks, this is equivalent to deviate from the main road to reach the destination.

As the solution of the SPP is nontrivial, a smart solution method is required. Among all SPP solution algorithms, perhaps the most famous one is Dijkstra's algorithm. The latter gained popularity due to its high efficiency in solving the SPP with nonnegative arc costs.

Although the classical SPP applies to several practical problems, it neglects various real-world elements. For example, if the cost c_{ij} is the travel time between two geographical points i and j, it is expected that, at certain times of the day, c_{ij} increases due to traffic at rush hours. Hence, the shortest path may differ at instants t_0 and t_1 if $t_0 \neq t_1$. When c_{ij} depends on time, the problem is known as Time-Dependent SPP (TDSPP). Another variant is the Stochastic SPP (SSPP), where the arc costs are uncertain, i.e., they have a probability density function (pdf). Both TDSPP and SSPP can be merged to create the Stochastic Time-Dependent SPP (STDSPP).

2.5 The Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is one of the most significant problems in the Operations Research (OR) field. Essentially, the VRP aims at finding an optimal route scheduling for a vehicle fleet to serve a finite set of geographically-scattered customers, subject to several operational constraints [35][36][9]. Although the VRP is a widely-known problem, neither a unique formulation nor a unique solution method exists. In reality, VRP formulations and their solution methods depend on the nature of the problem to address.

Several surveys and studies [9][37][23] cite the 1959 article by Dantzig and Ramser, *The Truck Dispatching Problem* [38], as the first VRP article. There, a fleet of trucks must deliver gasoline from a central depot to several gas stations. Dantzig and Ramser modeled the problem as an optimization problem and developed a method to solve it. In the following years, several studies devoted to solving the VRP were conducted. These new solutions methods provided better solutions and were more efficient than the original method developed by Dantzig and Ramser.

In addition to enhanced solution methods, new variants of the problem were developed. These variants arise as new cases where operational constraints differ from the original VRP, and new cases with more realistic elements. Besides, improvements in computational capacity and communications allowed researchers to explore more ambitious VRP formulations and their corresponding solution methods. Currently, more than 60 years have passed since the first VRP article, and more than a thousand studies where the VRP is the main topic have been published[37][35].

To provide a better understanding of the problem, we present a classical VRP formulation. This formulation is extracted from [9], which provides an extensive review of several classical VRP formulations. Consider the complete graph $G = (V,\ A)$, where $V = \{0, \ldots, n\}$ is the vertex set and $A = \{(i, j) : \forall i, j \in V\}$ is the arc set. The vertices, also called *nodes*, are geographical points such that vertices $i = 1, \ldots, n$ represent customers, and vertex 0 represents the depot. Each arc $(i, j) \in A$ is associated a nonnegative cost c_{ij}. This cost represents the cost a vehicle spends when traveling from i to j. If G is a directed graph, then $c_{ij} \neq c_{ji}$, i.e, the cost matrix is asymmetric, and the problem is called asymmetric VRP. Otherwise, the problem is called symmetric VRP. In most practical cases, the cost matrix satisfies the triangle inequality,

$$c_{ik} + c_{kj} \geqslant c_{ij}\ \forall i, j, k \in V.$$

The latter requirement states that the arc (i, j) is always the most convenient way to travel from i to j. Notice that, if the path from i to j is equal to the shortest path, the triangle inequality holds for (i, j). The same occurs when the cost is proportional to the Euclidian distance between i and j. The latter also implies that the problem is symmetric.

Each customer i for each $i = 1, \ldots, n$ requires a known nonnegative demand d_i, whereas the depot has an imaginary demand $d_0 = 0$. In the depot, there is a set of m identical vehicles. These vehicles deliver all requirements the customers demand. In most realistic scenarios, vehicles have a maximum capacity C such that $C \geqslant d_i$ for each $i = 1, \ldots, n$. When the latter constraint is considered, the VRP is also called Capacitated VRP (CVRP).

Considering all the above, the VRP is defined as the problem of finding a collection of exactly m routes (one for each vehicle) of minimum cost. The total route cost is the sum of all arc costs in that route. Besides, the routes are such that each vehicle starts and ends at the depot; each customer is visited exactly once by a vehicle; and the sum of all demands each vehicle transports does not exceed capacity C.

The above VRP formulation can be extended by adding more operational constraints, which produce a new VRP variant. For example, a popular VRP variant is the VRP with Time Windows (VRPTW). In this variant, customers have a service time (in addition to its demand), and a time interval where the operation can take place. Sometimes, the vehicle arrives at a customer before its time window begins. In that case, the vehicle must wait until the operation can take place.

In addition to more operational constraints, the VRP can include several real-world elements. For example, instead of a fixed arc cost, consider a time-dependent arc cost. This time dependency allows the VRP to address evolving traffic networks where peaks hours occur at certain times of the day. Other real-world elements can be addressed, such as road network stochasticity, dynamic and stochastic customer demands, online route updates, among many others. Including several real-world elements and online route updates may lead to a reduction of operational costs and a vast improvement of service quality.

The VRP is simple to formulate, yet very difficult to solve. The simplest VRP version, the CVRP, is a generalization of the well-known Travelling Salesman Problem (TSP). The latter problem is NP-Hard. As a result, the CVRP and all variants that generalize it are NP-Hard [14]. This property implies that the problem cannot be solved in polynomial time. In fact, as the problem size increases, the required time to solve it grows exponentially. Therefore, exact methods, i.e., methods that find the global optimum, can take an enormous amount of time to solve large problem instances. Besides, when the VRP incorporates real-world elements, the complexity grows even more.

Fleet owners may require to solve large VRP instances in short periods. This requirement makes several exact methods unpractical. As a result, several heuristics to solve the VRP have been introduced. A heuristic is an algorithm that approximates the optimum faster than exact algorithms[39]. Therefore, they are usually employed to solve large VRP instances or very complex real-world problems. However, the development of a heuristic usually aims at solving a very particular problem. Consequently, the heuristic may not work with a new problem variant.

We refer to [9] as an extensive review of exact solution methods for classical VRP formulations. For a review of metaheuristics, we refer to [39]. In this thesis work, the solution scheme employs a GA to solve the Electric VRP. Chapter 3 provides a review of GA to solve the VRP, whereas following Section 2.6 introduces the so-called Electric VRP (E-VRP).

2.6 The Electric VRP (E-VRP): Variants and Solution Approaches

2.6.1 What makes the E-VRP so different from the VRP?

As reviewed in the previous section, in the last 60 years, several VRP variants have been addressed, and plenty of high-quality solution methods have been developed. Nonetheless, the majority of these works consider ICEVs only. The latter have large driving ranges, and a refueling them is quite fast and easy, as there are sufficient gas stations available almost everywhere. On the other hand, the driving range of EVs is nearly a quarter of the driving range of ICEVs.

To achieve longer driving ranges, EVs can recharge the battery in operation. Unfortunately, the recharging infrastructure is not as spread as the refueling infrastructure, and the recharging time of an EV battery could be substantial. As a result, a considerable portion of the operation may be spent by recharging operations, whereas CS availability is not ensured [8][23].

Because of the different operational behavior of EVs, most classical VRP formulations and their respective solution methods are hard –if not impossible– to adapt in the context of EV fleets. Consequently, the Electric Vehicle Routing Problem (E-VRP) emerges as a VRP variant where all vehicles are electric. That is, the E-VRP must address all classical VRP operational constraints plus all constraints introduced by EVs.

2.6.2 Variants and State of the Art

This section provides a brief review of some remarkable E-VRP variants and recent developments that concern this thesis work. For a detailed review of E-VRP variants and solution approaches, we refer to the survey conducted in [23].

According to [8], the first E-VRP work was published in 2011 by [40]. In that study, the authors introduce the Recharging Vehicle Routing Problem (RVRP), which considers the operation of EVs as an extension of the distance-constrained VRP with time windows. EVs can recharge their battery at customer locations using fast recharging and assuming a constant recharging amount. Their results show that time windows are a critical constraint when EVs have a maximum driving range and high recharging times.

As of 2012, the study in [41] makes another remarkable contribution. It introduces the Green VRP (GVRP), a VRP variant that tackle Alternative Fuel Vehicle (AFV) fleets. This variant is the first to consider vehicle detours to refuel them. However, it does not address neither load capacities nor time windows. To solve the problem, the authors develop two heuristics to exclusively solve the G-VRP: the Modified Clarke and Wright Savings (MCWS) algorithm and the Density-Based Clustering Algorithm (DBCA). Their results show solutions highly depend on travel distance and the refueling behavior. This study is regarded as a milestone for the future development of more sophisticated VRP variants that tackle AFV fleets, where one of the most prominent trends has been the development of the E-VRP.

The study conducted in [22] in 2014 is the first one to introduce the E-VRPTW with recharging times that depend on the SOC level entering a CS. In that case, EVs recharge an enough amount to return to the depot. Recharging times are calculated using a linear approximation. The authors develop a mixed-integer program, which is solved using Variable Neighborhood Search (VNS) and Tabu Search (TS). Their solution method is capable of solving up to 100-customer instances.

Mixed fleets of EVs and ICEVs have also been addressed. The case in [42] considers that scenario in a Capacitated E-VRP with Time Windows (E-CVRPTW) variant. They also tackle different charging technologies to prevent incompatibility between EVs and CS, and constrain the SOC to operate within a safe zone. The latter policy aims at prolonging the total lifespan of the EV battery. As other studies, a linear approximation estimates recharging times. The authors develop a Charging Routing Heuristic (CRH) and a new Inject-Eject Routine-Based Local Search (IELS) to solve the problem. The CRH generates initial solutions, and IELS improves them. Their results are applied to instances with up to 500 nodes successfully.

In [27], they extend the work in [22] by assuming a mixed fleet and a realistic EV consumption model. The latter allows the problem to estimate energy consumption accurately. Recharging times are estimated with a linear function. The authors use an Adaptive Large Neighborhood Search (ALNS) algorithm to solve the problem. Their results show that the solution methods work with instances of up to 75 customers. They conclude that real-world modeling of energy consumption and recharging times are crucial for future E-VRP development. Besides, the vehicle load is relevant as it diminishes when

The work in [7] introduces a new level of realism to the E-VRP. The authors consider a nonlinear charging function that allows the problem to estimate recharging times accurately. This new problem variant is known as the E-VRP with Nonlinear charging function (E-VRPNL). The authors consider three recharging technologies and the possibility of doing partial recharging. On this occasion, the cost function also minimizes the recharging time, allowing EVs to spend the least amount of time in CSs. To solve the problem, they split the solution into sequencing (order of customers) and charging (inserting charging operations); the latter has its own name: Fixed-Route Vehicle Charging Problem (FRVCP). Their results show that up to 12% of vehicles recharge in the nonlinear zone, and most routes have multiple recharging operations.

In [43], the authors tackle a practical E-VRP problem. Their variant considers the E-CVRPTW, time-dependent travel times, and a real traffic network of Beijing, China. They provide a Dynamic Dijkstra algorithm to calculate the shortest path, considering the time dependency of travel times. They develop a 50-customer instance using the real data and solve it with a Genetic Algorithm (GA). Their solution approach finds feasible solutions in about three hours. The study is extended in [43], where the problem adds a realistic energy consumption model. The problem is also solved with a GA. They show that the energy consumption is highly dependent on traffic profiles.

In [44], the authors develop a routing system for an airport shuttle. Their system addresses a charge scheduling at the depot, considering assigned routes. The same SOC constrain policy as [42] is applied in this work. They assume that a set of charging operations can take place

in a programming horizon. They also assume a closed-loop dispatcher that is capable of deviating vehicles to public CS in case they do not have enough energy to come back to the depot. To solve the problem, the authors use a Differential Evolution (DE) scheme. Their implementation is applied to a 6-vehicle fleet successfully.

In all of the above-reviewed works, E-VRP formulations address multiple CS visits by copying them. That is, if a set F represents a set with s CS, then another set F' contains $\beta \cdot s$ dummy CS, where β is the number of CS replicates. Most works do not explain how to set the value of β. According to [7], choosing β is a nontrivial task, and its value affects the complexity of the problem substantially. To address this latter issue, [7] develops a simple iterative heuristic. It consists of starting with $\beta = 0$, then increase β by one, and repeat until a time limit is reached (they set a maximum time of 100 hours) or the cost function $f(\cdot)$ is such that $f(s_\beta) = f(s_{\beta-1})$, where s_β are solution candidates assuming β.

The heuristic developed by [7] may be useful in several applications. However, as they did not develop an exact algorithm, their solutions may differ sometimes; therefore, their optimal β may vary as well. In most recent works, [28] and [24] aim at improving the formulation of the E-VRPNL introduced by [7] by addressing the CS copies issue. [25] tackles the problem with CS Paths (CSP). A CSP is a possible detour an EV may follow to visit a CS. That is, given two non-CS nodes i and j, the corresponding charging path starts at i and finishes at j, and it contains a sequence of CS. The latter sequence could be empty. Using CSPs, [24] develops an arc-based formulation and adapts the FRVCP to that formulation. CSPs are calculated before solving the actual problem. To solve the FRCVP (subproblem of the E-VRPNL), the authors develop exact and heuristic algorithms. Their results show that the CS copies approach may lead to suboptimal solutions because good candidates are eliminated, or infeasible solutions are found. They also provide new BKS to the instances created by [7] in less computational time. In [28], the authors develop a very similar concept, but considering only one CS in the CSP and time windows. They adapt the instances created by Solomon by introducing a proper charging function. Their results show that their solution is suitable. Besides, their solution method is capable of finding current BKS.

Few studies have addressed the CS capacity problem. In particular, [25] studies the impact of limiting the capacity of some instances provided by [7]. They develop three new E-VRPNL models that are capable of tracking CS capacities. Two of them use dummy copies of CS, while the third one uses a CSP approach. They develop a two-stage heuristic to solve the problem. Their results show that the problem is incredibly challenging, even for small-size instances. Besides, they found that limiting CS capacities to 1 turns nearly 50% of the feasible solutions by [25] into infeasible solutions. That percentage drops to 11% and 2% assuming capacities of 2 and 3, respectively.

The survey conducted in [23] reports 78 E-VRP works as of 2019. Regarding their solution approaches, the authors found that 17 studies develop exact methods. The rest develop heuristics, where the three most common are ANLS, GA, and LNS. In most realistic cases, GA tends to be more used. Regarding this metaheuristic, the authors found that most implementations are not efficient. However, some GAs have proven to be competitive with other heuristics by finding BKS for the E-VRPTW developed by [22].

2.7 Discussion

In this section, the most significant benefits and drawbacks of EVs have been reviewed. The review shows that EVs are a highly promising alternative to ICEVs to enable green transportation in the public and private sectors. Most drawbacks come from battery-related issues. Until now, most EVs are propelled with LiIon batteries, which provide driving ranges of nearly a quarter the driving range of ICEVs, and their recharging time is substantial. As the recharging infrastructure is not well developed yet, the probability that a CS is available to provide recharging is low. Besides, the nonlinear behavior of the recharging process and the energy consumption makes it critical to develop proper estimation mechanisms.

The E-VRP is introduced as a VRP extension that addresses EV limitations. Since the latter pose several capacity constraints and nonlinearities, most classical VRP formulations and their solution methods are almost impossible to adapt in an E-VRP context. Therefore, a new research trend emerges dedicated to formulating and solving new E-VRP variants.

Most problems formulate the E-VRP as a MILP and solve it with a two-step heuristic. Regarding population-based solutions, GA is the most common approach. However, most of the GA implementations are poorly realized. Still, some works have successfully implemented efficient GAs and provide excellent solutions to some E-VRP variants. The latter suggests that GAs are suitable to solve the E-VRP, but they should be designed carefully.

To conclude, the review shows that most works tend to address just a few limitations of EVs. To the best of our knowledge, no study addresses the following constraints in a single formulation:

- Time-dependent travel times
- Limited CS capacities
- Nonlinear charging function
- Realistic EV energy consumption model
- Time windows
- Load Capacity
- Constrained SOC policy to take care of battery health
- Routes update based on real-time measurements of traffic and vehicle states.

Several studies agree that the above constraints are crucial to providing excellent decision-making for EV fleets.

Chapter 3

Genetic Algorithms: A Brief Review and its Applications in VRPs

3.1 Introduction

Several practical problems require lowering costs or improving the performance of systems. This performance can be modeled as a cost function representing how well the system operates, assuming a certain operational point. The optimization problem aims at finding the optimal point that maximizes or minimizes the cost function.

In the past decades, metaheuristics have become a popular standard to solve difficult optimization problems. This popularity is due to metaheuristics abilities to provide estimations of the global optimum very fast. This goal is achieved using several random guided techniques. This method accounts for finding the global optimum where multiple local optima are present, and preventing high execution times due to combinatorial explosion.

Genetic Algorithms (GAs) belong to the family of Evolutionary Algorithms (EA). The latter is based on the biological concept of evolution, where several individuals in a population strive to survive throughout generations. Individuals undergo three well-known operations: selection, crossover, and mutation. These operations allow EAs to improve the quality of the population, thus finding better results.

Several works utilize GA and EA indistinctly. However, in this work, EAs involve several other techniques, such as Differential Evolution (DE), Genetic Programming (GP), among others. As the main solution approach for the E-VRP is a GA, the only emphasis is put into this kind of algorithm.

In this chapter, we introduce the concept of hard optimization and the main reasons to use metaheuristics. Then, a summary of basic GA concepts is provided. Finally, a review of some typical GA implementations to solve the VRP is made. The review includes some of the most typical representations and their genetic operators.

3.2 Hard Optimization

According to the variables they manipulate, optimization problems can be classified into three categories: discrete, continuous, and mixed. Discrete optimization tackles problems where variables are discrete. A famous discrete optimization problem is the Traveling Salesperson

Problem (TSP). In this problem, the problem intends to find the shortest route a salesperson must follow to visit a finite collection of cities before returning to the departure city. On the other hand, continuous optimization deals with continuous variables. Finally, a mixed optimization problem considers both discrete and continuous variables. A quite trending mixed optimization problem is Hyperparameter Optimization (HPO). This problem intends to optimize the hyperparameters of a Machine Learning (ML) system. Solving this problem reduces human effort while improving the performance of ML algorithms.

Hard optimization involves dealing with optimization problems that are very difficult to solve. According to [16], two hard optimization problem families can be distinguished: some discrete problems where there is not a knowledge of exact algorithms to solve them in polynomial time. Some continuous problems where there are no known algorithms that can find the global optimum with total confidence. The development of exact algorithms, i.e., algorithms that find the global optimum, is subject to its practical implementation. In hard optimization, the required time to solve the problem increases dramatically when the problem size grows. As a result, several exact algorithms are not suitable in real-life scenarios.

Metaheuristics arise as methods to approximate the global optimum of hard optimization problems within less time than exact algorithms. These algorithms are usually based on real-world concepts, including biology, physics, and social interactions. Therefore, their implementation and workflow are not unique, allowing metaheuristic developers to create innovative approaches. Some remarkable metaheuristics are: Simulated Annealing (SA), Evolutionary Algorithms (EA), Tabu Search (TS), and Ant Colony Optimization (ACO).

3.3 Basic Concepts

3.3.1 Terminology

A GA is a population-based algorithm. That is, the algorithm does not provide a single solution candidate, but a collection of solution candidates. In the GA context, a population is a collection of several individuals. An individual encodes a candidate solution to the optimization problem. That is, the individual is an indirect representation of the solution, not the solution itself. Also, this representation is such that each individual encodes a solution in the search space.

Undoubtedly, some individuals contain better solutions than others. When an individual provides a better solution, its fitness is higher. In a GA approach, the fitness function associates a fitness value to each individual. That fitness value indicates how likely it is that the individual is selected to reproduce. Following the assumption of survival of the fittest, the fitness value also represents how good is the solution the individual provides. Sometimes, the fitness function is the cost function itself. However, it depends on the nature of the problem. In reality, it is necessary to construct a fitness function that best fits the problem.

A generation represents a single iteration of the algorithm. At each generation, individuals evolve due to three genetic operations: selection, crossover, and mutation. When a generation starts, the GA selects some individuals according to their fitness. These individuals will be used for reproduction; as a result, they are called parents. The mutation operator alters

the content of a single parent. On the other hand, the crossover operation combines two or more parents in a mating process. Thus, the resulting individuals are the children, which constitute the so-called offspring. In the next generation, the children become the parents of a new offspring, completing the generational loop. The GA finishes when a termination criterion is reached, for example, a maximum number of generations.

In the big picture, the GA uses genetic operations to select the best individuals (thus, the best solutions to the problem), whereas genetic operators provide diversity. The latter allows the GA to broadly explore the search space and prevent it from getting stuck in local minima.

3.3.2 Genetic Operators

Selection

Following the Darwinian concept of evolution, the fittest individuals will survive to generate a new offspring. This process is repeated so that, at each generation, the average quality of all individuals is better. This quality increase occurs because children are born from the best parents. In a GA approach, this process is called selection.

The selection procedure determines which individuals will reproduce to generate the offspring. The probability an individual is selected depends on its fitness, and, in general, individuals with higher fitness are more likely to be chosen. Thus, the selection procedure aims to maintain fitter individuals to improve the average fitness of the population at each generation.

Crossover

The crossover operation uses two ore more parents to generate one or more children. These children may or may not inherit information from their parents. However, in most cases, they do. The operator is stochastic, which implies that the children it produces may differ if the operation is repeated using the same parents.

In most GA applications, just a portion of the parents mate via crossover operation. This portion is determined by the crossover rate, which ranges from zero to one. The crossover operation uses the crossover rate to pick the parents that will reproduce.

Mutation

The mutation process alters the structure of several individuals at random. By doing that, the operator allows the GA to explore a zone around the individual it mutates. That is, mutation operation provides a random local search. In comparison to crossover operation, mutation operation can provide considerable solution improvement because new zones are explored. That is crucial to find the global optimum when the problem has several local minima. On the other hand, crossover operation may combine similar individuals that are not well-fitted or recombine identical individuals. These issues lead to local optima.

The mutation rate states the portion of children that will undergo a mutation process. This rate is highly problem-dependent. In some cases, the mutation rate is set very low to

prevent deviating from good solutions. High mutation rates can be useful to lower selection pressure because the fittest individuals are more likely to be altered. The special case where the mutation rate is one implies that all individuals will experience mutation. Therefore, the algorithm turns into a random search.

3.3.3 Generic GA

Algorithm 1 shows a generic GA structure.

Algorithm 1: Generic GA

```
k ← 0;
P(k) ← Initial population;
evaluate(P(k));
while notFinished() do
    P_0 ← select(P(k));
    P_1 ← crossover(P_0);
    mutate(P_1);
    evaluate(P_1);
    P(k + 1) ← newPopulation(P_1, P(k));
    k ← k + 1;
end
return bestIndividual(P(k))
```

3.4 Common Representations and Their Genetic Operations

In nature, an organism stores its genetic information in genes. The composition of several genes form a chromosome, which, in turn, gives rise to the genotype when it is combined with other chromosomes. In the GA scheme, an individual mimics a chromosome storing a collection of genes. The process of representing a solution candidate as an individual is called encoding. The inverse process, i.e., turning an individual into a solution candidate, is called decoding. We denote $\Psi(I)$ as the decoding operation over individual I.

The fitness function usually depends on the cost function of the optimization problem. As there are several ways to encode a solution, a proper decoding mechanism must also be provided. After decoding the individual, the GA can evaluate the fitness function according to its definition. Because fitness evaluation depends on more than one process, it is considered a computationally expensive procedure. Thus, a proper GA implementation must define an efficient decoding mechanism and an efficient evaluation procedure.

There are several encoding representations, which mainly depend on the nature of the problem to solve. In the following sections, a review of commonly used encoding mechanisms is provided.

3.4.1 Binary Encoding

In a binary encoding, the individual stores a solution in a string that contains only ones and zeros. This string has length L and does not change throughout the evolution. For example, the strings

$$I_1 = [010110] \quad I_2 = [110011]$$

can be two individual candidates with $L = 6$.

The decoding process turns these binary strings into values in the search space. For instance, a simple decoding method is to assume the string is an unsigned base-2 numeral. That is, the string can be turned into a positive integer number, according to

$$\Psi(I) = \sum_{j=0}^{n-1} I[j] \cdot 2^{n-1-j},$$

where $I[j]$ is the j-th bit value in the string, and n is the length of the string. Using this method to decode I_1 and I_2, we obtain:

$$\Psi(I_1) = 22, \qquad \Psi(I_2) = 51.$$

Notice that, according to this method, the individual can store up to 2^L different solutions.

If the problem is two-dimensional, L can be doubled to produce a two-gene individual as follows:

$$I = [\underbrace{010110}_{x_1} \vdots \underbrace{110011}_{x_2}]$$
$$\Psi(I) = (22, 51).$$

This representation can also store signed integer numbers, decimal numbers (as binary floating-points), or binary variables. Furthermore, one can create a custom decoding mechanism according to the problem requirements.

Crossover

Most classical crossover operations rely on merging parts of individuals. These operations are known as n-point crossover and consist of choosing n random points of the individual and swapping the content among these points. The most simple version, the 1-point crossover, splits each individual into two pieces. Then, the operator exchange those pieces. A 2-point crossover works similarly, but one must decide which parts to exchange.

Another typical crossover operator is the uniform crossover. Here, a random template with the size of individuals is generated. Each of the template components is a one or a zero. The operation consists of swapping values among individuals in the positions where the template has a one.

Mutation

When using a binary encoding, the mutation operator turns ones into zeros, and vice-versa. This process is known as bit-flip. The operator does a bit-flip after choosing several bits

randomly. In most cases, the probability that a bit is selected is very low, intending to prevent the destruction of good candidates.

3.4.2 Real Encoding

The real encoding approach considers that each individual is a vector containing real numbers. That is, each individual I_j is such that $I_j \in \Omega \subset R^n$, where Ω is a bounded search space. This search space may be equal to the optimization search space or not. Therefore, many methods exist to decode this representation kind. FOr example, the following candidate represent a real encoding:

$$I = [1.5, -8.1, 101.2]$$

We refer to [45] for a detailed review of genetic operations for real encoding approaches. In the following, a brief review of common operations is provided.

Crossover

In this case, the n-point operation is valid as well. That is, the individuals can be divided into several pieces to produce a recombination of them.

Another type of crossover operations derive from arithmetic operations. A simple, yet commonly used method is the Arithmetic Crossover (AMXO) operator, which combines two parents $x^{(1)} = [x_1^{(1)}, x_2^{(1)}, \ldots]$ and $x^{(2)} = [x_1^{(2)}, x_2^{(2)}, \ldots]$ to create two children $y^{(1)}$ and $y^{(2)}$ according to

$$y_i^{(1)} = \alpha_i x_i^{(1)} + (1 - \alpha_i) x_i^{(2)},$$
$$y_i^{(2)} = \alpha_i x_i^{(2)} + (1 - \alpha_i) x_i^{(1)},$$

where $y_i^{(k)}$ is the i-th component of child k, and α_i is a uniform random number.

Mutation

For real encoding, most mutation operators use a random function that alters some of the values in the individual. For example, a uniform mutation selects a random position i in the individual. The operator then converts the value in that position as follows:

$$I[i] = \alpha_i,$$

where α_i is a random number sampled from a uniform distribution.

Another typical example is the Gaussian mutation. In this case, the operator generates a random number β_i from a normal distribution. The operator then adds the number to the corresponding element in the individual:

$$I[i] = I[i] + \beta_i.$$

3.4.3 Integer Encoding

In this case, an individual is a collection of integer numbers. This encoding is useful in combinatorial problems where the order of certain variables is relevant. As an example, consider the Traveling Salesman Problem (stated in section 3.2). Here, the individual

$$I = [5, 2, 3, 4, 1]$$

can represent the route the salesperson must follow, where each integer is a city label.

Crossover

In most cases, this encoding stores information regarding the order of numbers. Therefore, a standard n-point crossover may be a destructive approach because some numbers may repeat or disappear. Still, the operation behavior depends on the nature of the problem.

Mutation

Mutation aims at reordering the integers in an individual. In this case, two operations can be distinguished. The first one is the 2-opt mutation. It consists of randomly selecting two positions in the individual and reversing the integers delimited by these positions. Another well-used yet straightforward operation is the swap mutatio, which merely swaps two integers in the individual.

3.5 Applications of GA to the VRP

GAs have been successfully applied to solve several VRP variants. They do it by adequately encoding routes and defining suitable genetic operations. Most works conclude that significant effort should be put in designing proper genetic operations. The latter are responsible for either allowing the GA to explore new routes and improving existing ones. In the following, a review of some remarkable representations is presented.

There are several techniques to encode candidate routes in a chromosome. In most cases, chromosomes encode routes with an integer representation, where integers can denote either customers or vehicles. When integers describe customers, one can choose among an explicit or an implicit representation. An explicit representation stores routes as is, i.e., each integer is a customer, and the order of integers states the customer sequences. The chromosome does not include the depot. Therefore, the decoding operation must insert the depot at the start and the end of the routes. To distinguish among routes, the decoding operator requires a delimiter that divides the routes. In most cases, this delimiter is a special character or a particular number that is not assigned to any customer.

An example of explicit representation is given in [46]. The authors separate routes with zeros. For example the individual

$$I = [0, 1, 4, 2, 0, 5, 3]$$

represents two routes: $\{1, 4, 2\}$ and $\{5, 3\}$. Notice that this encoding states that the number of vehicles is fixed. As a result, the recombination of individuals may lead to the destruction

of routes because customers may disappear or repeat. The latter usually occurs with simple n-point crossover operations. To address this issue, the authors develop the Best Cost-Best Route Crossover (BCBRC). This operator swaps the best routes among two parents, reallocating customers into different routes to prevent destruction.

In an implicit representation, chromosomes do not use a delimiter to differentiate among routes. Hence, they require an additional decoding mechanism. Although several decoding methods for an implicit representation exist, no agreement or theory benefits a particular one.

A relatively conventional implicit encoding is to store routes in an integer representation of length N, where each integer represents one of the N customers in the network. As they do not use any delimiter, they use a method to construct routes from individuals. For example, in [47] (CVRP) and [48] (VRPTW), they use a Push Forward Insertion Heuristic (PFIH). This heuristic iterates inserting customers until a constraint is not satisfied. When the latter occurs, a new route is created, and the procedure begins again. The study in [49] (CVRPTW) extends this procedure by adding a new phase that turns the last customer of a route into the first customer of another route.

Results from [47] show that their GA is suitable for instances of up to 150 customers. Furthermore, their GA was capable of finding the Best Known Solutions (BKS) of benchmark instances at that time, and, in some cases, improving them. Similar results were obtained by [48], where not only high-quality solutions were obtained, but high computational performance. In the case of [49], their results also improved the BKS of some benchmark instances.

Another implicit representation is studied in [50] to solve the simple VRP. The GA encodes the solution with an integer representation. Each integer is the label of a vehicle, whereas the location of that integer represents a customer. That allows the operator to use n-point crossover without additional problems. However, the encoding approach only assigns customers. Hence, the node sequences must be calculated by solving the TSP for each vehicle, using the assigned customers. Their solution approach is capable of solving instances with up to 199 customers.

In the case of E-VRP, as of 2019, 10 out of 78 E-VRP implementations have used GA as a solution approach [23]. That number of implementations turns GA as the second most used heuristic, just after ALNS. In [43], they tackle the E-VRP with charging time and variable travel time. The authors propose a GA with an explicit integer representation. The delimiters are integers greater than all the other node identifiers. As a result, they obtain routes where CSs always appear in the routes. Also, they assume a constant recharging time of 30 minutes. Their results show that GA can find feasible solutions of real-data instances with up to 50 customers in about three hours.

In [44], the authors tackle two problems: optimal routing and charging schedule of EV fleets. The optimal routing aims at finding classical least cost routes, considering a realistic EV consumption model. The charging scheduling addresses the problem of controlling the charging of EVs at private CS and public CS. The latter only applies when the EV does not have enough charge to return to the depot. To solve the problem, they use Differential Evolution (DE), another kind of EA. The DE uses differences between individuals to provide

less stochastic results. Their implementation is tested in an airport shuttle with six stops, obtaining satisfactory results. Another interesting result is that they test their implementation with a battery degradation model, finding that the SOC policy allows the battery to last longer.

3.6 Discussion

In this chapter, a review of GAs and their applications to VRP has been presented. GAs are useful to solve challenging optimization problems where finding the global optimum is difficult, and where the combinatorial explosion occurs. To correctly design a GA, one should carefully encode solutions and carefully design genetic operations. The latter are responsible for providing local search and diversity to solutions, which are the most relevant GA aspects.

The applications of GAs to VRP had its boom in the first decade of the 2000s. Researchers provided novel encodings to solve complex VRP variants with excellent results. Overall, GAs (and metaheuristics in general) are used when the problem addresses realistic elements. Most of their approaches consider implicit integer representations. To decode them, they use iterative insertion based on capacity constraints. However, in the last decade, few methods have been developed to solve the E-VRP. The major drawback is the exploitation of partial recharging and proper GA implementation. None of them have included partial recharging into the solution encoding. Most works use a second method to insert charging operations, whereas others use recharging policies such as fixed recharging time, full recharging, or recharging up to a certain SOC level. Most works are also poorly implemented. That is, the GA takes a lot of time to solve medium-size instances.

Finally, to the best of our knowledge, no E-VRP work where GA is the main solution method has addressed capacitated CS. Only one of them [44] provides a charge scheduling where routes are already assigned. However, this problem is very different from the proposed in this thesis work, where it is aimed to develop a synchronization mechanism among all CS and the fleet.

Chapter 4

Problem Formulation

4.1 Introduction

In this chapter, we formulate two E-VRP variants, each of them focusing on a particular stage. The first variant is called Offline E-VRP (Off-E-VRP), which intends to find EVs' initial routes before they begin the operation. The second variant is the Online E-VRP (On-E-VRP), which addresses the same criteria as the Off-E-VRP, but considering real-time measurements from the state of EVs and the traffic network. Solving the On-E-VRP allows the fleet management system to update the routes in-operation, thus providing a closed-loop control strategy.

4.2 Problem Statement

4.2.1 Offline E-VRP

Consider a directed network $D = (\{0\} \cup N \cup F, A)$, where $\{0\}$ is the depot, $N = \{1, \ldots, n\}$ contains n destinations that must be served, and $F = \{n+1, \ldots, n+s\}$ contains s CS where EVs can recharge their battery. The set V groups all the nodes in the network, i.e. $V = N \cup \{0\} \cup F$. The set A contains all the arcs that connect non-identical network nodes, i.e. $A = \{(i,j) : \forall i, j \in V, i \neq j\}$.

Each customer $j \in N$ requires a demand D_j, which is delivered by an EV. It takes a known time T_j to serve j. A time window $[T_j^-, T_j^+]$ constrains this service, i.e., j is served after T_j^- and before T_j^+. If the EV arrives too early, it must wait until the time window starts. Both T_j^- and T_j^+ belong to the set $\Omega \subset \mathbb{R}^+$, which contains the times of the day. The time window is such that its width is strictly larger than the service time, i.e., $T_j^+ - T_j^- > T_j$.

In order to serve all customers, a fleet M of m EVs is available. Each EV $i \in M$ has three critical limitations: it can carry a mass up to $\bar{D}_i$, it has a maximum battery capacity of $\bar{Q}_i$, and it has a maximum tour duration of $\bar{T}$; the latter is the same for all EVs. If i reaches a customer before its time window begins, it must wait until the time window begins. In some cases, it may be convenient to wait at the previous customer because traffic may vary. Therefore, one can choose to wait at the previous customer after serving it, or at the current customer before serving it. The routes each EV follow must begin and finish at the depot.

In real traffic networks, it is usual to experience higher congestion at certain times of the day. As reviewed in Section 2.5, including both time-dependent travel times and energy

consumption is crucial to provide a high-quality service. From now on, consider the arc $(i, j) \in A$. This arc defines the travel time the EV must spend and the energy amount the EV battery must provide. The travel time is $t_{ij}(t_0)$ and the energy consumption is $E_{ij}(w, t_0)$, where t_0 is the time of the day the EV departs from i towards j, and w is the weight the EV carries across the arc. The calculation of both travel time and energy consumption among nodes is further explained in Section 4.3.

As reviewed in Section 2.3.3, operating the EV battery near zero or one, and over large Depth of Discharge (DOD) leads to a faster capacity and power fade. Therefore, we include a SOC policy that constraints the SOC to operate among well-established boundaries. The SOC policy is defined by the interval $[\alpha^-, \alpha^+]$, where $\alpha^-, \alpha^+ \in [0, 100]\%$ such that $\alpha^+ > \alpha^-$.

When an EV does not have enough energy to complete the tour, it can visit a CS to recharge the battery. A CS $j \in F$ can do up to r_j charging operations at the same time. A single charging operation produces a SOC increment of q; that is, if an EV enters the CS with SOC p, then, it leaves with SOC $p + q$. Such recharging operation takes a time $\Delta_j(p, q)$, which is characterized by a concave nonlinear charging function. To calculate these charging times, we use the method presented by [7], where a piece-wise function approximates the charging function of a CS.

Three variables define the whole operation of EV $i \in M$: the sequence of nodes S^i, the charging plan L^i, and the departure time x_0^i. These three variables form the *route* of i. The sequence of nodes is a vector that contains the nodes i will visit. The charging plan is a vector that stores information about charging operations. This information allows the EV to know when and how much to recharge. Finally, departure time x_0^i defines the moment i begins the operation. A detailed description of these three variables is presented in Section 4.4. From a fleet point of view, variables S, L, and x_0 group the routes of all vehicles. That is $S = [S^1,\ S^2, \ldots,\ S^m]$, $L = [L^1,\ L^2, \ldots,\ L^m]$, and $x_0 = [x_0^1,\ x_0^2, \ldots,\ x_0^m]$.

The problem is to find an optimal set of routes that allows the fleet to fulfill the operation satisfying all operational constraints. The criteria to choose the routes is to minimize total travel times, total energy consumption, total charging times, and total charging costs. A solution to the problem must return: the optimal sequences of nodes (S^*), the optimal charging plans (L^*), and the optimal departure times (x_0^*). A feasible solution satisfies:

- all customers are visited precisely once,
- all EVs do not exceed their limitations,
- CS capacities are not surpassed,
- all EVs begin and end operation at the depot,
- all customers are visited within their time windows.

4.2.2 Online E-VRP (On-E-VRP)

In the Off-E-VRP, travel times and energy consumptions among customers depend on the time of the day. At peak hours, one can expect higher travel time and lower energy consumption due to traffic. This behavior is a *recurrent event* because it similarly repeats every day. Other events like accidents, weather change, or demonstrations cannot be accurately predicted because their occurrence probability is very low. That is why they are called *nonrecurrent events*.

As reviewed in Section 2.5, non-recurrent events can dramatically affect the operation of the fleet. The latter occurs because EVs might undergo unexpected circumstances, leading to scenarios where travel times and energy consumptions are different from those considered in the Off-E-VRP. Besides, real EV speeds are non-deterministic; therefore, the real operation may differ from what was originally planned.

In the Online E-VRP (On-E-VRP), the dispatcher can recalculate the routes based on real-time measurements. These measurements provide information about the current state of the traffic network and the EVs. As a result, the system can react to sudden traffic fluctuations and prevent EVs from following infeasible or sub-optimal routes. Such ability to updates routes based on measurements provides a closed-loop control operation of the EV fleet.

Let k^* be a discrete counter representing the instant when the dispatcher receives measurements from the fleet and the traffic network. Each time the dispatcher collects a new set of measurements, k^* increases by one. At instant k^*, the network is redefined as a directed graph

$$D(k^*) = \left[\{0\} \cup F \cup N(k^*), A(k^*)\right],$$

where $\{0\}$ is the depot, F is the set containing all available CSs, and $N(k^*)$ is the set containing all non-served customers at instant k^*. The set $A(k^*)$ contains all the arcs connecting the network nodes at instant k^*, i.e.

$$A(k^*) = \{(i,j) : \forall i,j \in V(k^*), i \neq j\},$$

where $V(k^*) = N(k^*) \cup \{0\} \cup F$. The initial instant $k^* = 0$ is the moment when the first EV departs from the depot.

Each customer $j \in N(k^*)$ has the same properties stated in the Off-E-VRP. These properties are: a demand D_j, a service time T_j, and a time window $[T_j^-, T_j^+]$ such that $T_j^+ - T_j^- > T_j$. The total number of customers in $N(k^*)$ can vary from k^* to k^*+1 because EVs are currently traveling; thus, they can serve some customers between k^* and k^*+1.

The fleet $M(k^*)$ contains $m(k^*)$ EVs at instant k^*. Each EV $i \in M(k^*)$ must visit a known set of customers $N^i(k^*)$, which have been previously assigned. The customers in $N^i(k^*)$ are such that the EV never carries more weight than its weight limitation $\bar{D}_i$. The maximum tour duration is now referred as maximum remaining time, and it is recalculated as

$$\bar{T}(k^*) = \bar{T} + t_0^i - t_{k^*},$$

where $\bar{T}$ is the maximum tour duration, t_0 is the time of day EV i departs from the depot, and t_{k^*} is the time of the day at the moment of collecting measurements for the k^*-th time.

The fleet size $m(k^*)$ might vary from instant k^* to instant $k^* + 1$ because some EVs finish their operation.

EVs do not necessarily depart from the depot. Their initial conditions may vary from instant k^* to $k^* + 1$ because they move forward as they are operating. At instant $k^* + 1$, the node where EV $i \in M(k^* + 1)$ departs from is called *the critical node of* i, denoted S^i_{crit}. The critical node must be such that the EV reaches it after the optimization at k^* finishes; otherwise, the EV will not receive the route update in time. The dispatcher determines which node in the route at k^* is the critical node at $k^* + 1$. It also determines the EV state when it starts the service there, which is called *critical state*. This state is used to calculate the initial conditions according to the operation that occurs at the critical node: if S_{crit} is a CS, the dispatcher must recalculate the SOC increment the charging operation makes. Section 6.3.1 provides a detailed explanation on how to calculate critical nodes and initial conditions.

Travel times and energy consumptions may vary from instant k^* to $k^* + 1$ as well. The dispatcher can update the values of $t_{ij}(t_0)$ and $E_{ij}(w, t_0)$ for every $(i, j) \in A(k^*)$ at each instant k^*.

The problem is to find a new set of optimal routes each EV in the fleet will follow. These routes will begin at the critical nodes and have the following properties. At instant k^*, the new routes are defined by the new node sequences $S^*(k^*)$, the new charging plan $L^*(k^*)$, and new initial conditions x^*_0. The criteria to choose the new routes hold: minimize travel times, total energy consumption, total charging times, total charging costs, and total waiting times. A feasible solution satisfies the same conditions stated in the Off-E-VRP, except that:

- each EV begins its operation at the critical node S^i_{crit},
- all EVs finish the operation at the depot.

4.3 Travel Times and Energy Consumption Between Nodes

To address the dynamic behavior of velocity due to traffic, we consider that the travel time and energy consumption across arc $(i, j) \in A$ depends on the time of the day the EV departs from i towards j. We denote this time of the day as t_0. The travel time is denoted $t_{ij}(t_0)$ and the energy consumption $E_{ij}(w, t_0)$, where w is the payload the EV carries across (i, j). The values $t_{ij}(t_0)$ and $E_{ij}(w, t_0)$ are referred to as *travel time profile* and energy consumption profile of arc (i, j), respectively.

In this section, we develop two methods to estimate both $t_{ij}(t_0)$ and $E_{ij}(w, t_0)$ throughout the day using numerical data sets. The first method in Section 4.3.1 allows us to calculate travel time and energy consumption using the Euclidean distance between two nodes. The second method in Section 4.3.2 is developed to calculate travel time and energy consumption from shortest paths. In both methodologies, we divide the day into several equally-spaced periods. Then, we assign a known value to travel time and energy consumption at the beginning of each period. We then use interpolation to calculate the values within the beginning of two periods. This procedure allows us to estimate travel time and energy consumption throughout the day.

For energy consumption, we develop an additional two-stage procedure. First, we calculate the energy the EV consumes by traveling from start to destination, carrying no payload. The latter is known as *no-cargo energy consumption*, denoted $E_{ij}(0, t_0)$. Second, we address the problem of including more mass using a realistic EV energy consumption model.

4.3.1 Travel Time and Energy Consumption in Straight Line Arcs

In this section, we develop a method to calculate travel times and energy consumption for straight line arcs. The aim is to estimate $t_{ij}(t_0)$ and $E_{ij}(w, t_0)$ for any valid time t_0 and payload w, and any arc $(i, j) \in A$.

We start by dividing the times of the day into h equally-spaced periods, each of them of length ΔT. We assume that, at the beginning of each period, there are travel time and no-cargo energy consumption measurements available, i.e., for the travel time, we know the values

$$\{t_{ij}(0),\ t_{ij}(\Delta T), \ldots,\ t_{ij}((h-1)\Delta T)\},$$

and for the no-cargo energy consumption, we know the values

$$\{E_{ij}(0,0),\ t_{ij}(0, \Delta T), \ldots,\ t_{ij}(0, (h-1)\Delta T)\}.$$

To calculate the values between the beginning of two periods, we use interpolation. Therefore, we are capable of estimating the travel time and no-cargo energy consumption at any time of the day.

To include more payload mass, we use the following equation

$$E_{ij}(w, t_0) = E_{ij}(0, t_0) \cdot \left[1 + \frac{w}{w_{EV}}\right] - \frac{w}{w_{EV}} \cdot \beta, \tag{4.1}$$

where $E_{ij}(0, t_0)$ is the no-cargo energy consumption, w_{EV} is the EV plus driver mass, w is the payload, and β is value that depends on the travel time at t_0 and the arc length. See Appendix A for a detailed development of Eq. (4.1).

Considering the above, it is crucial to calculate travel times and no-cargo energy consumption at the beginning of each period. In our case, we obtain those values as follows. We use real traffic data from one of Santiago de Chile's most congested areas. That data is used to develop a complete network with several origins and destinations. We solve the DS-SPP using [33] between all nodes in the previous network. This procedure allows us to obtain several travel times and no-cargo energy consumption values between all nodes, at different times of the day. We scale those profiles to match a straight one-kilometer arc profile using the distance between nodes. All one-kilometer arc profiles are then averaged to obtain two one-kilometer arc profiles: one for travel time and one for no-cargo energy consumption. We then use these one-kilometer arc profiles to calculate travel time and no-cargo energy consumption of straight arcs of any length. Such procedure is explained in the following paragraph.

Let $t_{1km}(t_0)$ and $E_{1km}(0, t_0)$ be the travel time and no-cargo energy consumption profiles for the one-kilometer arc, respectively. These profiles are known. To calculate the travel

time across arc $(i, j) \in A$, we use the distance d_{ij} between the two nodes. Then, we make the following assumption: at time t_0, the velocity in all network arcs is the same and has the value $v(t_0)$. Then, we simply obtain the travel time in arc (i, j) as

$$t_{ij}(t_0) = d_{ij} \cdot t_{1km}(t_0).$$

For the energy consumption, we do the exact same procedure. From Appendix A, we see that, in Eq. (A.4), the energy consumption has a linear relationship with the distance. Therefore, we obtain that

$$E_{ij}(0, t_0) = d_{ij} \cdot E_{1km}(0, t_0)$$

In reality, travel time and energy consumption are not linearly related to distance; this assumption is just an approximation. To include more accurate values, one must solve the DS-SPP in the target network. However, this is a computationally expensive procedure. In the next section, we develop a methodology to address such cases and incorporate better approximations of travel time and energy consumption from shortest paths.

4.3.2 Travel Time and Energy Consumption from Shortest Paths

In real life, the travel time and the energy consumption an EV spends depends on the path it follows and the traffic state. In this case, a path refers to a road-level route the EV goes through. As explained in Section 2.4, solving the Shortest Path Problem (SPP) allows an EV to travel from start to destination spending the least cost. However, the length of shortest path the EV follows may differ from the Euclidean distance. Therefore, the procedures to calculate travel time and energy consumption in the previous section may lead to poor estimations.

In this work, we solve the Dynamic and Stochastic Shortest Path Problem (DS-SPP) for several destinations. We use the procedure proposed by [33] which considers a prognosis-based decision-making strategy to solve the DS-SPP. It consists on a multi-stage strategy that begins by solving the deterministic K-SPP. Then, the shortest paths are disturbed to obtain significant differences among them. Finally, the paths are evaluated in detail using a particle-filtering approach. The latter returns a set of particles and weights, which allows us to estimate the probability density functions (pdf) of both travel time and energy consumption.

The algorithm proposed by [33] returns the shortest path, and the pdf of the travel time and energy consumption for a single arc of our problem considering a specific departure time and mass. Therefore, if we only consider the EV mass, we must solve the problem at least $hn \cdot (n-1)$ times, where h is the number of periods in a day and $n \cdot (n-1)$ the number of arcs in the network. Such high number of cases makes solving the DS-SPP computationally impractical. Furthermore, we must address the problem of considering different mass values because the EVs deliver packages; thus, the payload they carry decreases while they operate.

To address the issue of solving the DS-SPP too many times and different mass values across arcs, we make two assumptions. First, late in the night and early in the morning, traffic does not vary considerably; therefore, one can expect very little variations of the travel time and energy consumption pdfs at that hours. Thus, we calculate the pdfs that represent the travel

time and energy consumption in that interval and assign them for the periods that cover that interval. Second, we address the problem of different mass by solving the DS-SPP only considering the EV mass; that is, we calculate the no-cargo energy consumption. Then, we use the methodology explained in Section 4.3.1 to incorporate more mass using the realistic energy consumption model.

Now, we provide an insight on how to calculate travel times and energy consumption from a road-level path using the solution the algorithm in [33] returns. An arc $(i, j) \in A$ in our E-VRP (both offline and online) is, in reality, the concatenation of several road intersections, each of them joined by a road. That is, the path from start i to destination j can be represented as $A_{ij} = \{a_0, a_1, ..., a_n\}$, where each a_k is a road intersection (equivalent to a node in the road-level network). Let t_{k+1} be the travel time from a_k to a_{k+1}. We assume that the EV leaves each a_k instantly. As a result, the total travel time from i to j is the sum of the expected travel time across each road:

$$t_{ij}(t_0) = \sum_{k=0}^{n-1} \mathbb{E}_{t_{k+1}|t_k}[t_{k+1}],$$

where $\mathbb{E}_{t_{k+1}|t_k}[\cdot]$ indicates the expected value of the arrival time at a_{k+1} given that the EV leaves a_k at t_k. The energy consumption follows the same kind of behavior. Is we assume a mass w and a departure time t_0, then:

$$E_{ij}(w, t_0) = \sum_{k=0}^{n-1} \mathbb{E}_{t_{k+1}|t_k}[E_{k+1}(w)]$$

where $E_{k+1}(w)$ is the energy consumption traveling from a_k to a_{k+1} carrying a payload w.

Calculating the above expected values can be challenging because the non-linear behavior of the system makes it difficult to obtain the travel time and energy consumption pdfs. To address this issue, we use the method from [33]. The author develops a prognosis-based method to solve the DS-SPP for EVs using a particle filter approach. Such a method fits our problem, as it returns the estimation of travel time and no-cargo energy consumption pdfs. An additional advantage of estimating the pdfs of travel time and energy consumption using [33] is that we can use them to emulate real traffic conditions. In the simulation framework detailed in Section 6.3.2, we assume that both costs follow normal distributions and their expected value and variance are known. Those values come from solving the DS-SPP using [33].

Including more mass into the EV may lead to path changes to reduce energy consumption. Therefore, it is ideal for solving the DS-SPP according to the current payload the EV carries. Nonetheless, the high number of scenarios increases the complexity of the problem considerably. To address the latter issue, we fix the path the EV follows from i to j and modify the energy consumption using Eq. (4.1). A detailed explanation on how to obtain the latter equation is explained in Appendix A. The following steps summarize this process:

1. Consider the EV leaves i towards j at time t_0.

2. Solve the DS-SPP to find the expected values of $t_{ij}(t_0)$ and $E_{ij}(0, t_0)$ with their pdf estimations.

3. Use Eq. (4.1) to find the expected value of $E_{ij}(w, t_0)$, i.e. adjust the energy consumption according to the current payload the EV carries.

4.4 Electric Vehicle State Space Model

In this section, we introduce an event-based EV state-space model. This model tracks three variables: the time of the day and SOC each EV has, and the payload each EV carries when it starts a service. A service refers to either serve a customer or recharge the battery in a CS. Before defining the state-space model, let us formally introduce the variables that define the routes of EVs and other auxiliary variables.

The sequence of nodes EV $i \in M$ will follow is defined by the vector

$$S^i = [S^i_0, \ldots, S^i_{s_i-1}]^T,$$

where s_i is the length of S^i, and each S^i_k is the k-th node the EV visits. The length s_i depends on the number of customers and the charging operations in the route. The vector

$$L^i = [L^i_0, \ldots, L^i_{s_i-1}]^T$$

has the same length of S^i and defines the charging plan, where each L^i_k represents the SOC increment at the k-th stop. Thus, if S^i_k is a customer, then $L^i_k = 0$. On the other hand, if S^i_k is a CS, a SOC increment must occur, i.e., $L^i_k > 0$.

Notice that we have used the sub-index k to mark the k-th stop. The value k is a discrete counter that tracks the model events. Each event indicates that EV $i \in M$ is at stop k in S^i, that is, at node S^i_k. Therefore, if the vehicle moves forward to the next stop, the counter increases by one, and the EV arrives at node S^i_{k+1}. As the sequence of nodes is made up of s_i nodes, k ranges from 0 to $s_i - 1$. We denote this range of values as $K^i = \{0, 1, ..., s_i - 1\}$ for vehicle i. As each EV has its counter we refer to k as a *local counter*.

If an EV arrives at a customer node before its time window begins, the EV must wait until it starts. However, as reviewed in Section 4.3, energy consumption between two nodes vary throughout the day. Therefore, if the EV waits a certain amount of time before traversing the arc, it may occur that the energy consumption is lower than not waiting at all. Hence, two waiting time options are available:

- $\hat{w}^0(k)$ is the waiting time before starting the time window at S^i_k. If the EV already satisfies the time window, or S^i_k is a CS or the depot, then $\hat{w}^0(k) = 0$.

- $\hat{w}^1(k)$ is the waiting time after finishing the operation at S^i_k. This waiting time allows the EV to accomplish the time window at the next stop, i.e. the time window of node S^i_{k+1}. If the EV already satisfies that time window, or S^i_{k+1} is a CS or the depot, then $\hat{w}^1(k) = 0$.

The above implies that there are two possibilities to allow the EV to accomplish the time window of node S^i_k:

- waiting $\hat{w}^0(k)$ time units at S^i_k right before serving S^i_k, or

- waiting $\hat{w}^1(k-1)$ time units at S^i_{k-1} right after serving S^i_{k-1}.

We choose the option that minimizes the energy consumption across S^i_{k-1} and S^i_k. To do that, assume that the EV goes through arc (S^i_{k-1}, S^i_k). If the EV waits $\hat{w}^1(k-1)$ at S^i_{k-1}, it then consumes an energy E^1. On the other hand, if it waits $\hat{w}^0(k)$ at S^i_k, it consumes an energy E^0. Therefore, the real waiting times are $w^0(k)$ and $w^1(k-1)$, which depend on E^0 and E^1 as follows:

$$w^0(k) = \begin{cases} 0 & \text{if} \quad E^0 > E^1 \\ \hat{w}^0(k) & \text{otherwise.} \end{cases} \tag{4.2}$$

$$w^1(k-1) = \begin{cases} 0 & \text{if} \quad E^1 > E^0 \\ \hat{w}^1(k-1) & \text{otherwise.} \end{cases} \tag{4.3}$$

The time an EV stays in a node is referred to as *stop time*, which includes two elements: how long it takes the operation at that node, and how much to wait before and after the operation. The operation time at the k-th stop is denoted $T_{op}(k)$, which depends on the node type of S^i_k. If S^i_k is a customer, then $T_{op}(k)$ is the service time of S^i_k. If S^i_k is a CS, then $T_{op}(k)$ is the time it takes to perform the charging operation. Finally, if S^i_k is the depot, then $T_{op}(k)$ is zero. These cases are summarized as

$$T_{op}(k) = \begin{cases} T_{S^i_k} & \text{if} \quad S^i_k \in N \\ \Delta_{S^i_k}\left(x_2(k), L^i_k\right) & \text{if} \quad S^i_k \in F \\ 0 & \text{if} \quad S^i_k = 0 \end{cases} \tag{4.4}$$

The requirement at node S^i_k also depends on its type. If S^i_k is a customer, the requirement is known. Otherwise, the requirement is zero:

$$D(k) = \begin{cases} D_{S^i_k} & \text{if } S^i_k \in N \\ 0 & \text{if } S^i_k \in F \cup \{0\} \end{cases} \tag{4.5}$$

The state-space model considers three variables for a single EV at the moment it begins an operation at a node: the time, the battery SOC, and the payload. These variables evolve according to the local counter k. Thus, they are defined as follows:

- $x^i_1(k)$ is the time of the day i begins the operation at S^i_k. At the next stop S^i_{k+1}, the time $x^i_1(k+1)$ is the sum of five elements: the starting operation time at S^i_k ($x^i_1(k)$); the operation time at S^i_k; the waiting time at S^i_k after finishing the operation there; the travel time between S^i_k and S^i_{k+1} when departing from S^i_k; and the waiting time at S^i_{k+1} before starting the operation there. The initial condition $x^i_1(0)$ is the time the EV departs from the first node in S^i.

- $x^i_2(k)$ is the battery SOC when the vehicle begins the operation at S^i_k. At the next stop S^i_{k+1}, the SOC $x^i_2(k+1)$ is the sum of three elements: the SOC when the EV

starts the operation at S_k^i ($x_2^i(k)$); the SOC increment at stop S_k^i; and minus the energy consumption between S_k^i and S_{k+1}^i when departing from S_k^i and subtracting the requirement at S_k^i. The initial condition $x_2^i(0)$ is the battery SOC when the vehicle departs from the first node in S^i.

- $x_3^i(k)$ is the payload when the EV begins the operation at S_k^i. At the next stop S_{k+1}^i, the payload $x_3^i(k+1)$ is the sum of two elements: the payload when the EV starts the operation at S_k^i ($x_3^i(k)$), and minus the requirement at S_k^i. The initial condition $x_3^i(0)$ is the sum of all customer requirements in S^i.

Equations (4.6) to (4.8) define state equations of EV i:

$$x_1^i(k+1) = x_1^i(k) + T_{op}(k) + w^1(k) + t_{S_k^i S_{k+1}^i}\left(x_1^i(k) + T_{op}(k) + w^1(k)\right) + w^0(k+1) \quad (4.6)$$

$$x_2^i(k+1) = x_2^i(k) - e_{S_k^i S_{k+1}^i}\left(x_3^i(k) - D(k),\ x_1^i(k) + T_{op}(k) + w^1(k)\right) + L_k^i \quad (4.7)$$

$$x_3^i(k+1) = x_3^i(k) - D(k) \quad (4.8)$$

To simplify notation, equations (4.6) to (4.8) are reduced into state vector equation (4.9):

$$\begin{bmatrix} x_1^i(k+1) \\ x_2^i(k+1) \\ x_3^i(k+1) \end{bmatrix} = \begin{bmatrix} F_1\left(x^i(k), S_k^i, S_{k+1}^i, L_k^i\right) \\ F_2\left(x^i(k), S_k^i, S_{k+1}^i, L_k^i\right) \\ F_3\left(x^i(k), S_k^i, S_{k+1}^i, L_k^i\right) \end{bmatrix} \quad (4.9)$$

4.4.1 Initial Conditions in the On-E-VRP

In the On-E-VRP, EV $i \in M(k^*)$ does not begin the operation at the depot, but at the critical node S_{crit}^i. As stated by the EV model, initial conditions are defined at the moment i departs from the first node in its route. This node can be the depot (if the EV has not begin the operation yet), a customer, or a CS. Thus, we consider the following cases:

1. if S_{crit}^i is the depot, then we allow the dispatcher to adjust the departure time,
2. if S_{crit}^i is a customer, we allow the dispatcher to alter the waiting time (if any) or add a waiting time after finishing the service,
3. if S_{crit}^i is a CS, we allow the dispatcher to alter the recharging amount.

Considering the above three cases, it is necessary to formally define a relationship between the state when the EV arrives at the critical node, the node type of the critical node, and the initial conditions that will be passed to the On-E-VRP. To do that, the dispatcher receives the following information:

- The type of S_{crit}^i.
- The SOC increase at S_{crit}^i, known as L_{crit}^i. This SOC increase is zero if S_{crit}^i is a customer or the depot. Otherwise, if S_{crit}^i is a CS, L_{crit}^i is a known positive value. The

latter is obtained from the charging plan of previous solutions of the On-E-VRP, or from the Off-E-VRP.

- A triplet $X^i_{crit} = [x^i_{1crit}, x^i_{2crit}, x^i_{3crit}]^T$ known as *critical state*, which indicates the time of the day, battery SOC, and payload EV i has when it begins the service at S^i_{crit}. If the latter is the depot, then the triplet is the departure state.

The critical state and the node type are used to calculate initial conditions. If S^i_{crit} is a customer or the depot, then we only modify departure times using an auxiliary variable x^i_{1off}. This variable is directly added to the departure time to alter its value. Notice that x^i_{1off} is free when the EV is at the depot. However, when the EV is at a customer, x^i_{1off} must be such that the departure time at the customer is equal or greater than the time the EV finishes the service. The SOC x^i_{2crit} and payload x^i_{3crit} remain the same. On the other hand, if S^i_{crit} is a CS, then we only modify the recharging amount using the auxiliary variable L^i_{off}. That value is directly added to L^i_{crit} to alter its value. Notice that this also affects the departure time. Finally, in all cases, we must add the waiting time after service at S^i_{crit}, denoted w^1_{crit}.

Now, we can define the initial conditions using the offset values defined above:

$$
\begin{aligned}
S^i_0 &= S^i_{crit} \\
L^i_0 &= \begin{cases} 0 & \text{if } S^i_{crit} \in N \cup \{0\} \\ L^i_{crit} + L^i_{off} & \text{if } S^i_{crit} \in F \end{cases} \\
x^i_1(0) &= \begin{cases} x^i_{1crit} + T_{S^i_{crit}} + x^i_{1off} + w^1_{crit} & \text{if } S^i_{crit} \in N \cup \{0\} \\ x^i_{1crit} + \Delta(x^i_{2crit}, L^i_0) + w^1_{crit} & \text{if } S^i_{crit} \in F \end{cases} \\
x^i_2(0) &= x^i_{2crit} + L^i_0 \\
x^i_3(0) &= x^i_{3crit} - D_{S^i_{crit}}
\end{aligned}
$$

4.5 Counting the number of vehicles at each node

The operation of different EVs is coupled by the constraint of CS capacity because the presence EVs in a CS reduces the capacity for remaining EVs. In this section, we develop a synchronization method to count the number of EVs at each node and impose the CS capacity constraints. This method uses the start-of-service times from the state-space model defined in Section 4.4 to detect when and where EVs arrive and leave.

We start by defining a global counter $\bar{k}$. This counter is initialized as zero and increases by one when any of the two following events occur: an EV arrives at a node, or an EV departs from a node. Consider the vector $\theta(\bar{k}) \in \mathbb{R}^{|V|}$. At instant $\bar{k}$, the j-th component of $\theta(\bar{k})$ contains the number of EVs at node j when the $\bar{k}$-th event occurs. For example, if at instant $\bar{k}$ two vehicles are visiting node 5, then $\theta_5(\bar{k}) = 2$. If an EV leaves node 5, the event $\bar{k}+1$ "leaving a node" is triggered and $\theta_5(\bar{k}+1) = 1$.

The latter example suggests that vector $\theta(\bar{k})$ evolves dynamically. This dynamic behavior makes the j-th element of $\theta(\bar{k})$ to increase or decrease by one if the event at $\bar{k}$ is *an EV arrives to j*, or *an EV departs from j*, respectively. The latter is summarized by the following

equation:

$$\theta(\bar{k}+1) = \theta(\bar{k}) + \delta(\bar{k})\gamma(\bar{k}) \tag{4.10}$$

where $\delta(\bar{k})$ is a scalar value such that

$$\delta(\bar{k}) = \begin{cases} +1 & \text{if at instant } \bar{k}\text{, a vehicle arrives to a node} \\ -1 & \text{if at instant } \bar{k}\text{, a vehicle leaves a node,} \end{cases} \tag{4.11}$$

and $\gamma(\bar{k}) \in \mathbb{R}^{|V|}$ is a vector such that its j-th component satisfies

$$\gamma_j(\bar{k}) = \begin{cases} 1 & \text{if at instant } \bar{k}\text{, event occurs at node } j \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

The value $\delta(\bar{k})$ tracks the kind of event, whereas the vector $\gamma(\bar{k})$ tracks the node where the event happens. The initial condition of $\theta(\bar{k})$ depends on the problem to solve. In the Off-E-VRP, $\theta(0) = [m, 0, \ldots, 0]^T$, since all m EVs depart from the depot at the beginning of the operation. On the other hand, in the On-E-VRP, $\theta(0) = [\theta_0^*, \theta_1^*, \ldots, \theta_N^*]^T$ where θ_j^* is the number of EVs at node j at the moment of gathering all measurements.

4.6 Formulation as Non-linear Program

4.6.1 Cost Function and Decision Variables

In this section, we formulate the Off-E-VRP and the On-EVRP as non-linear programs. In the Off-E-VRP, decision variables are node sequences S^i, charging plan L^i, and departure times x_0^i of each EV $i \in M$. In the On-E-VRP, decision variables are node sequences S^i, charging plan L^i, and initial condition offsets: t_{off}^i if critical node is a customer, or L_{off}^i if critical node is a CS, for each EV $i \in M(k^*)$.

In both Off-E-VRP and On-E-VRP, the cost has the same definition. Equation (4.13) summarizes it as the weighted sum of four elements: total travel times among nodes, total charging times, total charging costs, and total energy consumption:

$$
\begin{aligned}
J(S,L,x_0) = \quad & \omega_1 \underbrace{\sum_{i=1}^{m}\sum_{k=0}^{s_i-1} t_{S_k^i S_{k+1}^i}\left(x_1^i(k) + T_{S_k^i} + w^1(k)\right)}_{J_1 \text{ (total travel time)}} \\
& + \omega_2 \underbrace{\sum_{i=1}^{m}\sum_{\substack{S_k^i \in F \\ k \in K^i}} \Delta_{S_k^i}\left(x_2^i(k), L_k^i\right)}_{J_2 \text{ (total charging time)}} \\
& + \omega_3 \underbrace{\sum_{i=1}^{m}\sum_{\substack{S_k^i \in F \\ k \in K^i}} \xi_{S_k^i} L_k^i}_{J_3 \text{ (total charging cost)}} \\
& + \omega_4 \underbrace{\sum_{i=1}^{m}\sum_{k=0}^{s_i-1} e_{S_k^i S_{k+1}^i}\left(x_3^i(k) - D(k), x_1^i(k) + T_{S_k^i} + w^1(k)\right)}_{J_4 \text{ (total energy consumption)}}
\end{aligned}
$$

$$
J(S,L,x_0) = \omega_1 J_1 + \omega_2 J_2 + \omega_3 J_3 + \omega_4 J_4 \tag{4.13}
$$

where $S = [S^1, \ldots, S^m]$ $L = [L^1, \ldots, L^m]$, and $x_0 = [x_0^0, \ldots, x_0^m]$ group decision variables of each EV. If S_k^i is a CS, $\xi_{S_k^i}$ is the kWh price. The weights ω_1, ω_2, ω_3, and ω_4 tune the importance of the variables they multiply.

Minimization of travel times allows EVs to visit more customers in less time; thus, increasing incomes. Minimization of charging times lowers the impact of charging operations as they reduce the total maximum tour time available. Minimization of charging costs allows the optimization to differentiate among CS technologies. Finally, minimization of the energy consumption allows EVs to travel further, as battery capacity is limited. In the following sections, the constraints of both the Off-E-VRP and the On-E-VRP are defined.

4.6.2 Off-E-VRP Constraints

The following constraints are initial conditions. Constraint (4.14) forces all EVs to start at the depot; constraint (4.15) indicates that EVs do not recharge when they begin the operation; constraint (4.16) is time of the day the EVs begin operation; constraint (4.17) states that all EVs begin with a SOC equal to the upper SOH policy bound; constraint (4.18) states that each EV carries a weight equal to the sum of all customer requirements in its route.

$$S_0^i = 0 \qquad \forall i \in M \qquad \text{(vehicles start from the depot)} \tag{4.14}$$

$$L_0^i = 0 \qquad \forall i \in M \qquad \text{(vehicles do not recharge at the depot)} \tag{4.15}$$

$$x_1^i(0) = x_0^i \qquad \forall i \in M \qquad \text{(starting time of service)} \tag{4.16}$$

$$x_2^i(0) = \alpha^+ \qquad \forall i \in M \qquad \text{(SOC leaving depot)} \tag{4.17}$$

$$x_3^i(0) = \sum_{S_k^i \in N} D_{S_k^i} \quad \forall i \in M,\ \forall k \in K^i \qquad \text{(Weight freight leaving depot)} \tag{4.18}$$

The following constraints are terminal conditions. Constraint (4.19) forces all EVs to end at the depot; constraint (4.20) prevents EVs to exceed the maximum tour time duration.

$$S_{s_i-1}^i = 0 \qquad \forall i \in M \qquad \text{(vehicles end at depot)} \tag{4.19}$$

$$x_1^i(s_i - 1) \leqslant \bar{T} + x_1^i(0) \qquad \forall i \in M \qquad \text{(maximum service time)} \tag{4.20}$$

Constraint (4.21) prevents EVs from carrying more weight than their maximum weight limitation.

$$\sum_{S_k^i \in N} D_{S_k^i} \leqslant \bar{D}_i \quad \forall i \in M,\ \forall k \in K^i \qquad \text{(vehicles end at depot)} \tag{4.21}$$

Constraints (4.22) and (4.23) define time windows.

$$T_{S_k^i}^- \leqslant x_1^i(k) \qquad \forall i \in M,\ \forall k \in K^i,\ S_k^i \in N \qquad \text{(time window lower bound)} \tag{4.22}$$

$$x_1^i(k) \leqslant T_{S_k^i}^+ - T_{op}(k) \qquad \forall i \in M,\ \forall k \in K^i,\ S_k^i \in N \qquad \text{(time window upper bound)} \tag{4.23}$$

Constraint (4.24) defines the state vector equation.

$$\begin{bmatrix} x_1^i(k+1) \\ x_2^i(k+1) \\ x_3^i(k+1) \end{bmatrix} = \begin{bmatrix} F_1\left(x^i(k), S_k^i, S_{k+1}^i, L_k^i\right) \\ F_2\left(x^i(k), S_k^i, S_{k+1}^i, L_k^i\right) \\ F_3\left(x^i(k), S_k^i, S_{k+1}^i, L_k^i\right) \end{bmatrix} \quad \forall i \in M,\ \forall k \in K^i \qquad \text{(State equation)} \tag{4.24}$$

The following constraints are SOH policies. Constraint (4.25) bounds the SOC when the EV arrives at a node, while constraint (4.26) bounds the SOC when the EV departs.

$$\alpha^- \leqslant x_2^i(k) \leqslant \alpha^+ \qquad \forall i \in M,\ \forall k \in K^i \qquad \text{(SOH policy 1: SOC bounds)} \tag{4.25}$$

$$\alpha^- \leqslant x_2^i(k) + L_k^i \leqslant \alpha^+ \qquad \forall i \in M,\ \forall k \in K^i \qquad \text{(SOH policy 1: SOC bounds)} \tag{4.26}$$

Constraint (4.27) are the dynamics of the counting vector and constraint (4.28) the initial value of the counting vector. Constraint (4.29) limit the maximum parallel charging operations at CSs.

$$\theta(\bar{k}+1) = \theta(\bar{k}) + \delta(\bar{k})\hat{\gamma}(\bar{k}) \quad \forall \bar{k} \in \bar{K} \qquad \text{(counting vector dynamics)} \tag{4.27}$$

$$\theta(0) = [m, 0, \ldots, 0]^T \qquad \text{(initial counting vector)} \tag{4.28}$$

$$\theta_j(\bar{k}) \leqslant r_j \quad \forall j \in F, \ \forall \bar{k} \in \bar{K} \qquad \text{(limit on parallel charging operations)} \tag{4.29}$$

Finally, the following constraints define decision variables domains.

$$S_k^i \in N \cup F \quad L_k^i \in \mathbb{R}_0^+ \quad x_0^i \in \Omega \quad \forall i \in M, \ \forall k \in K^i \qquad \text{(Decision variables domain)} \tag{4.30}$$

4.6.3 On-E-VRP Constraints

Initial conditions must be modified to include the critical point. Constraint (4.31) forces all EVs to start at the critical node; constraint (4.32) defines the SOC increase at the critical node; constraint (4.33) states the time of the day each EV departs from the critical node; constraint (4.17) states the SOC of each EV when it departs from the critical node; constraint (4.18) states that each EV departs carrying a weight equal to the sum of all its assigned customer requirements.

$$S_0^i = S_{crit}^i \qquad \forall i \in M^* \qquad \text{(init. node)} \tag{4.31}$$

$$L_0^i = \begin{cases} 0 & \text{if } S_{crit}^i \in N \cup \{0\} \\ L_{crit}^i + L_{off}^i & \text{if } S_{crit}^i \in F \end{cases} \qquad \forall i \in M^* \qquad \text{(init. SOC increae)} \tag{4.32}$$

$$x_1^i(0) = \begin{cases} x_{1crit}^i + T_{S_{crit}^i} + x_{1off}^i + w_{crit}^1 & \text{if } S_{crit}^i \in N \cup \{0\} \\ x_{1crit}^i + \Delta(x_{2crit}^i, L_0^i) + w_{crit}^1 & \text{if } S_{crit}^i \in F \end{cases} \qquad \forall i \in M^* \qquad \text{(init. time)} \tag{4.33}$$

$$x_2^i(0) = x_{2crit}^i + L_0^i \qquad \forall i \in M^* \qquad \text{(init. SOC)} \tag{4.34}$$

$$x_3^i(0) = x_{3crit}^i - D_{S_{crit}^i} \qquad \forall i \in M^* \qquad \text{(init. weight)} \tag{4.35}$$

End condition constraint (4.19) holds, as EVs must return to the depot. Constraint (4.36) updates the maximum remaining tour time.

$$x_1^i(s_i - 1) \leqslant \bar{T}^* + x_1^i(0) \quad \forall i \in M \qquad \text{(maximum service time)} \tag{4.36}$$

Constraint (4.21) is discarded because the offline problem has already assigned customers. Thus, this constraints will always be satisfied.

Constraints (4.37) and (4.38) define time windows just on customers each EV must visit.

$$T_{S_k^i}^- \leqslant x_1^i(k) \quad \forall i \in M, \ \forall k \in K^i, \ \forall S_k^i \in N^i \qquad \text{(time window lower bound)} \tag{4.37}$$

$$x_1^i(k) \leqslant T_{S_k^i}^+ - T_{op}(k) \quad \forall i \in M, \ \forall k \in K^i, \ \forall S_k^i \in N^i \qquad \text{(time window upper bound)} \tag{4.38}$$

State vector dynamics remain the same; thus, constraint (4.24) holds.

SOH policy constraints (4.25) and (4.26) remains the same.

The initial condition of the counting vector (4.28) is replaced by constraint (4.39), which contains the number of vehicles at each node at the moment of beginning the new operation.

$$\theta(0) = [\theta_0^*, \theta_1^*, \ldots, \theta_{n+s}^*]^T \qquad \text{(initial counting vector)} \tag{4.39}$$

Decision variables domains are replaced as:

$$\begin{aligned}
S_k^i &\in N^i \cup F \\
L_k^i &\in \mathbb{R}_0^+ \\
x_{1off}^i &\in \begin{cases} \{0\} & \text{if } S_{crit}^i \in F \\ [-w_{crit}^1, \infty) & \text{if } S_{crit}^i \in N \cup \{0\} \end{cases} \quad \forall i \in M(k^*) \quad \forall k \in K^i \quad \text{(Decision variables domain)} \\
L_{off}^i &\in \begin{cases} \{0\} & \text{if } S_{crit}^i \in N \cup \{0\} \\ [-L_{crit}, \infty) & \text{if } S_{crit}^i \in F \end{cases}
\end{aligned} \tag{4.40}$$

4.7 Discussion

In this chapter, two new E-VRP variants have been introduced. The first one is the Off-E-VRP, which determines the routes each EV will follow before starting the operation. The second one is the On-E-VRP, which intends to update the routes according to measurements of the traffic network and EVs' current state. The problems are modeled as nonlinear programs, which are quite different from many common VRP variants modeled as linear mixed programs. The main reason to model a nonlinear program is to drop the dependency of CS copies, which increment the complexity of the problem.

Chapter 5

Solution Scheme for the Off-E-VRP and the On-E-VRP

5.1 Introduction

In chapter 4, two E-VRP variants have been presented: the Off-E-VRP and the On-E-VRP. The Off-E-VRP aims to find the optimal routes before starting the operation. These routes are constructed using historical data of travel time and energy consumption among nodes. The On-E-VRP closes the control loop by updating the routes when real-time measurements of the traffic and vehicle state are available. The significant differences among both problems require two different solution methods that adjust to each problem's nature.

For the Off-E-VRP, the solution is obtained before EVs begin their operation. Thus, it is assumed that there is enough time to solve the problem. For the On-E-VRP, new routes must be found in a small amount of time while EVs are serving customers. However, the exploration space is smaller because customers are already assigned, and the routes previously calculated are good initial candidates.

In this chapter, three Genetic Algorithms (GA) are introduced. The first two GAs, αGA and βGA, aim to solve the Off-E-VRP. αGA has the property of assigning customers to each EVs. βGA only works when customers have already been assigned, that is, it can be used to improve the routes αGA finds. The third GA, onGA, aims to solve the online problem. Hence, this algorithm must be fast in comparison to αGA and βGA. To solve it fast, individuals use a different encoding scheme to find solutions in fewer generations, and the fitness evaluation procedure does not evaluate all constraints.

All GAs evaluate fitness and handle constraints following the same procedure. This procedure is detailed in Section 5.2. αGA is explained in Section 5.5, βGA in Section 5.6, and onGA is explained in section 5.7.

5.2 Fitness Evaluation and Constraint Handling

Any GA in charge of solving any of the problems in Chapter 4 will decode individuals into S (routing plan), L (charging plan), and x_0 (initial conditions). These variables are used to evaluate all constraints in the problem. Therefore, the fitness evaluation procedure and constraint handling schemes are the same for any GA that solves the problems described in Chapter 4.

Let $\Gamma(I_j)$ be the fitness of individual I_j. In this work, it is defined as:

$$\Gamma(I_j) = -J\left(\Psi(I_j)\right) - \Pi(I_j, \mathcal{K}_1), \tag{5.1}$$

where $J(\cdot)$ is the cost function in Eq. (4.13), $\Psi(I_j)$ decodes I_j into decision variables S, L, and x_0, $\mathcal{K}_1$ is a large positive number, and $\Pi(\cdot)$ is a penalization function that only affects infeasible individuals.

Using a penalization function is one of several other methods to handle constraints [51]. This scheme has been chosen because of its simplicity and efficacy to handle continuous variables. The penalization method makes individuals that store infeasible solutions less fitted when they are far from the feasible search space: *the more the infeasible, the less the fitness*. In the following, we introduce a formal definition of the penalization function.

Consider that the problem has p inequality constraints and, to evaluate them, it requires q variables. The optimization vector $\bar{x}_j \in \mathbb{R}^q$, obtained from individual I_j, contains all this necessary variables, whereas matrix $A \in \mathbb{R}^{p\times q}$ and vector $b \in \mathbb{R}^p$ are such that all inequality constraints in the problem can be written as

$$A\bar{x}_j \leqslant b.$$

To measure how infeasible an individual is, we use a quadratic penalization distance function $D_i(I_j)$, defined as:

$$D_i(I_j) = \begin{cases} w_i \left(A_i\bar{x}_j - b_i\right)^2 & \text{if not } A_i\bar{x}_j \leqslant b_i \\ 0 & \text{otherwise,} \end{cases} \tag{5.2}$$

where A_i is the i-th row in A, b_i is the i-th element in b, and w_i is a weight greater than zero. The latter weight tunes the importance of constraints, that is, one can state that some constraints are more important than others. Using the distance function and allowing certain infeasible routes to be accepted, the penalization of individual I_j is the weighted sum of a cumulative distance and the large constant value:

$$\Pi(I_j, \mathcal{K}_1) = \begin{cases} \sum_{i=1}^{p} D_i(I_j) + \mathcal{K}_1 & \text{if} \quad \text{a constraint is violated} \\ 0 & \text{if} \quad \text{no constraint is violated.} \end{cases}, \tag{5.3}$$

where $\mathcal{K}_1 \in \mathbb{R}^+$ is a large constant. Notice that, using this penalization method, a feasible individual will always be better than an infeasible individual.

In our case, equality constraints are hard constraints; thus, they will always be achieved. These constraints originate from the EV dynamic model and the θ vector model. The algorithm evaluates these models using decision variables. Therefore, no further methodologies are required to tackle these kinds of constraints.

To calculate the fitness of I_j, the algorithm follows the next steps. First, decode I_j into S, L, and x_0 using $\Psi(I_j)$. Second, decision variables are passed as inputs to the EV dynamic model to calculate the state variables and the vector θ throughout the operation (see Section 4.4 and Section 4.5). This procedure results in an optimization vector $\bar{x}_j$, which contains all variables of the problem, i.e., it is a candidate solution. Third, once $\bar{x}_j$ is calculated,

the algorithm checks its feasibility. Fourth, if the solution is infeasible, the algorithm must calculate the penalization. Fifth, the algorithm calculates the cost function. Sixth, the algorithm calculates fitness.

In practical E-VRP implementations, construct $\bar{x}_j$, A, and b is quite inefficient. The insertion of CS alters the length of the routes, which alters the length of the optimization vector as well. One cannot a-priori know the number of charging operations the EVs require. Therefore, if we want to implement the matrix comparison, we must construct $\bar{x}_j$, A, and b for each individual in the population, for each generation. This is severely impractical.

To address the latter issue, we do not use a matrix comparison. On the other hand, we make use of data structures such as dictionaries and tuples in Python, which have an average time complexity $O(1)$ when requesting an item. That allows us to store the values from the optimization and compare constraints directly.

5.3 Encoding Partial Recharging Operations

As reviewed in Section 3.4, GAs require encoding the solutions in a precise manner. In the case of E-VRPs, most previous works on GAs use an integer representation that stores routes. However, none of them tackle a case where continuous variables are present in the problem. As a result, no encoding mechanism to represent partial recharging exists.

This section introduces two novel encoding mechanisms that allow GAs to search for optimal partial recharging operations. Both mechanisms rely on the concept of charging paths (see Section 2.6.2). A charging path connects two non-CS nodes (i.e., the depot or customers) including one or more CS between them. Therefore, these representations allow GAs to insert CS in the sequence between two non-CS nodes. These representations receive the names of Charging Path Blocks (CPBs). The notion *block* is because, as explained in the next sections, they are a portion of the individual, not the individual itself.

5.3.1 Charging Path Block: Type 1

A CPB Type 1 (CPB-1) stores n^* charging operations, where n^* is an hyper-parameter of the algorithm. A CPB-1 has the following structure:

$$[\underbrace{a_1^1, a_2^1, a_3^1}_{a^1}, \underbrace{a_1^2, a_2^2, a_3^2}_{a^2}, \ldots, \underbrace{a_1^{n^*}, a_2^{n^*}, a_3^{n^*}}_{a^{n^*}}],$$

where each a^j is a vector of size three that represents a charging operation. These charging operations are defined as follows. Let a^j be any charging operation, that is:

$$a^j = [a_1^j, a_2^j, a_3^j],$$

where $a_1^j \in N \cup \{-1, 0\}$, $a_2^j \in F$, and $a_3^j \in \mathbb{R}^+$. The interpretation of a^j is the following. After visiting node a_1^j, the EV will recharge at CS a_2^j the amount a_3^j. After that charging operation, the EV will continue its original path. However, if a_1^j is -1, then the recharging operation does not take place at all. In that case, a_2^j and a_3^j have no relevance.

For example, consider two EVs: EV 1 and EV 2. Their sequences of customers are $[5, 2, 4, 7]$ for EV 1, and $[3, 1, 9, 8, 6]$ for EV 2. Still, we want to obtain the following routes and charging plans:

$$S^1 = [0, 5, 2, 4, \mathbf{11}, 7, 0] \qquad S^2 = [0, 3, 1, 9, \mathbf{12}, 8, 6, 0]$$
$$L^1 = [0, 0, 0, 0, \mathbf{25.2}, 0, 0] \quad L^2 = [0, 0, 0, 0, \mathbf{33.5}, 0, 0, 0],$$

where the bold text indicates a charging operation. The following CPB-1 with $n^* = 3$ can insert the charging operations in those sequence of customers:

$$[\underbrace{9, 12, 33.5}_{a^1}, \underbrace{-1, 11, 50.5}_{a^2}, \underbrace{4, 11, 25.2}_{a^3}].$$

This CPB-1 alters both sequence of customers by doing the following. The operation a^1 states that the EV that visits customer 9 (EV 2 in this example) will visit, just after serving that customer, CS 12 and increase the SOC level by 33.5%. Similarly, operation a^3 state that the EV that visits customer 4 (EV 1 in this example) will visit CS 11 and recharge 24.2% right after serving that customer. Operation a^2 does not have relevance because it has a -1 in the first position. Therefore, values 11 and 50.5 do not have any effect.

Notice that some CPB-1 may repeat some customers. For example, the charging operations $a^1 = [3, 11, 20.5]$ and $a^2 = [3, 12, 22.5]$ affect customer 3. In that case, the charging operations are inserted in the order they appear. That is, CS 11 is inserted right after customer 3. Then, CS 12 follows the same directions, and it is inserted right after customer 3 as well. This method leads to the sequence 3-12-11.

5.3.2 Charging Path Block: Type 2

A CPB Type 2 (CPB-2) stores n_i charging operations, where n_i is the number of customers EV $i \in M$ must visit. Therefore, this encoding method only works when EVs have assigned customers. Consider that EV i has the following sequence of customers

$$[b_1, b_2, \ldots, b_{n^i}].$$

The CPB-2 is a vector of size $2n_i$ with the following structure

$$[\underbrace{c_1^1, c_2^1}_{c^1}, \underbrace{c_1^2, c_2^2}_{c^2}, \ldots, \underbrace{c_1^{n_i}, c_2^{n_i}}_{c^{n_i}}],$$

where $c_1^j \in F \cup \{-1\}$ and $c_2^j \in \mathbb{R}^+$ for any $j \in \{1, 2, \ldots, n_i\}$. The interpretation is as follows. After visiting customer b_j, the EV will visit CS c_1^j and recharge the amount c_2^j, for any $j \in \{1, 2, \ldots, n_i\}$. If c_1^j is -1, then the charging operation does not occur at all.

Let us consider the same example from the previous section. That is, the sequences of customers are $[5, 2, 4, 7]$ for EV 1, and $[3, 1, 9, 8, 6]$ for EV 2. The desired routes and charging plans are:

$$S^1 = [0, 5, 2, 4, \mathbf{11}, 7, 0] \qquad S^2 = [0, 3, 1, 9, \mathbf{12}, 8, 6, 0]$$
$$L^1 = [0, 0, 0, 0, \mathbf{25.2}, 0, 0] \quad L^2 = [0, 0, 0, 0, \mathbf{33.5}, 0, 0, 0],$$

Therefore, two CPB-2 are required: one for EV1, and one for EV2. The following two CPB-2 candidates will produce the desired routes:

$$(CPB-2)^1 = [\underbrace{-1, 35.7}_{c^1}, \underbrace{-1, 11.5}_{c^2}, \underbrace{\mathbf{11}, \mathbf{25.2}}_{c^3}, \underbrace{-1, 22.7}_{c^4}]$$
$$(CPB-2)^2 = [\underbrace{-1, 15.8}_{c^1}, \underbrace{-1, 14.8}_{c^2}, \underbrace{\mathbf{12}, \mathbf{33.5}}_{c^3}, \underbrace{-1, 52.1}_{c^4}, \underbrace{-1, 5.2}_{c^5}]$$

$(CPB-2)^1$ states that, after serving the third customer in its sequence, EV 1 will visit CS 11 and recharge 25.2%, i.e. after customer 4. Similarly, $(CPB-2)^2$ states that, after serving the third customer in its sequence, EV 2 will visit CS 12 and recharge 33.5%, i.e. after customer 9.

5.4 Considerations for Genetic Operations

Selection operation does not depend on the structure of individuals, but on their fitness. Therefore, we allow all GAs to use the same selection method. All GAs implement a deterministic tournament selection of size Υ. To prevent losing the best individual, all GAs implement elitism. Elitism is done by keeping the best κ individuals of generation k, and appending them into generation $k+1$. The worst κ individuals of generation $k+1$ are dismissed.

Regarding mutation and crossover operations, we allow them to choose which procedure they perform from a pool of operations. That allows the algorithm to perform local search with high probability, and global search with high probability. Due to the enormous number of probability combinations one could set, we fix the probability of occurrence of operations. This probability is detailed with the description of the operation.

5.5 αGA

5.5.1 Encoding

Each individual is the concatenation of three blocks: a customers' block, a charging operations' block, and a departure times' block. The customers' block stores the sequence of customers using a direct integer representation. That is, the integer in position k is the k-th customer the EV will visit in its sequence. The delimiter is the symbol '|', which indicates the end of a customer sequence. The charging operations' block is a CPB-1 (see Section 5.3.1) that inserts charging operations between the customers defined in the customers' block. The departure times' block encodes departure times in minutes using a real representation.

The length of the individual is

$$L = N + 3m + 3n^*,$$

where N is the number of customers in the network, m is the fleet size, and n^* is the number of charging operations the CPB-1 stores. Therefore, allowing more charging operations will increase the complexity of the algorithm. As this is the first time an encoding of this type

is used, it is hard to determine an exact value for n^*. In this work, we use the following heuristic

$$n^* = 2m.$$

Let us illustrate this encoding with an example. Consider a fleet of size $m = 2$, a maximum number of allowed charging operations $n^* = 3$ (the heuristic is not used here), and the following routes, charging plans, and departure times for each EV:

$$\begin{aligned} S^1 &= [0, 5, 2, 4, 11, 7, 0] & S^2 &= [0, 3, 1, 9, 12, 8, 6, 0] \\ L^1 &= [0, 0, 0, 0, 25.2, 0, 0] & L^2 &= [0, 0, 0, 0, 33.5, 0, 0, 0] \\ x_1^1(0) &= 510.0 \quad \text{(8:30 AM)} & x_1^2(0) &= 540.0 \quad \text{(9:00 AM)}. \end{aligned}$$

These decision variables can be encoded using the following individual candidate:

$$[\overbrace{\underbrace{5, 2, 4, 7, |}}^{\text{Route 1}}, \overbrace{\underbrace{3, 1, 9, 8, 6, |}}^{\text{Route 2}}, \overbrace{\underbrace{9, 12, 33.5,}}^{a^1} \overbrace{\underbrace{-1, 11, 50.5,}}^{a^2} \overbrace{\underbrace{4, 11, 25.2,}}^{a^3} \overbrace{\underbrace{510.0}}^{\text{Dep. time 1}} , \overbrace{\underbrace{540.0}}^{\text{Dep. time 2}}]$$

Customers' block | Charging operations' block (CPB-1) | Departure times' block

5.5.2 Initial Population

The GA creates an initial population in three steps using the procedures described in appendix B.1:

i. The algorithm receives a candidate individual constructed with a heuristic (see Section 6.2).

ii. Create new individuals by mutating the individual from the previous step. Repeat until one-third of the population is filled.

iii. Fill the population with random individuals.

5.5.3 Mutation and Crossover Operations

To perform any of the two operations, the GA selects one of the three blocks in the individual with the same probability. Then, it performs mutation or crossover according to the block type they modify. We allow the operator to perform several operations, each of them with a different probability of occurrence. Of course, if only one operation can occur, it receives a probability of one.

Customers' block

- **Mutation**
  - Chooses a customer sequence randomly. Then, the operator swaps two customers in that sequence. (Probability: 0.4)
  - Swaps two random values in the block, without considering the last "|" symbol. The latter must always remain there. (Probability: 0.4)

– Chooses a random customer in a customer sequence. Then, the operator removes that customer from the sequence and inserts it into another sequence randomly. (Probability: 0.4)

- **Crossover**

  – The operator combines two randomly selected customer sequences doing the following. First, notice that some customers may repeat among sequences. Thus, the operator finds these repeated customers. Second, the operator removes repeated customers from the individual. Third, the repeated customers are inserted in other customer sequences in a random position. Fourth, the original sequences are swapped among individuals. (Probability: 1).

Charging operations' block

- **Mutation**

  – First, randomly selects a charging operation a^j. Second, samples three values: a^* from $N \cup \{-1\}$, b^* from F, and c^* from $U(-10, 10)$. Third, the charging operation is modified as follows:

$$a^j = [a_1^j + a^*, a_2^j + b^*, |a_3^j + c^*|].$$

  (Probability: 0.8)

  – First, randomly selects a charging operation a^j. Second, samples three values: a^* from $N \cup \{-1\}$, b^* from F, and c^* from $U(20, 30)$. Third, the charging operation is modified as follows:

$$a^j = [a^*, b^*, c^*].$$

  (Probability: 0.2)

- **Crossover**

  – Permutes two charging operation in the same positions among two individuals. (Probability: 1.0)

Departure times' block

- **Mutation**

  ▪ Picks a random index i in the range of the block's length. Consider that the block contains the value d_i. The operator replaces d_i by $|d_i + a^*|$, where a^* is sampled from $U(-60, 60)$. (Probability: 1)

- **Crossover**

  – Permutes two values in a random selected position. (Probability: 1).

5.5.4 Algorithm Overview

Algorithm 2 summarizes all procedures αGA performs.

Algorithm 2: αGA

Input : Network $D(V, A)$; Fleet M; Hyper-parameters (H); constructed candidate individual (I^0)

Output: Optimal node sequences, charging plan, and departure times (S^*, L^*, x_0^*)

initialization;
$\mathcal{P} \leftarrow []$;
Append I^0 to $\mathcal{P}$;
Fill $\mathcal{P}$ up to $H[popSize]/3$ with mutations of I^0;
Fill the rest of $\mathcal{P}$ with random individuals;

loop;
for *generation* $\leftarrow 1$ **to** $H[maxGenerations]$ **do**
  eliteIndividuals←Clone the best $H[\kappa]$ individual in $\mathcal{P}$; `/* Save` κ `best individuals */`
  $\hat{\mathcal{P}}$ ←`TournamentSelection`$(\mathcal{P}, H[\Upsilon])$;
  for *child* in $\hat{\mathcal{P}}$ with probability $H[MUTPB]$ **do** `/* Mutate */`
    child←`Mutate-A`(*child*, H);
  end
  for $(child1, child2)$ in $\hat{\mathcal{P}}$ with probability $H[CXPB]$ **do** `/* Mate */`
    $(child1, child2)$ ←`Crossover-A`(*child*1, *child*2, H);
  end
  Insert *eliteIndividuals* in $\hat{\mathcal{P}}$; `/* Elitism */`
  Remove the $H[\kappa]$ individuals with lowest fitness from $\hat{\mathcal{P}}$;
  $\mathcal{P} \leftarrow \hat{\mathcal{P}}$; `/* Update population */`
end
bestIndividual← Individual in $\mathcal{P}$ with the highest fitness.;
$(S^*, L^*, x_0^*) \leftarrow \Psi^{\alpha}(bestIndividual)$;
return S^*, L^*, x_0^*

5.6 βGA

5.6.1 Encoding

This GA intends to solve the problem when each EV has been assigned to a set of customers. Therefore, we opt for a different simplistic encoding that will allow the algorithm to explore the search space with new genetic operations. In this work, βGA will be used to improve the routes αGA finds.

The operation of a single EV is stored in a *sub-individual*. A sub-individual is the concatenation of three blocks: a customers' block, a charging operations' block, and a departure times' block. Do not confuse with the blocks of αGA. In the latter, the blocks form the

individual. In this algorithm, the blocks form a sub-individual.

Consider EV $i \in M$. The customers' block stores the sequence of customers i will visit using a direct representation. No delimiter is required because the number of customers is known. Right after the customers' block comes the charging operations' block. In this case, the sub-individual uses a CPB-2 (see Section 5.3.2), which uses the customers' block to create the charging paths. The departure times' block represents the departure time from the depot using a single real number.

The whole individual is the concatenation of all the sub-individuals that state the routes of each EV:

$$[I^1, I^2, \ldots, I^m],$$

where I^i is the sub-individual of EV i, and m is the fleet size.

As an example, let us represent the same example from Section 5.5.1 using this encoding. Remember the routes, charging plans, and departure times are:

$$\begin{aligned} S^1 &= [0, 5, 2, 4, 11, 7, 0] & S^2 &= [0, 3, 1, 9, 12, 8, 6, 0] \\ L^1 &= [0, 0, 0, 0, 25.2, 0, 0] & L^2 &= [0, 0, 0, 0, 33.5, 0, 0, 0] \\ x_1^1(0) &= 510.0 \quad \text{(8:30 AM)} & x_1^2(0) &= 540.0 \quad \text{(9:00 AM)}. \end{aligned}$$

The two following sub-individual candidates I^1 and I^2 store the routes and departure times of EV 1 and EV 2, respectively:

$$I^1 = [\ \underbrace{5, 2, 4, 7}_{\text{Custs. block 1}}\ ,\ \underbrace{-1, 11.2, -1, 27.8, 11, 25.2, -1, 24.9}_{\text{Charg. ops. block 1 (CPB-2)}},\ \underbrace{510.0}_{\text{Dep. time block 1}}\ ]$$

$$I^2 = [\underbrace{3, 1, 9, 8, 6}_{\text{Custs. block 2}}, \underbrace{-1, 37.0, -1, 18.9, 12, 33.5, -1, 9.1, -1, 39.6}_{\text{Charg. ops. block 2 (CPB-2)}}, \underbrace{540.0}_{\text{Dep. time block 2}}\ ]$$

As a result, the full individual is the concatenation of both sub-individuals:

$$I = [I^1, I^2].$$

The result from αGA may have a charging path with multiple CS between two non-CS nodes. As βGA uses the solution from αGA as an initial point and the CPB-2 only allows charging paths with a single CS, we must turn the charging path with multiple CS visits into a one with a single visit. To do that, we just simply take the last charging operation in the path. We allow this behavior as a different exploration method.

5.6.2 Initial Population

This GA constructs an initial population in two steps, using the methods detailed in appendix B.2:

i. Turn the best individual found by αGA into an individual with a βGA encoding, using `ConstructIndividual-B` (algorithm 9). The encoding βGA uses does not allow

charging paths with more than one CS. However, αGA does. Therefore, if there are routes with more than one CS in the charging path, `ConstructIndividual-B` will only consider the last CS. We discuss this issue at the final discussion of this chapter.

ii. Fill the population with random individuals.

5.6.3 Mutation and Crossover Operations

The GA perform mutation and crossover over sub-individuals. Hence, the first step is to choose any sub-individual randomly, with equal probability. After that, the GA does any of the following operations. As with αGA, this GA is allowed to choose among a pool of operations with a certain probability.

Customers' Block

- **Mutation**
  - Permutes two values randomly. (Probability: 0.55)
  - Splits the block into two pieces A and B. Then, it swaps the pieces to produce a block BA. (Probability: 0.4)
  - Replaces the whole block by a new randomly-generated block. This new random block is a permutation of customers the EV must visit. (Probability: 0.05)
- **Crossover**
  - Permutes the entire block among sub-individuals. (Probability: 1.0)

Charging Operations' Block

- **Mutation**
  - Alters a single charging operation. First, it samples a random index i in the length of the customers' block. Thus, the charging operation is located in the index $i_0 + 2 \cdot i$, where i_0 is the position where the CPB-2 begins in the individual. Second, remember that a charging operation in a CPB-2 has a structure $\{c_1^i, c_2^i\}$. The operator changes c_1^i by a number sampled from $F \cup \{-1\}$, and c_2^i by $|c_2^i + c^*|$, where c^* is a value sampled from $U(-10, 10)$. (Probability: 1.0)
- **Crossover**
  - Permutes the entire block among individuals (Probability: 0.5).
  - Performs a left-1-point crossover among the two blocks (Probability: 0.5).

Departure Times' Block

- **Mutation**

- If the value in the block is a, the operator replaces it by $|a + a^*|$, where a^* is sampled from $U(-20, 20)$. (Probability: 1.0)

- **Crossover**

  - Permutes the values among individuals. (Probability: 1.0)

5.6.4 Algorithm Overview

Algorithm 3 summarizes βGA. Notice that this algorithm requires the routes found by αGA.

Algorithm 3: βGA

Input : Network $D(V, A)$; Fleet M; Routes found by αGA (S, L, x_0); Hyper-parameters (H)

Output: Improved node sequences, charging plan, and departure times (S^*, L^*, x_0^*)

initialization;
$\mathcal{P} \leftarrow$ `InitialPopulation-B`(S, L, V, M);
loop;
for *generation* $\leftarrow 1$ **to** $H[maxGenerations]$ **do**
  eliteIndividuals$\leftarrow$ `CloneBest`$(\mathcal{P},\ H[\kappa])$; `/* Save` κ `best individuals */`
  $\hat{\mathcal{P}} \leftarrow$`TournamentSelection`$(pop, H[\Upsilon])$;
  for *child* in $\hat{\mathcal{P}}$ with probability $H[MUTPB]$ **do** `/* Mutate */`
    child$\leftarrow$`Mutate-B`$(child,\ H)$;
  end
  for $(child1, child2)$ in $\hat{\mathcal{P}}$ with probability $H[CXPB]$ **do** `/* Mate */`
    $(child1, child2) \leftarrow$`Crossover-B`$(child1,\ child2,\ H)$;
  end
  Insert *eliteIndividuals* in *offspring*;
  Remove the $H[\kappa]$ individuals with lowest fitness from $\hat{\mathcal{P}}$;
  $pop \leftarrow \hat{\mathcal{P}}$;
end
bestIndividual$\leftarrow$ Individual in $\mathcal{P}$ with the highest fitness.;
$(S^*, L^*, x_0^*) \leftarrow \Psi^{\beta}(bestIndividual)$;
return S^*, L^*, x_0^*

5.7 onGA

5.7.1 Encoding scheme

This algorithm solves a problem equivalent to βGA: find routes when customers are already assigned. The main difference is that the number of EVs and the customers they visit will change throughout the operation due to the system's online behavior. As a result, the individual will change its size each time the optimization occurs. We remark that, to decode this individual, the GA knows the critical node where the EV begins the operation (see

Section 4.2.2).

The representation is very similar to the one used by βGA. The whole individual is the concatenation of m^* sub-individuals, where m^* is the fleet size at the moment of the optimization. Each sub-individuals stores the information of a single EV in three blocks: a customers' block, a charging operations' block, and an offset block. The customers' block stores the sequence of customers the EV will follow. As with βGA, no delimiter is required because the number of assigned customers for each vehicle is known. However, the charging operations' block is a CPB-1, not a CPB-2. The number of charging operations the CPB-1 stores is n^*, a hyper-parameter of the algorithm. The offset block stores the offset that will be applied to initial conditions.

Customers' blocks do not contain the critical nodes because they are always inserted at the beginning of the routes. Therefore, it does not make sense to store them in the individuals. To allow EVs to visit CSs after the critical node, we allow charging operations to occur after visiting them doing the following. Consider EV $i \in M$, and remember that a single operation a^j is defined by three numbers (see Section 5.3.1):

$$a^j = [a_1^j, a_2^j, a_3^j].$$

In onGA, a^j belongs to the set of customers the EV has not visited considering the critical node, that is, $a_1^j \in N^i \cup \{S_{crit}^i\}$, which differs from αGA and βGA, where $a_1^j \in N$ and $a_1^j \in N^i$, respectively.

Let us provide a visualization of this encoding, continuing with the example from Section 5.5.1. Consider that the optimal routes and charging plans given by any other optimizer were:

$$\begin{aligned} S^1 &= [0,5,2,4,11,7,0], & S^2 &= [0,3,1,9,12,8,6,0] \\ L^1 &= [0,0,0,0,25.2,0,0], & L^2 &= [0,0,0,0,33.5,0,0,0], \\ x_1^2(0) &= 510.0, & x_1^2(0) &= 540.0. \end{aligned}$$

If no customer has been visited yet and each EV is allowed to do at most $n^* = 2$ charging operations, two sub-individuals candidates are:

$$I^1 = [\ \underbrace{5,2,4,7}_{\text{Custs. block 1}}\ ,\ \underbrace{-1,11,23.7,4,11,25.2}_{\text{Charg. ops. block 1 (CPB-1)}},\ \underbrace{10.0}_{\text{Offset block 1}}\ ]$$

$$I^2 = [\underbrace{3,1,9,8,6}_{\text{Custs. block 2}},\underbrace{9,12,33.5,-1,11,24.5}_{\text{Charg. ops. block 2 (CPB-1)}},\ \underbrace{-5.0}_{\text{Offset block 2}}\ ]$$

Then, the full individual is the concatenation of both sub-individuals:

$$I = [I^1, I^2].$$

As the operation of the vehicles has not started yet, the critical point is the depot. Hence, the departure times are the ones stated by either αGA or βGA. To continue with the examples from Section 5.5.1, consider that EV 1 departs at 510 min (8:30 AM), and EV 2 departs at 540 min (9:00 AM). As the offsets of EV 1 and EV 2 are 10.0 and -5.0, respectively, their

departure times are modified to 520 min (8:40 AM) for EV 1, and 535 min (8:55 AM) for EV 2.

For the sake of clarity, we provide two more examples considering that EVs have moved forward following their original routes. The first example addresses a case where both critical points are customers. Consider that, after measuring, the following will happen:

- EV 1 has visited customer 5 and it is currently at customer 2. It is estimated that, when the EV departs from customer 2, the time will be 660, and the SOC will be 64%.
- EV 2 has visited customers 3 and 1, and it is currently at customer 9. It is estimated that, when the EV departs from customer 9, the time will be 695, and the SOC will be 62%.

Now, consider the following two sub-individuals for each EV:

$$I^1 = [\ \underbrace{7,4}_{\text{Custs. block 1}}\ , \underbrace{7,11,22.7,-1,11,25.2,}_{\text{Charg. ops. block 1 (CPB-1)}}\ \underbrace{1.0}_{\text{Offset 1}}\ ]$$
$$I^2 = [\ \underbrace{6,8}_{\text{Custs. block 2}}\ , \underbrace{9,12,30.5,-1,11,24.5,}_{\text{Charg. ops. block 2 (CPB-1)}}\ \underbrace{0.0}_{\text{Offset 2}}\ ]$$

As a result, both individuals lead to the following two new decision variables:

$$\begin{aligned} S^1 &= [2,7,11,4,0], & S^2 &= [9,12,6,8,0] \\ L^1 &= [0,0,22.7,0,0], & L^2 &= [0,30.5,0,0,0], \\ x_1^2(0) &= 661.0, & x_1^2(0) &= 695.0. \end{aligned}$$

In this second example, the critical point of EV 1 is a customer, but the critical point of EV 2 is a CS. Consider that:

- EV 1 has visited customers 5 and 2, and it is currently at customer 4. It is estimated that, when the EV departs from customer 4, the time will be 698, and the SOC will be 42%.
- EV 2 has visited customers 3, 1, and 9, and it is currently at CS 12. The EV arrives at CS 12 at minute 715, with SOC 40%, and it is supposed to recharge 33.5 %.

Now, consider the following two sub-individuals for each EV:

$$I^1 = [\ \underbrace{7}_{\text{Custs. block 1}}\ , \underbrace{4,11,28.7,-1,11,25.2,}_{\text{Charg. ops. block 1 (CPB-1)}}\ \underbrace{5.0}_{\text{Offset 1}}\ ]$$
$$I^2 = [\ \underbrace{6,8}_{\text{Custs. block 2}}\ , \underbrace{-1,12,31.5,-1,11,24.5,}_{\text{Charg. ops. block 2 (CPB-1)}}\ \underbrace{-8.0}_{\text{Offset 2}}]$$

In this case, both individuals lead to the following two new decision variables:

$$\begin{aligned} S^1 &= [4, 11, 7, 0], & S^2 &= [12, 8, 6, 0] \\ L^1 &= [0, 25.2, 0, 0], & L^2 &= [25.5, 0, 0, 0], \\ x_1^2(0) &= 703.0, & x_1^2(0) &= 712.0 + \Delta_{12}(0.4, 0.335 - 0.08). \\ x_2^2(0) &= 42.0\%, & x_2^2(0) &= 0.4 + 0.335 - 0.08. \end{aligned}$$

5.7.2 Initial Population

Consider that the optimization occurs at instant k^*. We assume that the state of both traffic and EVs might have changed due to uncertainty. However, we suppose that the optimal routes found at instant $k^* - 1$ are good initial candidates. The latter assumption is true when no significant events happened between $k^* - 1$ and k^*. As a result, we adapt the old optimal routes in $k^* - 1$ to be valid in k^* and construct an individual. The rest of the population is filled with random individuals. The procedure is done using the methods detailed in appendix B.3:

i. Construct an individual with `ConstructIndividual-O` passing routes starting right after the critical nodes (algorithm 10).

ii. Fill the rest of the population with random individuals.

5.7.3 Mutation and Crossover Operations

This GA also performs genetic operations over sub-individuals, like βGA. First, the GA randomly chooses the sub-individual to alter with equal probability. Then, one of the following pool of operations is done.

Customers' Block

- **Mutation**
  - Permutes two values randomly. (Probability: 0.85)
  - Splits the block into two pieces A and B. Then, it swaps the pieces to produce a block B-A. (Probability: 0.1)
  - Replaces the whole block by a new randomly-generated block. This new random block is a permutation of customers the EV has not visited yet, i.e., a permutation of N^i. (Probability: 0.05)
- **Crossover**
  - Permutes the whole blocks among individuals. (Probability: 1.0)

Charging Operations' Block

- **Mutation**

- Samples three values: a^* from $N^i \cup \{-1\}$, b^* from F, and c^* from $U(-10, 10)$, where N^i is the set of remaining customers EV i must visit. Then, in the sub-block, a is replaced by a^*, b is replaced by b^*, and c is replaced by $|c + c^*|$. (Probability: 1.0)

- **Crossover**

  - Swaps the whole CPB-1 among the two sub-individuals. (Probability: 1.0)

Offset Block

- **Mutation**

  - Consider the value in the block is v, and an empty variable a. If the critical node is a customer, a is assigned a value sampled from $U(-8, 8)$; otherwise, the value is sampled from $U(-5, 5)$. Then, the operator samples b from $B(1, 0.8)$. Finally, the block value is replaced by $b \cdot |v + a|$. The binary variable b allows the operator to reset the offset to zero sometimes, as it might be a good candidate that got lost after going through too many genetic operations. (Probability: 1.0)

- **Crossover**

  - Permutes the values among both individuals. (Probability: 1.0)

5.7.4 Algorithm

The onGA procedure is shown in Algorithm 4.

5.8 Discussion

This chapter introduces three GAs. Two of them, αGA and βGA aim at solving the off-E-VRP, whereas onGA aims at solving the on-E-VRP. To encode routes, three novel representations are introduced. These representations are capable of storing partial recharging, which none of the reviewed works in GA to solve E-VRP have done before. Although the benefits of these encoding operations are many, we remark some critical aspects of using them.

When using a CPB-1, one should pass the maximum number of allowed charging operations n^*. However, no method exists to correctly choose this value. Another drawback is that the algorithm must search for the customers to insert the charging operation. That could be computationally expensive. The main benefit of this representation is that it allows EVs to visit several CS in a charging path. When using a CPB-2, one can improve the efficiency of decoding operations. However, this representation does not allow multiple CS in a charging path. We allow this encoding to explore the search space using different genetic operations.

Finally, we remark that we allow genetic operations to choose which action to perform from a pool of operations. Some of them receive more probability than others. The operations with less probability tend to be destructive, and that is the main reason to lower their probability.

Algorithm 4: `onGA`

Input : Updated network $D(V(k^*), A(k^*))$; Fleet $M(k^*)$; Critical points (X_{crit}); Hyper-parametes (H); Max. execution time (t_{GA})

Output: New routes and initial conditions S^*, L^*, x_0^*

initialization;

$\mathcal{P} \leftarrow$ `InitialPopulation-O`(S, L, X_{crit});

$time \leftarrow 0$

loop;

while $time < t_{GA}$ **do** `/* Internal time, not by variable assignation */`

  eliteIndividuals$\leftarrow$ `CloneBest`$(\mathcal{P},\ H[\kappa])$; `/* Save` κ `best individuals */`

  $\hat{\mathcal{P}} \leftarrow$`TournamentSelection`$(\mathcal{P}, H[\Upsilon])$;

  for *child* in $\hat{\mathcal{P}}$ with probability $H[MUTPB]$ **do** `/* Mutate */`

    child$\leftarrow$`Mutate-O`$(child,\ H)$;

  end

  for $(child1, child2)$ in $\hat{\mathcal{P}}$ with probability $H[CXPB]$ **do** `/* Mate */`

    $(child1, child2) \leftarrow$`Crossover-O`$(child1,\ child2,\ H)$;

  end

  Insert *eliteIndividuals* in *offspring*;

  Remove the $H[\kappa]$ individuals with lowest fitness from $\mathcal{P}$;

  $\mathcal{P} \leftarrow \hat{\mathcal{P}}$;

end

bestIndividual$\leftarrow$ Individual in $\mathcal{P}$ with the highest fitness.;

$(S^*, L^*, x_0^*) \leftarrow \Psi^{On}(bestIndividual, X_{crit})$;

return S^*, L^*, x_0^*

Chapter 6

Online Decision-Making System for the E-VRP

6.1 Introduction

Every day, the dispatcher must solve the Off-E-VRP before starting the operation. This procedure constructs the tentative routes each vehicle will follow. This stage is known as *pre-operation stage*. The pre-operation stage ends when an EV begins its service tour. Immediately after the first EV begins its service tour, the dispatcher begins solving the On-E-VRP. This latter event initiates the *online operation stage*. In this stage, the dispatcher continuously receives data from all EVs and the traffic network and uses it to solve the On-E-VRP. After solving the latter problem, the dispatcher is capable of updating the routes of EVs.

The EVs operate in a very asynchronous manner. When measuring their state, some of them are traveling across an arc, others are serving customers, others may not have begun their operation trip yet, and others may have returned to the depot already. Instead of tackling the asynchronous behavior directly, we develop a synchronization method. This method estimates the future state of the vehicles, based on the current measurements. Using these estimations, we select the nodes in the routes that are ahead of the current positions. These nodes receive the name of *critical nodes*, and they will be the starting nodes of the new routes the on-E-VRP will calculate.

The synchronization layer is responsible for performing the synchronization process. This process allows the system to create a new On-E-VRP instance, which is solved with OnGA. Both processes (synchronization and instance creation) are repeated in a loop until all vehicles reach the depot. Once all vehicles reach the depot, the online operation ends, and the whole day of operation is finished.

In Section 6.2, we detail the pre-operation stage to develop initial routes. In Section 6.3, we detail the online stage. Here, the method to calculate critical nodes and critical states is fully explained. Also, we include the simulation method to emulate real-traffic networks.

6.2 Pre-operation Stage

The pre-operation stage is the first procedure the system performs in the day. We consider the scenario where the dispatcher decides how many EVs to use employ. Therefore, the pre-operation stage is responsible for making three decisions:

1. decide how many EVs to use,
2. assign each EV a set of customers to visit;
3. and generate initial routes for the EVs.

The first step is to solve αGA. The dispatcher uses Algorithm 7 from Appendix B.1 to generate a candidate individual for αGA. The algorithm orders customers according to their time window lower bound in increasing order. It then traverses the ordered customers, adds them into a new route, and each time the requirements in the route surpass the weight limit of EVs, a new route is started. As a result, Algorithm 7 generates a solution candidate and the initial fleet size.

After the initial fleet size is calculated, the system solves αGA. When βGA finishes, the system codes the routes from αGA into a βGA encoding. Then, βGA is executed. After this process, the system verifies if any of the two algorithms returns a feasible solution. If that is the case, the routes with the best optimization result, i.e., best fitness between αGA and βGA, are selected as the initial routes. Otherwise, the system inserts a new vehicle into the fleet, and, from now on, it develops initial solution candidates with Algorithm 8 from Appendix B.2. The latter procedure creates route candidates with a fixed fleet size; that is, it does not calculate how many EVs to use. Algorithm 5 summarizes the pre-operation stage according to the previous steps.

If the fleet owner cannot modify the fleet size, i.e., cannot insert more EVs, then the only option is to use Algorithm 8. In practical scenarios, one can develop a method to select how many EVs to use each day using both methods. For example, if the system determines that the operation can be fulfilled with 10 EVs, and there are 15 EVs available, we could use Algorithm 8 to solve the problem with the 15 EVs anyways. Then, we can compare the benefits and drawbacks of each strategy in terms of operational cost and externalities.

6.3 Online Stage

The online stage begins when an EV starts its operation. At that moment, the system begins collecting measurements from the traffic network and the EVs. Also, we initialize the global On-E-VRP counter k^* to zero. Then, every $t_{GA} + t_{safe}$ time units, a new measurement is available, and k^* increases by one. t_{GA} is the maximum time the dispatcher gives OnGA to find a new solution, i.e., OnGA iterates until its execution time reaches t_{GA} or it reaches the maximum number of generations. t_{safe} is a time frame that allows the dispatcher to communicate the new routes to EVs, gather new measurements, and calculate critical nodes and critical points safely. Then, after a measurement is collected, the dispatcher solves the On-E-VRP using OnGA. The latter process repeats until all vehicles have finished their

Algorithm 5: `Pre-operation stage`

Input: Traffic network ($\{0\} \cup N \cup F, A$); description of a single EV ($\aleph$); αGA hyper-parameters (H^{α}); βGA hyper-parameters (H^{β}); maximum number of EV insertions ($\delta\bar{M}$)

$I^0,\ m_0 \leftarrow$ ConstructIndividual1-A(N, $\aleph$.maxPayload) (Algorithm 7) ;
$m \leftarrow m_0$;
$feasible \leftarrow$ False;
while not $feasible$ **do**
  $I^{*\alpha} \leftarrow \alpha\texttt{GA}((V, A),\ H^{\alpha},\ m,\ I^0)$;
  $I^{*\alpha}_{\beta} \leftarrow$ Turn $I^{*\alpha}$ into βGA encoding;
  $I^{*\beta} \leftarrow \beta\texttt{GA}\big((V, A),\ H^{\beta},\ m,\ I^{*\alpha}_{\beta}\big)$;
  if $\Pi(I^{*\alpha}, H^{\alpha}.\mathcal{K}_1) == 0$ or $\Pi(I^{*\beta}, H^{\beta}.\mathcal{K}_1) == 0$ **then**
    `/* Stop optimization */`
    feasible$\leftarrow$ *True;*
  else
    `/* Add new EV */`
    $m \leftarrow m + 1$;
    if $m - m_0 > \delta\bar{M}$ **then**
      $I^* \leftarrow \arg\max_{I \in \{I^{*\alpha}, I^{*\beta}\}} \Gamma(I)$;
      return $\Psi(I^*)$
    end
    $I^0 \leftarrow$ConstructIndividual2-A(N) (Algorithm 8);
  end
  $I^* \leftarrow \arg\max_{I \in \{I^{*\alpha}, I^{*\beta}\}} \Gamma(I)$;
  return $\Psi(I^*)$
end

operation.

When collecting a measurement, some EVs may be traveling across an arc, others may be serving customers, others may not have begun their operation trip yet, and others may have already returned to the depot. Such asynchronous behavior requires a synchronization method that allows the system to solve the On-E-VRP correctly. This synchronization method consists of predicting the future state of EVs since the moment of observation. In particular, we focus on predicting the moments each EV will arrive and begin the service at the next nodes in its route, after the measurement.

By predicting the moment each EV will begin the service at future nodes, we can safely choose a node so that there is enough time to execute OnGA and solve the On-E-VRP. That is, choose a node such that the EV arrives after $t_{GA} + t_{safe}$ since the moment of measuring. The first nodes that ensure the latter condition are selected as the starting nodes of the new routes. These nodes receive the name of *critical nodes*, which are explained in the next section.

The online stage is summarized in Algorithm 6. In its essence, this stage solves the On-

Algorithm 6: `OnlineOperation`

Input: Traffic network (V, A); Fleet M; Initial Routes (S^0, L^0, x_0^0); onGA hyperparameters (H^{on}); Max. OnGA time (t_{GA}); Safe time (t_{safe})

$done \leftarrow False$;
while not *done* **do**
  $(V, A) \leftarrow$`UpdateNetwork()`;
  $X_{meas} \leftarrow$`GetVehiclesCurrentStates()`;
  $S_{crit}, X_{crit} \leftarrow$`CriticalNodes`$((V, A), X_{meas}, t_{GA}, t_{safe}, S^0, L^0, x_0^0)$;
  $M \leftarrow$`WhichVehiclesToRoute`$(S_{crit}, X_{crit}, (V, A))$;
  if M is not empty **then**
    $(S^*, L^*, x_0^*) \leftarrow$`onGA`$((V, A),\ M,\ S_{crit},\ X_{crit},\ H^{on}))$;
    $S^0, L^0, x_0^0 \leftarrow$`UpdateRoutes`$(S^*, L^*, x_0^*)$;
    Send new routes to EVs;
  else
    $done \leftarrow True$;
  end
end

E-VRP to adjust the portion of the routes that come after the critical points, according to collected measurements. The steps to do that are:

1. Calculate critical nodes.
2. Develop an On-E-VRP instance according to measurements.
3. Execute OnGA to solve the On-E-VRP instance.
4. Send the new routes to all EVs.
5. Loop until all EVs reach the depot.

The solution given by onGA may be equal or better than the old solution. The latter occurs because onGA constructs an initial individual based on the current routes. Therefore, routes are *always* updated, even if new routes are equal to previous routes. The latter procedure is called `UpdateRoutes`, which receives the new node sequences S^*, the new charging plan L^*, and the new initial conditions x_0^* from OnGA. Algorithm 6 summarizes the online method.

6.3.1 Synchronization Layer

In Section 4.2.2, the On-E-VRP is introduced as the problem of finding new routes for each vehicle. The problem assumes that the vehicles start at well-defined nodes, and the initial conditions at that node are known. As mentioned before, those starting points are the critical nodes. In reality, EVs are not necessarily in the critical nodes. Therefore, we need a procedure to make EV states appear like they start from the critical nodes in the On-E-VRP. This procedure is called *state synchronization*, which is performed by the *synchronization layer*.

State synchronization consists of forecasting the future state of EVs based on their current

position and traffic state. The method focuses on forecasting the time and SOC each EV has by starting the service at the next nodes, assuming the current route (previous to updating it with OnGA). Once every start-of-service time and SOC are calculated, the goal is to find the first node such that the EV begins a service after $t_{GA} + t_{safe}$. The node that accomplish such feature is the critical node. We label the start-of-service EV state at the critical node as *critical state*. As a result, in the On-E-VRP, all EVs start at the critical nodes, i.e., we synchronize the EV states to begin at them.

Before introducing the method to forecast EV states, we explain how critical nodes are used. First, let k^* be the global On-E-VRP counter, which tracks the measurement instant. Therefore, at instant k^*, the fleet is $M(k^*)$, which may vary its size from previous instants because some EVs finish their operation. The old route, charging plan, and initial conditions of EV $i \in M(k^*)$ are denoted $S^i(k^* - 1)$, $L^i(k^* - 1)$ and $x_0(k^* - 1)^i$ respectively. As EVs are moving, some nodes in $S^i(k^* - 1)$ might have been already visited.

We illustrate the latter with an example. Consider two EVs that, at certain measurement instant $k^* - 1$, are following the next sequence of nodes:

$$S^1 = [4, 2, 5, \mathbf{6}, 13, 25, 12, 0] \quad S^2 = [12, \mathbf{3}, 11, 10, 8, 7, 0].$$

At the next instant k^*, the synchronization layer calculates the critical nodes, denoted in bold font. As a result, in the On-E-VRP, 6 and 3 will be fixed, and OnGA will update the routes after them, without moving the last 0, which represents the depot:

$$S^1(k^*) = [\mathbf{6}, \underbrace{13, 25, 12,}_{\text{Adjust}} 0] \quad S^2(k^*) = [\mathbf{3}, \underbrace{11, 10, 8, 7,}_{\text{Adjust}} 0].$$

Notice that the previous example only addresses the sequence of nodes. However, the charging plan may also be updated.

To estimate arrival EV states at future nodes, we must address two cases. At the moment of measuring, the EV can be either traversing an arc or performing a service. The latter includes serving a customer or recharging the battery. In both cases, we develop a different forecasting mechanism that estimates the arrival time and SOC at the next node in the current sequence. Then, the arrival times and SOC at future nodes are calculated. We then include waiting times to calculate start-of-service instants and use EV state equations from Section 4.4 to calculate the start-of-service states at future nodes.

In the first case, consider that, at measurement instant k^*, EV $i \in M(k^*)$ is traveling across arc $(S^i_j, S^i_{j+1}) \in A(k^*)$. The synchronization layer receives the arc portion η that the EV has traveled. Thus, the remaining portion is $(1 - \eta)$, and we use this value to estimate the time and SOC i has when arriving at S_{j+1}. To do that, consider that the current time of the day is t^*, and the current weight the EV carries is w^*. Then, do the following:

1. arrival time at S^i_{j+1} is estimated as $t^* + (1 - \eta) \cdot t_{S^i_j S^i_{j+1}}(t^*)$;

2. arrival SOC at S^i_{j+1} is estimated as $e^* - (1 - \eta) \times E_{S^i_j S^i_{j+1}}(w^*, t^*)$,

where w^* is the mass i carries across the arc, and $t_{S_j^i S_{j+1}^i}(\cdot)$ and $E_{S_j^i S_{j+1}^i}(\cdot)$ are the updated travel time and energy consumption functions at instant k^*.

In the second case, at measurement instant k^*, EV $i \in M(k^*)$ is performing a service at node $j \in V(k^*)$. The synchronization layer receives the time the EV departs from that node, including the waiting time after that service. Then, the synchronization layer calculates the arrival time at the next node.

After the arrival time at the next nodes are calculated for each EV, the state synchronization process is done. We then calculate the start-of-service times in the remaining nodes in the old route using the state equation from Section 4.4. For each EV $i \in M(k^*)$, the system then finds the first node where the start-of-service time is greater than $t^* + t_{GA} + t_{safe}$, where t^* is the time of the day the measurement is collected. This node is fixed as the critical node, denoted S_{crit}^i. The critical state is denoted $[x_{1crit}^i, x_{2crit}^i, x_{3crit}^i]^T$. The latter is passed to the On-E-VRP to calculate the initial conditions.

6.3.2 Simulation Framework

In this section, we detail the methodology to simulate a real traffic network. This simulation involves two processes: updating the traffic network, and emulate the movement of EVs. Updating the traffic network state refers to updating travel times and energy consumption between nodes based on current traffic congestion. That is, redefine each $t_{ij}(t_0)$, and each $E_{ij}(w, t_0)$ for each $(i, j) \in A$. The latter procedure is referred as `UpdateNetwork()`, which is latter used in the online stage. Emulate the movement of EVs requires making each EV follow the routes calculated by the decision-making strategy, at the same time they go through the new definitions of travel times and energy consumptions.

Developing a method to compute shortest paths from real traffic data and real-time measurements is not among this thesis goals. A more in-depth study on real-time updates of shortest paths can be found in [33]. There, the author provides an online methodology to solve the Dynamic-Stochastic Shortest Path Problem (DSSP) and develops a PF-based method to estimate the probability density functions (pdf) of travel times and energy consumption. We use this method to find the best routes using real data because it provides the pdfs, which are then used to simulate the traffic network's realizations.

Also, modelling traffic congestion with realistic elements is a difficult problem by itself, and it is not among this thesis goals. To emulate real-traffic networks, we simplify the problem by using time travel and no-cargo energy consumption pdfs we obtain after solving the DS-SPP using [33]. Then, we can calculate the expected value and the standard deviation for each arc and each timestamp with these pdfs. Then, it is assumed that the distributions are normally distributed. As a result, the simulation emulates a noisy network, where the expected values are known, and traffic realizations are generated using the expected values with an additive white Gaussian noise. The latter simulation is valid as we intend to verify that OnGA is capable of updating routes if traffic congestion varies.

Following the methodology from Section 4.3.2, we divide the times of the day in h periods of equal length. For each period start, we store the expected travel time and no-cargo energy consumption between nodes. In this simulation framework, we also store the standard

deviation of the travel time and no-cargo energy consumption at each period start. We obtain those statistics by solving the DS-SPP using [33].

The following procedure emulates the update of travel times and energy consumption. Consider the arc $(i, j) \in A(k^*)$, at measurement instant k^*. Its travel time follows a distribution $N(\mu_{ij}^{tt}(t_0), \sigma_{ij}^{tt}(t_0))$, while the no-cargo energy consumption $N(\mu_{ij}^{ec}(t_0), \sigma_{ij}^{ec}(t_0))$. Therefore, each update is done as follows:

1. Do $a \leftarrow sample\left(N\left(0, \sigma_{ij}^{tt}(t_0)\right)\right)$ for each timestamp t_0.
2. Do $b \leftarrow \text{sample}\left(N\left(0, \sigma_{ij}^{ec}(t_0)\right)\right)$ for each timestamp t_0.
3. Update $t_{ij}(t_0) \leftarrow [\mu_{ij}^{tt}(t_0) + a]$.
4. Update $\bar{E}_{ij}(t_0) \leftarrow [\mu_{ij}^{ec}(t_0) + b]$.

To emulate the movement of EVs, we consider a sample time ΔT^*. Then, each time the network is updated, the simulator checks the route of each EV. Then, it makes each EV move forward according to the route and the traffic network. We record the arrival and departure states of EVs when they reach nodes. When they are traveling, we record the arc portion the EV traverses when it travels a time ΔT^*.

Chapter 7

Simulation Tests

7.1 Experimental Setup

We conduct two studies to test the overall decision-making system performance. In the first study, we study the pre-operation stage by solving the Off-E-VRP over several artificially-generated and two real-data based instances. The results emulate the initial routes each EV will follow. In the second study, we use the traffic data from one of the two real-data based instances to emulate a real traffic behavior, according to Section 6.3.2. Emulating a real traffic behavior allows us to study the online stage by solving the On-E-VRP until all EVs finish their operation.

In both experiments, we consider a homogeneous EV fleet consisting of several Nissan Leaf vehicles. The real parameters and operational constraints of these EVs are detailed in Table 7.1. CS technologies can be slow, normal, and fast. Those technologies have the charging functions shown in Figure 7.1, which are extracted from [7] and scaled to match the battery capacity of our experiments. The time breakpoints in those charging functions remain unaltered.

7.1.1 Cost function weights

To set the cost functions weights, we use the following heuristic. As travel times and energy consumption varies throughout the day, we pick their average value for the whole day. These values are calculated in next Section 7.1.2 and shown in Figure 7.2. Then, we set the weight for travel time and energy consumption as the quotient between a number within $(0, 1]$ and the average value. A similar procedure is applied for recharging time and recharging costs, dividing by the average time it takes to fully recharge the battery and the average cost per kWh, respectively. Table 7.2 shows the final weights used in this case. In this case, we give more importance to energy consumption and recharging costs.

7.1.2 Artificially Generated Instances

We create two sets of instances with different areas: a 10 km by 10 km square, and a 20 km by 20 km square. For each area, we consider 10, 25, 50, 75, and 100 customers; and for each number of customers, we vary the number of CS two times. This procedure yields a total of 20 instances.

Customers demands are created by randomly sampling the following elements from uniform distributions. Time windows are defined within 9:00 and 15:00, with a width ranging

Table 7.1: Description of the EV energy consumption model parameters considering a Nissan Leaf EV [52], and operational constraints

Description	Parameter	Value	Unit
Mass (EV + driver + payload)	m	1521	$[kg]$
Rolling resistance coefficients	C_r	1.75	-
	c_1	4.575	-
	c_2	1.75	$[s/m]$
Air mass density	ρ_{air}	1.2256	$[kg/m^3]$
Frontal are of the EV	A_f	2.3316	$[m^2]$
Gravitational acceleration	g	9.8	$[m/s^2]$
Aerodynamic EV drag coefficient	C_D	0.28	-
Driveline efficiency	$\eta_{Driveline}$	0.92	-
Electric motor efficiency	η_{Motor}	0.91	-
EV battery efficiency	$\eta_{Battery}$	0.9	-
Regenerative braking efficiency	η_{RB}	$\begin{cases} e^{\left(\frac{-0.0411}{\lvert a(t)\rvert}\right)} & \text{if } a(t) < 0 \\ 0 & \text{otherwise.} \end{cases}$	-
Battery capacity	$\bar{Q}_i$	24	[kWh]
Maximum payload	$\bar{D}_i$	1200 (santiago 22 and 6)	[kg]
		553(other instances)	[kg]
Maximum tour time	$\bar{T}$	360	[min]
SOC upper bound	α^+	95%	-
SOC lower bound	α^-	20%	-

Table 7.2: Cost function weights

Weight	Value it multiplies	Value
ω_1	Travel time	45.085e-3
ω_2	Recharging time	1e-2
ω_3	Recharging cost	2e-3
ω_4	Energy consumption	2.291

from 2.5 hours up to 3.5 hours. Service times are within 8 to 15 minutes, whereas requirements within 10 kg to 80 kg. For each CS, we randomly pick with equal probability one of the three charging technologies and assign it to the CS. In all cases, CS capacities are set to three.

Each instance consists of a depot at the square's center, and customers and CS randomly positioned around it. We also consider, in all cases, a CS with normal technology at the depot. Node positions are created by sampling two values from $\mathcal{U}(-l/2, l/2)$, where l is the side length of the square. Travel time and zero-cargo energy consumption between nodes are generated using the average values from the profile in Figure 7.2. This profile is obtained by averaging the real data in Section 7.1.3 and scaling the data into a 1-kilometer arc equivalent. Then, we scale the profile values according to the euclidean distance between the instance nodes.

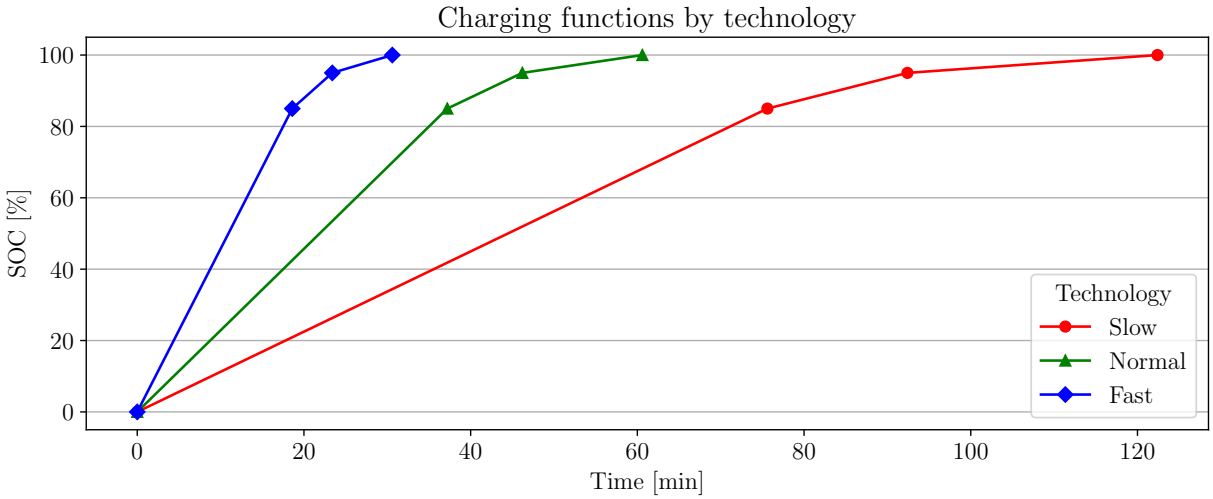


Figure 7.1: Definition of charging functions considered by the EV model, according to their technology (adapted from [7] for a 24 [kWh] battery)

7.1.3 Instances Based on Real-data

To test the solution method's performance over real-world scenarios, we develop two instances using a database containing information of one of Santiago de Chile's most congested areas. The database contains a road network covering a portion of Santiago, Estación Central, Providencia, and Las Condes districts. The database's geographical distribution is shown in Figure 7.3, which contains 326,637 nodes and 664,964 arcs definitions, representing road intersections and road segments, respectively. The database also provides the average velocity every 30 minutes, longitude, latitude, altitude, and miscellaneous elements such as road types and street names.

Using the database described above, we develop a 6-node instance and a 22-node instance. For the 6-node instance, we fix the depot and select five nodes around it, where four of them are customers, and one is a CS with normal technology. For the 22-node instance, we fix the depot and select 20 nodes around it, representing the customers. In both cases, nodes are placed by hand, selecting reachable points in highly-demanded streets.

As explained in Section 4.3.2, we solve the DS-SPP between all nodes using the method in [33]. The latter allows us to estimate the distribution of travel times and no-cargo energy consumption throughout the day. We then use those distributions to emulate real traffic conditions according to what is explained in Section 6.3.2. As solving the DS-SPP is a computationally expensive procedure, we apply the method from Section 4.3.2 where we fix time intervals where travel times and no-cargo energy consumption remain the same. In this case, we only calculate the shortest paths within 6:00 and 22:00. From 00:00 to 6:00, we assume the same shortest path calculated at 6:00, whereas from 22:00 to 00:00, we assume the shortest path calculated at 22:00.

Similarly to artificially developed instances, customers demands are created by randomly sampling the following elements from uniform distributions. Time windows are defined within 9:00 and 16:00, with a width ranging from 2 hours up to 3 hours. Service times are within 5

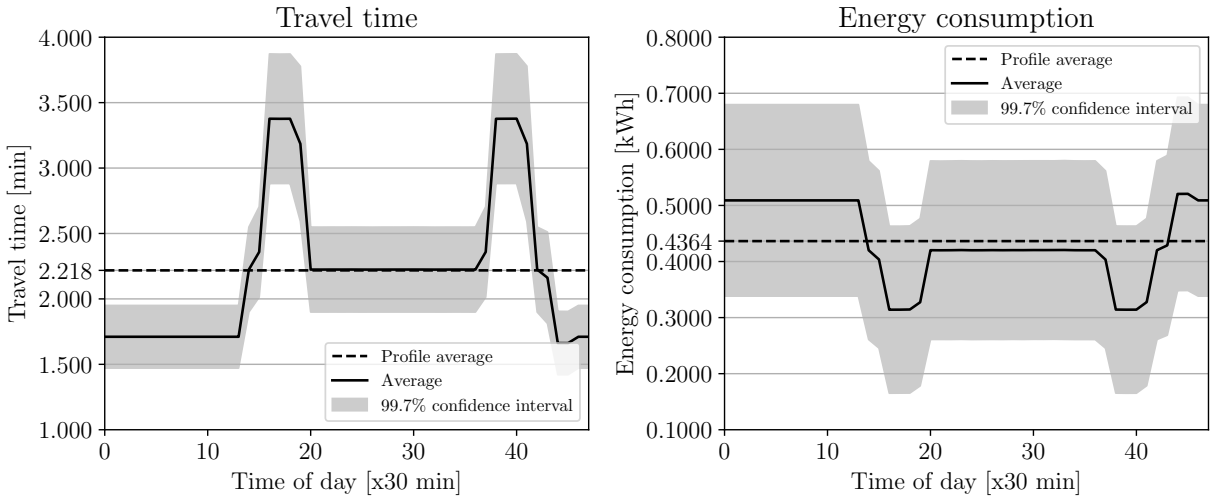


Figure 7.2: Travel time and energy consumption profiles for a 1 [km] arc length using Santiago's database

to 10 minutes, whereas requirements within 50 kg to 150 kg. In all cases, CS capacities are set to three.

Finally, we label the two instances as:

- santiago6: contains 1 depot, 4 customers, and 1 CS.
- santiago22: contains 1 depot, 20 customers, and 1 CS.

7.2 Study of the Pre-operation Stage

In this section, we study the performance of both αGA and βGA to solve the Off-E-VRP over the 22 instances described in the previous sections (20 artificial + 2 real-world). As described in Section 6.2, αGA is used first to assign the customers and calculate the first routes. Then, βGA aims to improve those routes. If the result is infeasible, the dispatcher inserts a new EV, according to Algorithm 5. Finally, the system registers the Off-E-VRP optimization results from both αGA and βGA, after the last EV insertion. The latter represents the performance of the initial routes before the EVs begin their operation. Such procedure emulates the pre-operation stage.

In Algorithm 5, we set the maximum number of EV insertions as $\delta\bar{M} = 3$. Tables 7.3 and 7.4 detail the hyper-parameters of αGA and βGA, respectively. These tables show the heuristic criterion used to tune the population size, the maximum number of generations, and the penalization constants to make them larger when the problem size increases.

Detailed results from αGA and βGA are shown in Tables 7.5 and 7.6, respectively. These results show that the proposed solution method can find feasible solutions for most instances. Feasible routes tend to exploit recharging operations to fulfill not only the SOC policy but other constraints. For example, the SOC and time EVs have when they arrive at the latter

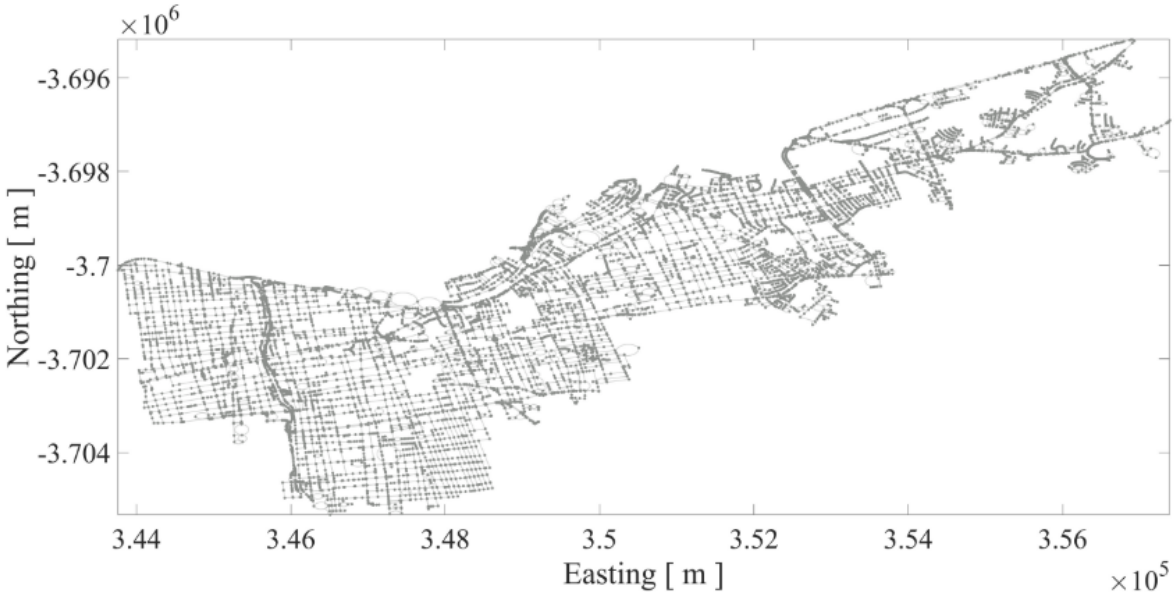


Figure 7.3: Distribution of database's nodes and arcs

Table 7.3: αGA hyper-parameters

Hyper-parameter	Description	Value
CXPB	Crossover prob.	0.7
MUTPB	Mutation prob.	0.9
μ	Population size	$1.5 \cdot (n + s + 1) + 10m + 50$
ρ	Max. generations	3μ
κ	Elite individuals	1
$\mathcal{K}_1$	Penalization constant	$10 \cdot (n + s + 1)$
Υ	Tournament size	5
n^*	Sub-bocks in the CPB-1	$2 \cdot m$

n: num. of customers, s: num of CS, m: fleet size

nodes can be altered by adjusting the recharging plan. Thus, one can consider recharging operations as time delay events that allow EVs to accomplish time windows; hence, lowering total waiting times at nodes. In practical situations, that is a crucial advantage because a proper synergy between recharging times and waiting times could allow the fleet to visit more customers.

Figure 7.4 shows the example of the operation of an EV that follows a feasible route. The route is extracted from the βGA solution of instance santiago22. In this case, EV 0 satisfies all operational constraints. Charging operations help the EV to maintain the SOC within the SOC policy. Furthermore, they also act as time delay events, which allows the EV to fulfill the time windows of the next customers. For example, the charging operation after customer 20 is enough to increase the SOC level near 95% and reach customer 17 right after its time window begins. Notice that, in this example, there is a situation where EV 0 waits right after serving customer 17. The latter implies that it is more convenient to wait there,

Table 7.4: βGA hyper-parameters

Hyper-parameter	Description	Value
CXPB	Crossover prob.	0.7
MUTPB	Mutation prob.	0.9
μ	Population size	$1.5 \cdot (n + s + 1) + 10m + 50$
ρ	Max. generations	2μ
κ	Elite individuals	1
$\mathcal{K}_1$	Penalization constant	$10 \cdot (n + s + 1)$
Υ	Tournament size	5

n: num. of customers, s: num of CS, m: fleet size

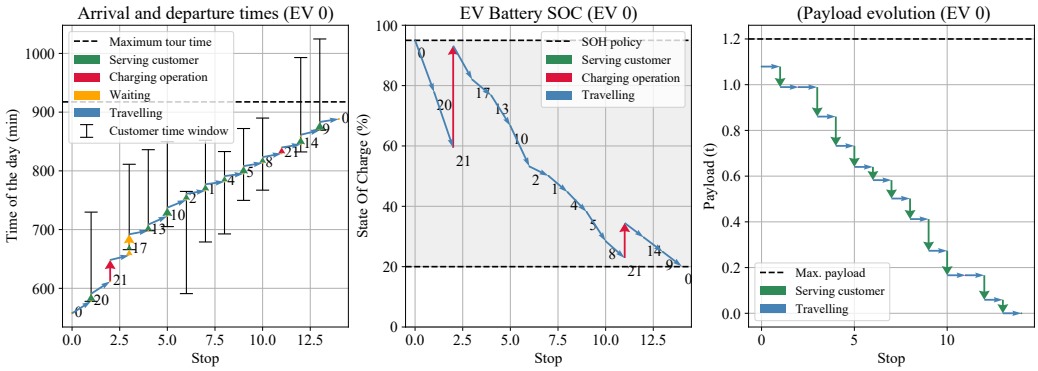


Figure 7.4: Feasible route example (from santiago22 instance)

right after serving the customer, instead of waiting before serving customer 13. This event proves that, on some occasions, it may be better to wait at the previous node because the energy consumption required to traverse the node is lower than the case where the EV waits at the next node.

To analyze the behavior of violated constraints, we visualize in Figure 7.5 the route of EV 0, extracted from one of the solutions of βGA for instance c50cs8_20x20km (not the final result). In this case, αGA yields a result where the EV must carry a mass higher than

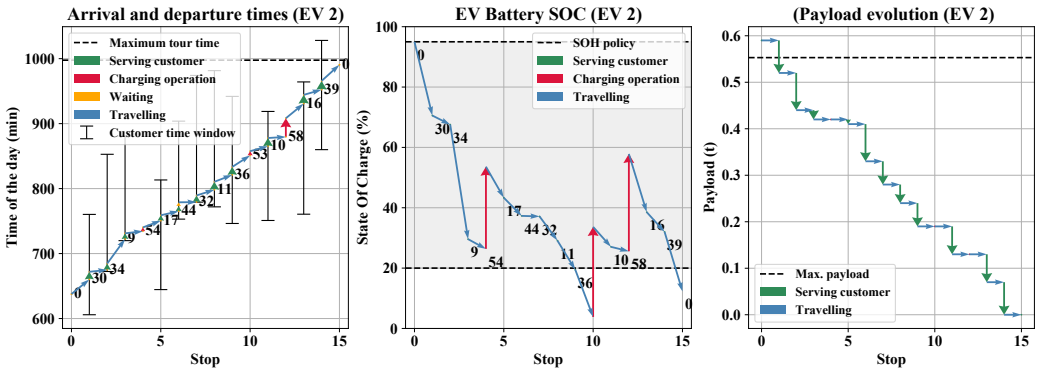


Figure 7.5: A route that breaks several constraints (from c50cs8_20x20km)

Table 7.5: Detailed results from αGA

Instance	Fitness	Distance	J1 [min]	J2 [min]	J3 [CLP$]	J4 [kWh]	Fleet size	Execution time (s)
c10cs1_10x10km	4.94	0	76.12	0	0	15.88	1	15.19
c10cs2_10x10km	4.36	0	66.78	0	0	14.15	1	16.59
c25cs3_10x10km	348.41	0	235.27	30.01	166426.49	48.8	2	67.98
c25cs4_10x10km	9.68	0	149.31	0	0	30.91	3	82.81
c50cs2_10x10km	1046.38	132.85	416.36	46.34	127945.66	88.09	5	241.94
c50cs5_10x10km	187.53	0	412.36	28.49	80228.81	85.92	5	273.95
c50cs8_10x10km	230.72	0	375.31	31.39	102952.84	79.42	4	294.17
c75cs10_10x10km	36.14	0	555.23	0	0	116.34	8	841.4
c75cs15_10x10km	242.07	0	541.89	54.6	103125.38	113.6	6	875.5
c100cs10_10x10km	153.4	0	754.25	9.03	52023.46	159.81	10	1626.8
c100cs20_10x10km	42.36	0	650.92	0	0	136.32	10	2051.61
c10cs1_20x20km	84.89	0	180.52	6.36	36594.38	36.7	2	22.61
c10cs2_20x20km	10.57	0	169.4	0	0	30.73	2	21.28
c25cs3_20x20km	712.44	74.72	350.94	38.89	122430.94	69.68	3	78.71
c25cs4_20x20km	669	0	405	60.22	321289.03	79.2	3	88.71
c50cs5_20x20km	1898.92	34.83	652.27	118.81	580094.15	139.4	5	455.78
c50cs8_20x20km	3167.75	590.05	782.66	117.15	917952.29	160.69	5	350.85
c75cs10_20x20km	2896.61	55.98	1089.29	174.18	889223.23	223.5	8	1380.16
c75cs15_20x20km	2731.17	66.57	1096.21	155.84	771198.36	222.3	9	1636.11
c100cs15_20x20km	3267.37	1218.38	1455.3	80.73	302295.83	293.13	14	2691.11
c100cs20_20x20km	4781.95	1166.64	1537.24	194.44	1067178.87	311.16	12	2481.3
santiago6	2.16	0	26.59	0	0	10.14	1	10.16
santiago22	300.06	0	233.25	75.81	141727.79	55.91	2	34.96

its limit. βGA will attempt its best to fulfill all constraints, but, as αGA gives a solution where the EV capacity is surpassed, βGA will never achieve that constraint. Notice that accomplishing the SOC policy is quite difficult as well. Here, the first customers have time windows very low compared to latter customers; thus, the EV must attempt to visit them first. However, visiting those customer draws a lot of energy from the EV battery. Therefore, the algorithm must adjust the charging plan to prevent the SOC from operating outside the SOC policy, while ensuring that the EV arrives at customers within their time windows.

The routes from the latter example are not necessarily definitive. The pre-operation stage procedure lowers the impact of violated constraints by inserting more vehicles. That reduces the number of customers each EV visits because we can now remove some customers from the previous EVs and add them to the new ones. Although the latter procedure may be beneficial, it is a drawback if the fleet size is fixed. That is, if the fleet owner cannot increase the fleet size, the best route he or she can opt for is the one described in Figure 7.5.

In practical situations, one can tolerate the violation of constraints up to a certain degree. Notice that, in the example in Figure 7.5, all customer time windows are accomplished. Besides, the tour time duration does not exceed the limit. The only violated constraints are the SOC policy and EV's weight limit. Thus, one may permit this route if the latter two constraints are not crucial. Nonetheless, the latter implies poor handling of the EV and its battery. Furthermore, the EV reaches a near-discharge state by arriving at CS 21 for

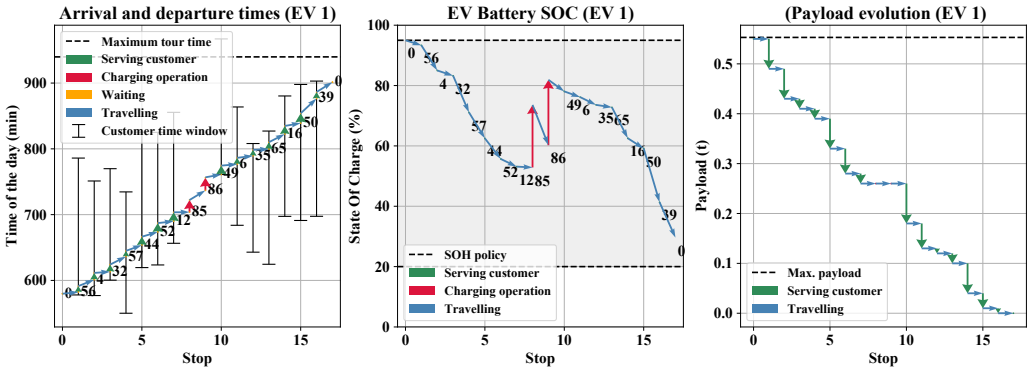
Table 7.6: Detailed results from βGA

Instance	Fitness	Distance	J1 [min]	J2 [min]	J3 [CLP$]	J4 [kWh]	Fleet size	Execution time [s]	Fitness Improv. [%]
cc10cs1_10x10km	4.94	0	76.12	0	0	15.88	1	20.25	0
c10cs2_10x10km	4.36	0	66.78	0	0	14.15	1	22.23	0
c25cs3_10x10km	348.41	0	235.27	30.01	166426.49	48.8	2	80.07	0
c25cs4_10x10km	9.68	0	149.31	0	0	30.91	3	93.02	0
c50cs2_10x10km	868.27	0.75	408.89	55.66	105144.85	86.34	5	270.37	17.02
c50cs5_10x10km	176.77	0	405.63	27.6	75069.15	84.6	5	308.34	5.73
c50cs8_10x10km	230.65	0	374.24	31.39	102952.84	79.19	4	330.87	0.03
c75cs10_10x10km	35.61	0	547.21	0	0	114.64	8	888.17	1.46
c75cs15_10x10km	252.61	0	538.17	45	108529.42	113.57	6	955.18	-4.35
c100cs10_10x10km	135.88	0	749.05	7.54	43443.58	158.69	10	1718.68	11.42
c100cs20_10x10km	42.33	0	650.58	0	0	136.16	10	2205.09	0.07
c10cs1_20x20km	84.89	0	180.52	6.36	36594.38	36.7	2	24.61	0
c10cs2_20x20km	10.57	0	169.4	0	0	30.73	2	24.59	0
c25cs3_20x20km	712.88	78.63	350.93	38.59	120697.55	69.68	3	85.38	-0.06
c25cs4_20x20km	604.7	0	404.35	54.64	289184.84	79.19	3	93.37	9.61
c50cs5_20x20km	1807.08	33	647.83	121.44	535234.95	138.13	5	304.94	4.83
c50cs8_20x20km	2770.46	220.08	781.18	139.44	889237.3	160.25	5	317.65	12.54
c75cs10_20x20km	2792.59	44.42	1078.07	163.22	838388.57	221.62	8	843.48	3.59
c75cs15_20x20km	2494.26	29.64	1080.2	170.91	646640.32	219.26	9	938.17	8.67
c100cs15_20x20km	3090.71	1058.7	1453.4	83.45	303833.47	293.09	14	2442.72	5.4
c100cs20_20x20km	4311.26	845.95	1539.93	215.63	1042009.06	311.22	12	2486.22	9.84
santiago6	2.16	0	26.59	0	0	10.14	1	11.65	0
santiago22	297.12	0	232.27	79.35	140255.14	56.02	2	35.78	0.97

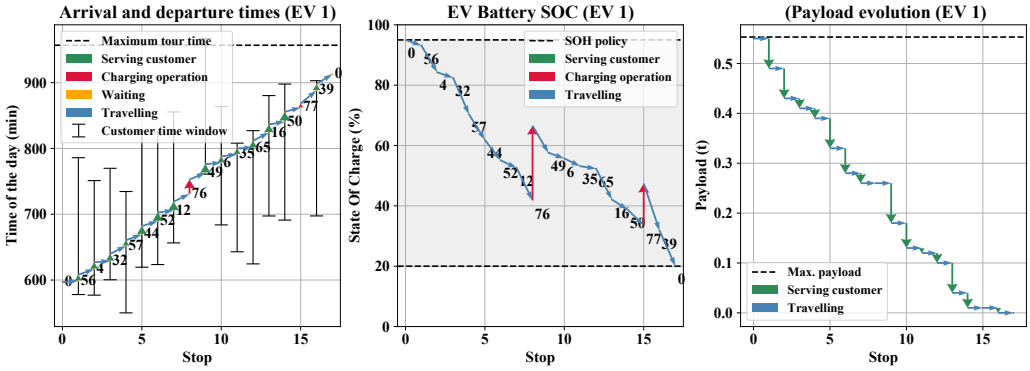
the second time; this can be very risky as the EV battery may discharge if the SOC is not well estimated or degrade considerably. In reality, one should study the trade-off between allowing such types of routes to be accepted or inserting a new EV into the fleet.

In Table 7.6, some βGA results have a negative fitness improvement. That is, βGA does not improve the solution at all. In all those cases, a worse result occurs because the best solution from αGA contains a charging path with more than one CS between two customers. For example, Figure 7.6a shows a case from instance c75_cs15_10kmx10km where EV 1 visits CS 85 and CS 86 right after serving customer 12. On the other hand, the representation in βGA only allows charging paths with a single CS between two customers to be inserted. As a result, the initial solution (constructed from the αGA result) is likely to be worse in terms of fitness. After some generations, βGA can find individuals with greater fitness, but not as good as the ones found by αGA. That is the case of the operation in Figure 7.6b, where EV 1 now only visits CS 76 right after customer 12.

The above behavior usually occurs when an EV visits a customer way far from the last visited customer, and the battery still has a high SOC level. In such a case, the EV may pass two CS with different technologies. The synergy between their prices and the charging time the EV spends in them may be better than when the EV only visits a single CS in two different charging paths. Such results are similar to the ones discussed by [24], where the authors find that allowing multiple visits to CS between non-CS nodes can improve the results of [7] for the E-VRP-NL. We remark that such behavior is typical but not always



(a) EV 1 operation using αGA results (Fitness: 2239.1)



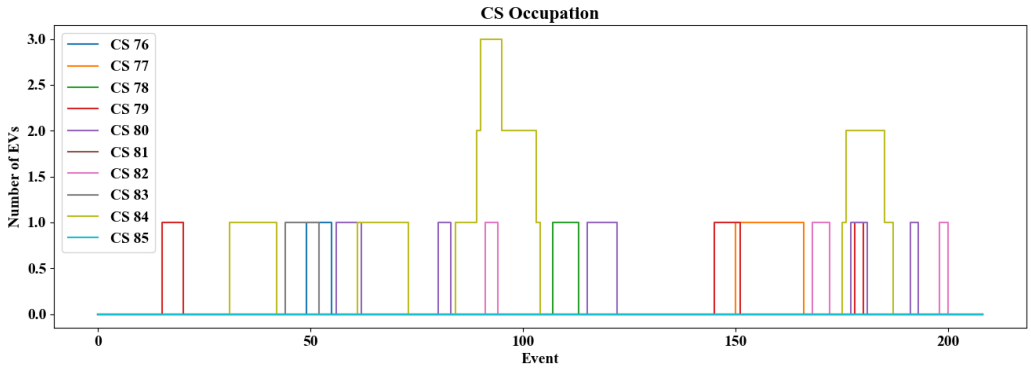
(b) EV 1 operation using βGA results (Fitness: 2296.59)

Figure 7.6: A case comparison where βGA gives a worse results than αGA (from instance c75_cs15_10kmx10km)

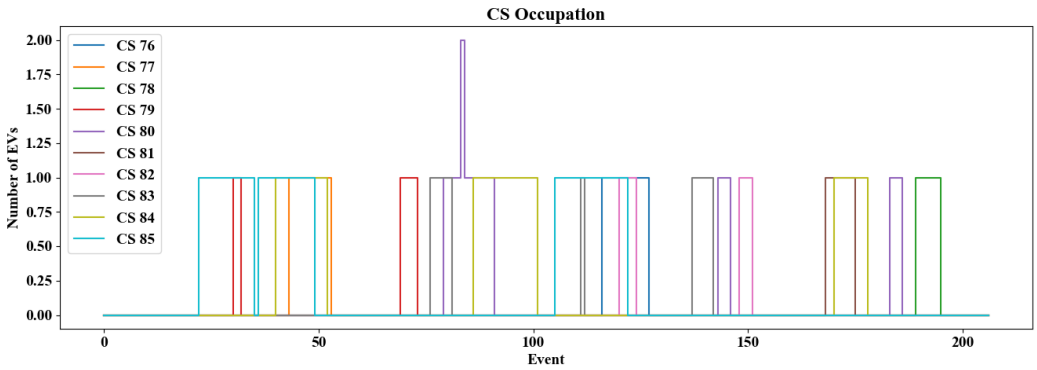
right. With the latter statement, we promote the usage of βGA to explore different kinds of solutions by using βGA encoding. In fact, βGA improves the fitness of αGA results in 70% of our instances.

For the 100$[km^2]$ instances, nearly 54% of the solutions require charging operations, whereas, for the 400$[km^2]$ instances, that number rises up to 90%. The latter occurs because, in the 400$[km^2]$ instances, the distance EVs must travel between nodes is higher, which leads to more energy consumption. As a result, EVs require recharging more frequently. The latter indirectly increases the problem's complexity because routes length tends to grow as more charging operations are required. That explains why it is more difficult for GAs to solve the 400$[km^2]$ instances fulfilling all operational constraints, especially the SOC policy.

Regarding CS capacities, Figure 7.7a shows a result from instance c75cs10_20x20km where CS 84 reaches its maximum capacity by recharging three EVs at the same time. We test the formulation and solution approach's ability to adjust the routes when CS capacities are lower. We re-run the optimization procedure again but with a maximum CS capacity of two. The new result is shown in Figure 7.7b. There, we highlight that none of the CS capacities are surpassed. Only CS 80 reaches a maximum capacity of two. Such a result verifies that, if the recharging infrastructure is not enough, the formulation should account for CS capacities.



(a) Max. CS capacity of 3



(b) Max. CS capacity of 2

Figure 7.7: Effect of varying maximum CS capacities in instance c75cs10_20x20km. The optimization procedure is capable of adjusting the routes so that CS capacities are never exceeded

The placement of CS is also very important. There are cases when some CSs are never visited. After analyzing several cases, we notice that there are two main scenarios that could lead to avoiding visiting a CS:

1. it is close to another CS that is cheaper,

2. it is too far away from all nodes.

We highlight instance c25_cs4_20x20km, where CS 26 is very close to CS 27, as shown in Figure 7.8. CS 27 implements fast technology whereas CS 26 normal technology. However, as the GAs find a solution using CSs with normal technology only, EVs never visit CS 27.

Regarding the computational demand of GAs, we highlight that executions times are very short compared to other GAs in the literature. For the 100 customer instances, our solution method takes around 40 minutes to run. According to the E-VRP survey in [23], the GA method studied by [53] reports execution times up to 10 minutes for 100-customer instances, which is considered among the shortest ones. However, their implementation neglects most of the realistic components addressed in this work. A fair comparable case is presented by [43], where they also use a GA to solve the problem considering several realistic elements. However,

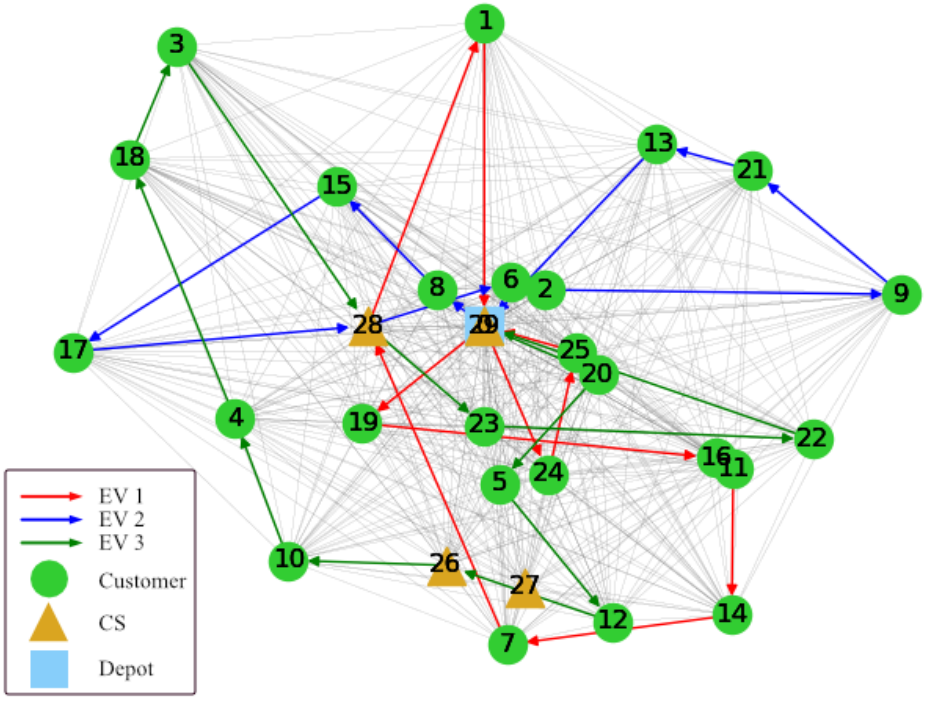


Figure 7.8: Routes of EVs from instance c25_cs4_20x20km. Here, no EV visits CS 27 although it implements fast technology

their implementation reports execution times up to 3 hours for 50-customer instances.

7.3 Study of the Online Stage

In this section, we study the performance of onGA to solve the On-E-VRP, using instance santiago22. We emulate the operational behavior of EVs using the simulation framework described in Section 6.3.2, considering the initial routes from the pre-operational stage and updating the routes with OnGA. As a result, this procedure allows us to test the online stage of the system. Each EV begins with the routes obtained from the pre-operation stage. Right after the first EV starts moving, the online stage begins, and the system initializes OnGA.

We run the simulation procedure 50 times to obtain realistic representations of the operation. Following the methodology from Section 6.3, we consider $t_{GA} = 4[min]$ and $t_{safe} = 0.5[min]$. OnGA runs using the hyper-parameters detailed in Table 7.7. For each simulation, we record the variable detailed in Table 7.8.

Figure 7.9 summarizes with boxplots the real costs obtained after simulating the online stage 50 times. The median of the online method's total cost is lower than the offline method. The latter implies that, more than 50% of times, updating the routes by solving the On-E-VRP leads to cost improvements. If we consider up to 75% of the data, we notice that these values gather around a similar cost obtained by the Off-E-VRP. The rest of the data indicates that solving the On-E-VRP leads to increasing the total cost in a less number of cases. We also plot boxplots for each cost. Notice that there is a significant reduction in

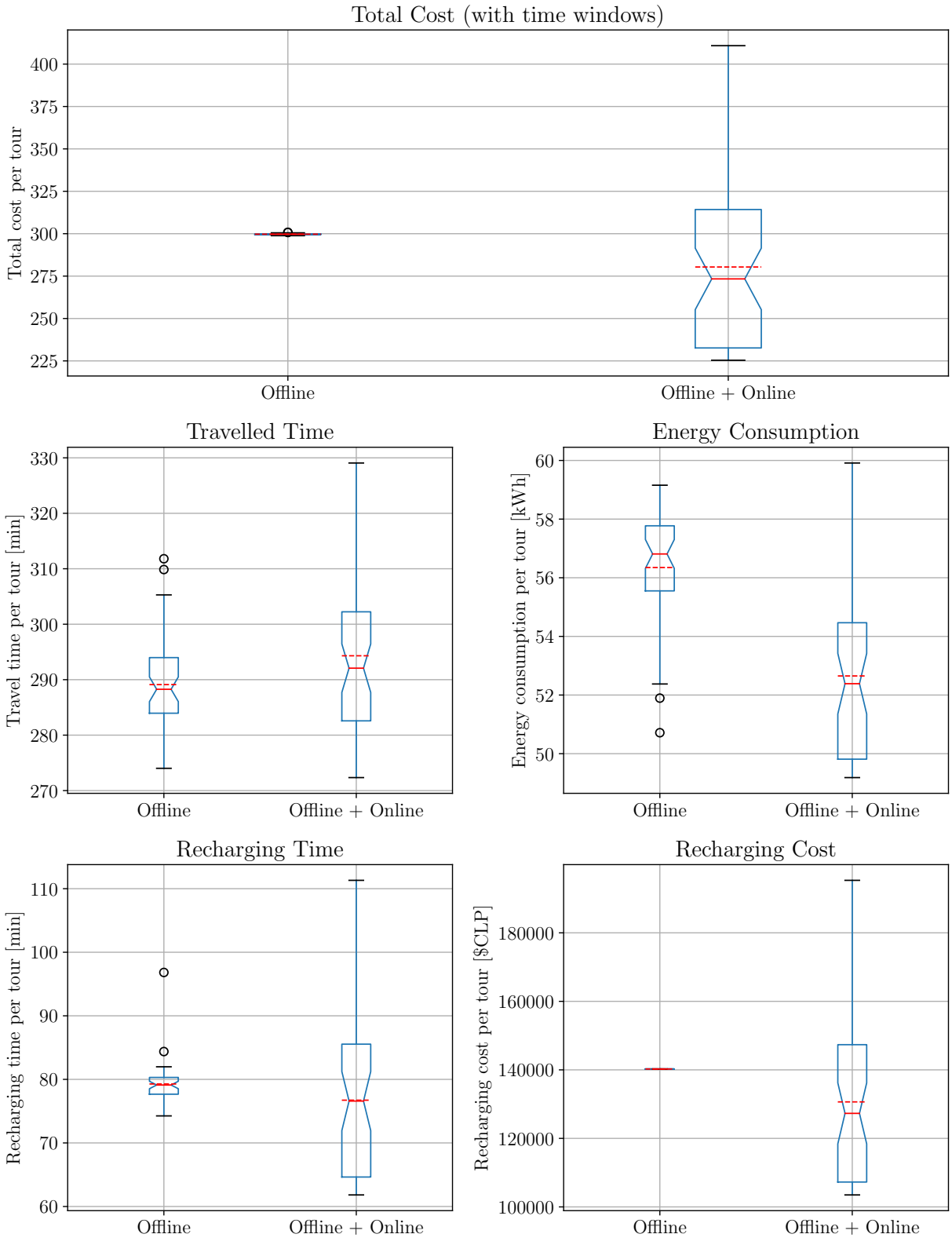
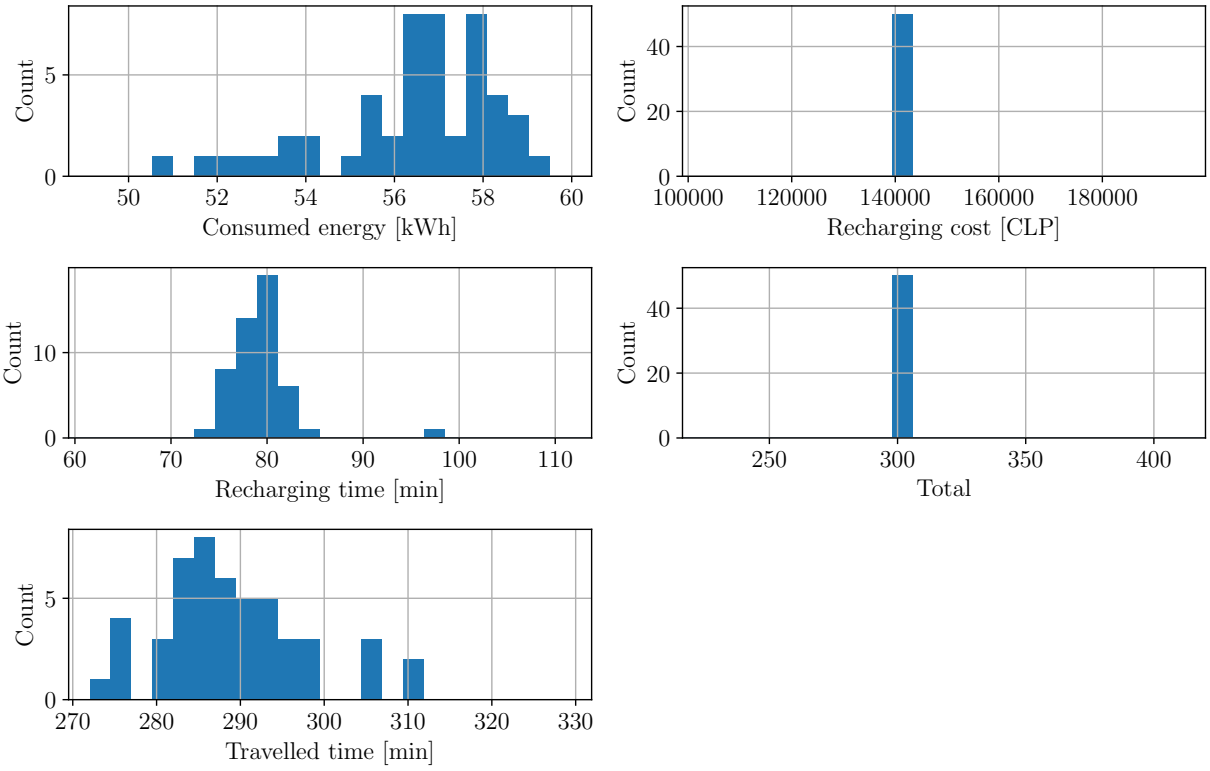
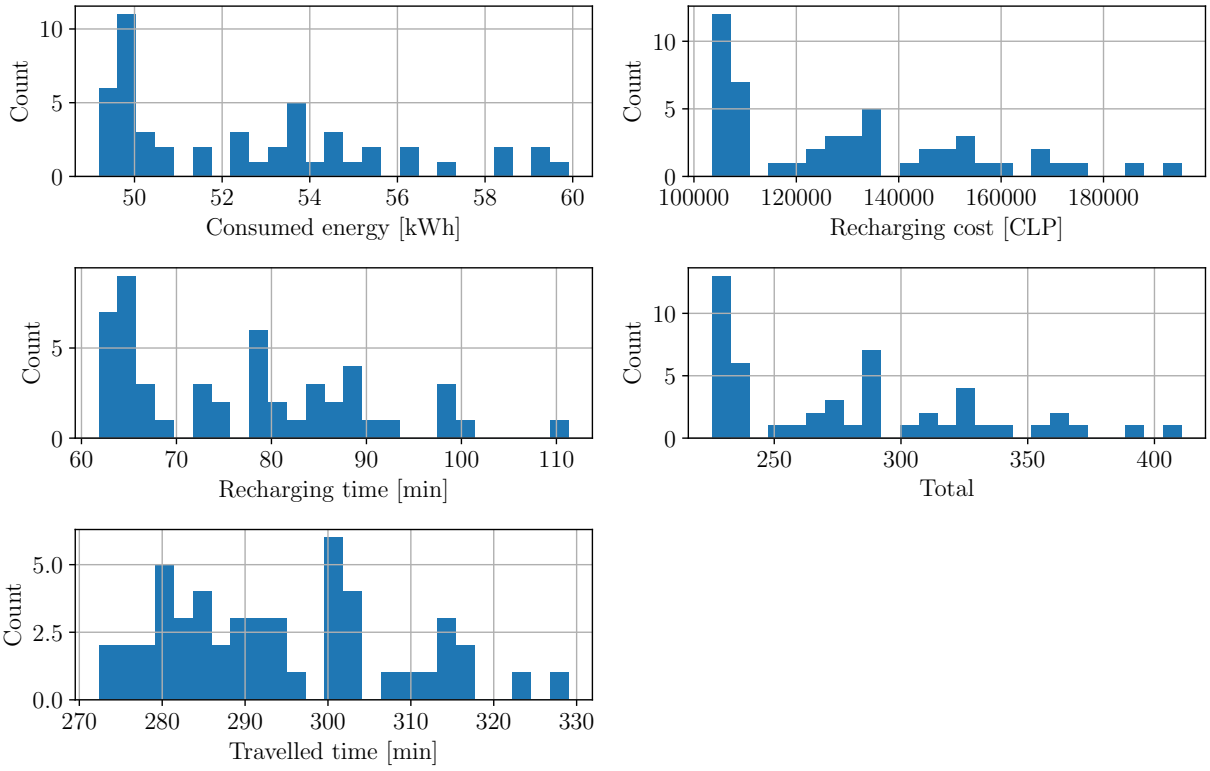


Figure 7.9: Boxplot comparison of real costs obtained after 50 simulations using both offline and online strategies. The red dashed line indicates the average, whereas the red continuous line the median.



(a) Offline



(b) Offline + Offline

Figure 7.10: Visualization of costs histograms after 50 simulations using both methodologies

energy consumption and smaller reductions in recharging times and recharging costs using the online methodology. On the other hand, travel times tend to increase using the online methodology.

We analyze the dispersion of costs using the online strategy by visualizing the cost histograms in Figure 7.10. In the offline case, we notice that, in Figure 7.10.a, costs tend to group around specific values. This result is expected as routes are fixed, and the real costs only depend on traffic realizations, which vary around the average values considered in the Off-E-VRP. In the online case, the consumed energy, recharging cost, and recharging time tend to group in lower values, as shown in Figure 7.10.b. Such behavior occurs because of the online optimization strategy. The travel time remains similar, as we did not give as much importance as the other costs. However, we also observe that values tend to group in smaller clusters in the online cost histograms. That occurs because the routes are updated according to traffic realizations and, due to the combinatorial nature of the problem, the optimizer will change the route to the best one according to that traffic realization. These updates mostly consists of swapping customers and altering recharging operations. Hence, we expect to obtain cost clusters around the costs of the bests routes in terms of fitness.

As mentioned in the previous section, αGA and βGA tend to adjust the recharging plan to make EVs accomplish customer time windows while ensuring they also accomplish the SOC policy. Nonetheless, there are cases where EVs reach customers just below their upper time window bounds. For example, in Figure 7.4, EV 0 leaves customer 2 just before the time window finishes. Besides, EVs tend to finish the operation or make mid-tour visits to CS when their battery has a SOC close to the SOC policy's lower bound. As a result, small changes in travel times and energy consumption can make EVs violate such constraints.

The latter implies that analyzing the cost is not enough to conclude about the On-E-VRP formulation and OnGA's performance. That is, we must compare constraint violations by using the offline and the online methodology. Table 7.9 summarizes the number and degree of constraint violations using both methodologies throughout the 50 simulations. Notice that the offline methodology violates a significant amount of constraints in comparison to the online methodology. Furthermore, the maximum degree of constraint violations is, in all cases, lower when using the online methodology. These results show that properly solving the

Table 7.7: OnGA hyper-parameters

Hyper-parameter	Description	Value
CXPB	Crossover prob.	0.6
MUTPB	Mutation prob.	0.9
μ	Population size	80
ρ	Max. generations	160
κ	Elite individuals	1
$\mathcal{K}_1$	Penalization constant	100000
$\mathcal{K}_2$	Penalization constant	200000
Υ	Tournament size	3
n^*	Sub-bocks in the CPB-1	2

Table 7.8: Variables the system records per simulation

Level	Variable	Notes
Fleet variables	Total travel time	[min]
	Total recharging time	[min]
	Total recharging cost	[$CLP]
	Total consumed energy	[kWh]
	Violated constraints	CS capacities
Single EV variables	Total travel time	[min]
	Total recharging time	[min]
	Total recharging cost	[$CLP]
	Total consumed energy	[kWh]
	Violated constraints	Time windows, SOC policy, max. tour time
Optimization variables	Execution time	[s]
	Real routes	

On-E-VRP leads to route adjustments that prevent EVs from breaking a significant amount of constraints and, if breaking any, the degree of violation is small.

The two most non-accomplished constraints are the upper time windows and the lower SOC policy. Those are the most challenging constraints to achieve because we cannot develop policies to prevent EVs from violating them. They purely depend on the dynamics of the EV and traffic realizations. Even though OnGA can predict that some constraints will not be accomplished and recalculate the routes, it is impossible to know the traffic state a-priori. The latter is the main reason why EVs still violate some constraints using the online method.

As OnGA recalculate routes attempting to prevent EVs from violating constraints under new traffic realizations, routes may change considerably. That is, the customer sequence, recharging plan, and departure times from nodes are significantly altered. Such changes explain results from Figure 7.9, where there are some cases where the operational cost is more significant using the online methodology. In next Section 7.3.1, we verify the latter assumption by studying the case without time windows. We obtain that the cost is considerably lower when time windows are not considered.

We remark that OnGA reacts to traffic realizations. Therefore, the routes it calculates are only optimal for that scenario. In our simulation framework, traffic realizations are independent of previous traffic realizations. However, in reality, traffic evolves following a stochastic and dynamic process. Furthermore, sudden traffic changes do not necessarily involve more congestion because, in some cases, congestion may be less. A proper estimation of future traffic congestion based on the current traffic state may considerably improve OnGA performance as new routes would consider such states. The latter is an intricate problem that is far from this thesis objectives.

Table 7.9: Summary of constraint violations from simulations of both offline and online strategies

	Offline					Offline + Online				
Constraint	Count	Average violation	Std. violation	Min. violation	Max. violation	Count	Average violation	Std. violation	Min. violation	Max. violation
Lower TW	0	0	0	0	0	0	0	0	0	0
UpperTW	54	2.16	1.49	0.4	10.35	8	1.12	0.64	0.17	2.15
Upper SOC policy	2	1.65	1.21	0.43	2.86	0	0	0	0	0
Lower SOC policy	111	3.5	2.75	0.21	11.72	6	3.31	2.98	0.02	8.97
Max. tour time	0	0	0	0	0	0	0	0	0	0
CS capacity	0	0	0	0	0	0	0	0	0	0
Total	167					14				

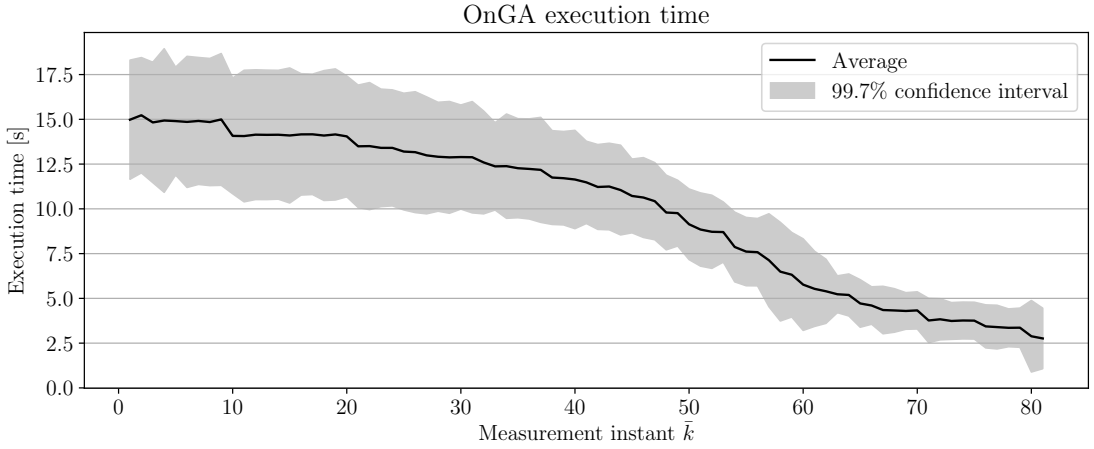


Figure 7.11: Average execution time of OnGA throughout the operation

Regarding OnGA's computational effort, we register the number of OnGA executions and their execution time. The number of OnGA executions oscillates between 81 and 98 times per simulation. Figure 7.11 shows OnGA execution time versus the measurement instant, until execution 81 (the minimum OnGA execution number registered). The average execution time decreases because the number of customers assigned to each EV diminishes as the operation goes by. The latter is equivalent to reducing the problem's complexity. Still, OnGA execution time never surpasses t_{GA}, which has been set to 4 minutes.

7.3.1 Effect of time windows in the final cost

In this section, we conduct an experiment to study the influence of time windows in the operational cost. As reviewed in Chapter 2, most E-VRP variants do not consider time windows. However, they usually tend to guide the route of vehicles. That occurs because certain customers must be visited at certain hours, making it impossible for EVs to visit them, even though it is convenient in terms of energy consumption.

As we have already analyzed the results considering time windows (in the previous section), we now study the case without them. That is, customers can be visited at any time of the

day. In this case, 50 simulations are run considering no time windows and the same hyper-parameters from the original case.

Figure 7.12 shows the comparison of boxplots of costs after running the simulations. In this case, all costs experiment a significant reduction, including the offline case. The latter occurs because, in this particular case, EVs goes through scenarios where costs are lower in average when they do not wait. From the individual costs, we notice that there is an outlier case in the recharging cost of the offline operation. The latter occurs because one EV tries to recharge above 100%. However, the CS limits that amount and the EV ends up recharging less that the original value.

Table 7.10 summarizes the number and degree of constraint violations. Under this scenario, both offline and online cases violate less constraints than the original case. In fact, the only constraint violations come from the SOC policy. For the online case, we also notice that the SOC policy is accomplished more times than the original case. These results show that, again, using an online methodology leads to accomplishing more constraints.

The latter results allows us to conclude that adding more constraints into the E-VRP will result in increasing costs. In particular, time windows positions will force EVs to visit them at certain orders. Therefore, we cannot allow routes that are time or energy efficient if they do not accomplish some constraints.

Table 7.10: Summary of constraint violations from simulations of both offline and online strategies, dropping time windows

	Offline					Online				
Constraint	Count	Average violation	Std. violation	Min. violation	Max. violation	Count	Average violation	Std. violation	Min. violation	Max. violation
Lower TW	0	0	0	0	0	0	0	0	0	0
UpperTW	0	0	0	0	0	0	0	0	0	0
Upper SOC policy	15	1.91	1.43	0.09	5	0	0	0	0	0
Lower SOC policy	107	4.68	3.56	0	18	3	0.33	0.21	0.16	0.63
Max. tour time	0	0	0	0	0	0	0	0	0	0
CS capacity	0	0	0	0	0	0	0	0	0	0
Total	122					3				

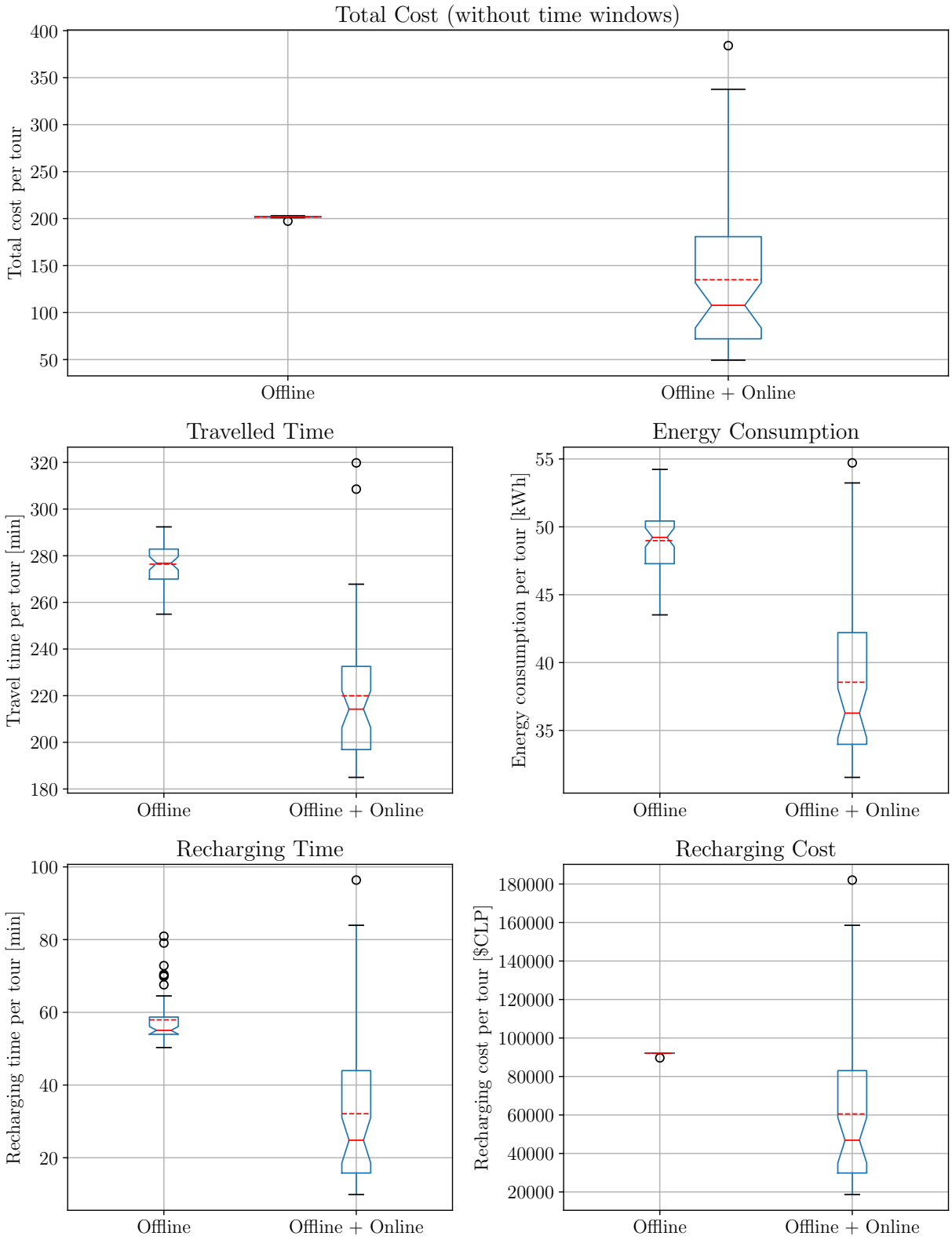


Figure 7.12: Boxplot comparison of real costs obtained after 50 simulation using both offline and online strategies, dropping time windows. The red dashed line indicates the average, whereas the red continuous line the median.

Chapter 8

Conclusions and Future Work

In this thesis work, a strategy to manage EV fleets for delivery purposes efficiently has been developed. The strategy consists of splitting the whole operation into two stages: pre-operation and online operation. In the former, the system calculates the initial routes of the EVs by solving the Off-E-VRP, whereas, in the second one, the system receives measurements and updates the routes by solving the On-E-VRP. The measurements contain information about the state of EVs and the traffic network. As a result, the second stage provides a closed-loop decision-making method to manage the EV fleet. In the following paragraphs, we review the fulfillment of specific objectives, which lead to achieving the main objective.

The Off-E-VRP is a new variant of the E-VRP that considers capacitated CS, a realistic EV energy consumption model, time-dependent travel times, and a SOC policy that ensures a longer battery lifespan. The problem is modeled as a nonlinear program, which formally describes the optimization problem to solve. Solving the Off-E-VRP allows the fleet manager to fulfill the delivery requirements, as it includes all operational constraints into the problem statement.

To solve the Off-E-VRP, we develop two GAs: αGA and βGA. Both GAs implement a novel encoding that allows them to search for optimal node sequences, recharging plans, and departure times. αGA uses an encoding that enables it to assign the customers each EV will visit. On the other hand, βGA only works when customers are already assigned. The pre-operation stage consists of solving the Off-E-VRP with αGA first, and then using βGA to find new solutions that may be better. Such a procedure defines the initial routes each EV will follow.

The On-E-VRP is a new E-VRP variant that aims to update the routes according to traffic network and EV state measurements. These route updates should allow EVs to fulfill the operation with the same constraints as the Off-E-VRP. As a result, solving the On-E-VRP allows the central dispatcher to act as a closed-loop management system of the EV fleet. To solve the On-E-VRP, we develop OnGA. This GA uses an encoding similar to αGA and βGA, which allows the algorithm to update the current routes and recharging plan.

Two case studies to test the performance of the system are conducted. The first study examines the pre-operation stage by solving several instances of the Off-E-VRP with αGA and then improving the solutions with βGA. Results show that the strategy can find feasible solutions for most instances. The latter means that it is expected that EVs will achieve all operational constraints before they begin the operation. Such operational constraints include customer time windows, maximum tour time duration, weight limit, SOC policy, and capacitated CS. The SOC policy will prevent the EV battery from degrading faster, whereas

addressing capacitated CS will prevent too many EVs from visiting CS simultaneously and exceeding their capacity limit. The solution method can also deal with several real-world elements such as time-dependent travel times, realistic EV energy consumption models, and nonlinear charging functions. The latter allows the system to accurately estimate the time an EV will spend at a CS, which is crucial as charging operations can take plenty of time.

In the second study, we analyze the online stage by emulating a traffic network with data collected from one of Santiago de Chile's most congested areas. We consider the routes found in the pre-operation stage as the initial routes of EVs. Those routes are then updated by solving the On-E-VRP with OnGA according to measurements collected by the simulation. Results show that updating routes can lower operational costs in more than 50% of cases. Furthermore, continuously solving the On-E-VRP allows EVs to considerably accomplish more operational constraints than by just following initial routes. The latter implies a better service quality, as fleet owner can now provide service guarantees for customers, and a good handling of EVs and their components.

From the results obtained in this work, it has become apparent that the following research topics are interesting. αGA, βGA, and OnGA can be improved by modifying genetic operations and the encoding. For genetic operations, one can develop more strategies to explore charging operations. For the encoding, one can investigate using customer blocks that do not need delimiters and charging operations where charging paths were previously calculated. The latter refers to a similar methodology developed in [24], where the authors calculate charging paths before optimizing; thus, the search space is reduced. This method may allow GAs to only focus on when and how much to recharge instead of when, how much and where.

Another remark is on the simulation framework. Although this thesis work develops a simulation environment to test OnGA, several real-life events are not properly modeled. The latter is because we consider independent realizations of the traffic, which implies that the current state of traffic does not depend on the previous states. We, therefore, encourage further research on properly simulating EVs under stochastic and dynamic traffic environments. Finally, we remark that, as stochasticity exists in this problem, one can address a stochastic version of the Off-E-VRP and the On-E-VRP. By addressing an stochastic Off-E-VRP, one can ensure that the assignation of customers, initial routes, and departure times are such that constraints are accomplished with more confidence. By addressing an stochastic On-E-VRP, one can benefit from route updates while ensuring constraints' fulfillment with certain degree of confidence.

Bibliography

[1] L. L. P. de Souza, E. E. S. Lora, J. C. E. Palacio, M. H. Rocha, M. L. G. Renó, and O. J. Venturini, "Comparative environmental life cycle assessment of conventional vehicles with different fuel options, plug-in hybrid and electric vehicles for a sustainable transportation system in Brazil," *J. Clean. Prod.*, vol. 203, pp. 444–468, 2018. DOI: 10.1016/j.jclepro.2018.08.236.

[2] A. A. Juan, C. A. Mendez, J. Faulin, J. de Armas, and S. E. Grasman, "Electric Vehicles in Logistics and Transportation : A," *Energies*, vol. 9, no. February, p. 86, 2016. DOI: 10.3390/en9020086.

[3] S. Pelletier, O. Jabali, G. Laporte, and M. Veneroni, "Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models," *Transp. Res. Part B Methodol.*, vol. 103, pp. 158–187, 2017. DOI: 10.1016/j.trb.2017.01.020.

[4] E. den Boer, S. Aarnink, F. Kleiner, and J. Pagenkopf, "Zero emissions trucks: An overview of state-of-the-art technologies and their potential," CE Delft, Tech. Rep. July, 2013, p. 151.

[5] W. Feng and M. Figliozzi, "An economic and technological analysis of the key factors affecting the competitiveness of electric commercial vehicles: A case study from the USA market," *Transp. Res. Part C Emerg. Technol.*, vol. 26, pp. 135–145, 2013. DOI: 10.1016/j.trc.2012.06.007.

[6] M. Schiffer, G. Walther, and S. Stütz, "Are ECVs breaking even? : Competitiveness of electric commercial vehicles in retail logistics," 2016.

[7] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas, "The electric vehicle routing problem with nonlinear charging function," *Transp. Res. Part B Methodol.*, vol. 103, pp. 87–110, 2017. DOI: 10.1016/j.trb.2017.02.004.

[8] J.-A. Montoya, "Electric Vehicle Routing Problems : models and solution approaches," PhD thesis, Université Angers, Dec. 2016.

[9] P. Toth and D. Vigo, *Vehicle routing: problems, methods, and applications*, 2nd ed. Society for Industrial and Applied Mathematics, 2014.

[10] S. Irnich, P. Toth, and D. Vigo, "Chapter 1: The Family of Vehicle Routing Problems," in, Nov. 2014, pp. 1–33. DOI: 10.1137/1.9781611973594.ch1.

[11] G. Desaulniers, J. Desrosiers, and S. Spoorendonk, "The Vehicle Routing Problem with Time Windows: State-of-the-Art Exact Solution Methods",," *Wiley Encycl. Opera- tions Res. Manag. Sci.*, vol. 8, Feb. 2011. DOI: 10.1002/9780470400531.eorms1034.

[12] R. Baldacci, A. Mingozzi, and R. Roberti, "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints," *Eur. J. Oper. Res.*, vol. 218, no. 1, pp. 1–6, 2012. DOI: 10.1016/j.ejor.2011.07.037.

[13] Y. Y. Tseng, W. Yue, and M. Taylor, "The role of transportation in logistics chain," *Proc. East. Asia Soc. Transp. Stud.*, vol. 5, pp. 1657–1672, Jan. 2005.

[14] J. Lenstra and A. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, pp. 221–227, Oct. 1981. DOI: 10.1002/net.3230110211.

[15] D. D. K. B. M. P. (D. T. Pham BE PhD, *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, 1st ed. Springer-Verlag London, 2000.

[16] P. S. E. T. A. C. Johann Dréo Alain Pétrowski, *Metaheuristics for Hard Optimization: Methods and Case Studies*, 1st ed. Springer, 2005.

[17] D. Goldberg, *The design of innovation: lessons from and for competent genetic algorithms*, 1st ed., ser. Genetic Algorithms and Evolutionary Computation. Springer, 2002.

[18] M. Gendreau and J.-Y. Potvin, "Tabu Search BT - Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques," in, E. K. Burke and G. Kendall, Eds., Boston, MA: Springer US, 2005, pp. 165–186. DOI: 10.1007/0-387-28356-0_6.

[19] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *Eur. J. Oper. Res.*, vol. 130, no. 3, pp. 449–467, 2001. DOI: https://doi.org/10.1016/S0377-2217(00)00100-4.

[20] E. Aarts, J. Korst, and W. Michiels, "Simulated Annealing BT - Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques," in, E. K. Burke and G. Kendall, Eds., Boston, MA: Springer US, 2005, pp. 187–210. DOI: 10.1007/0-387-28356-0_7.

[21] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, 2006. DOI: 10.1109/MCI.2006.329691.

[22] M. Schneider, A. Stenger, and D. Goeke, "The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations," *Transp. Sci.*, vol. 48, no. 4, pp. 500–520, 2014. DOI: 10.1287/trsc.2013.0490.

[23] T. Erdelic and T. Caric, "A Survey on the Electric Vehicle Routing Problem : Variants and Solution Approaches," *J. Adv. Transp.*, vol. 2019, pp. 1–48, 2019.

[24] A. Froger, J. E. Mendoza, O. Jabali, and G. Laporte, "Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions," *Comput. Oper. Res.*, vol. 104, pp. 256–294, 2019. DOI: 10.1016/j.cor.2018.12.013.

[25] ——, "A Matheuristic for the Electric Vehicle Routing Problem with Capacitated Charging Stations," Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport (CIRRELT), Research Report Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRET), Jun. 2017.

[26] U. Ritzinger, J. Puchinger, and R. F. Hartl, "A survey on dynamic and stochastic vehicle routing problems," *Int. J. Prod. Res.*, vol. 54, no. 1, pp. 215–231, Jan. 2016. DOI: 10.1080/00207543.2015.1043403.

[27] D. Goeke and M. Schneider, "Routing a mixed fleet of electric and conventional vehicles," *Eur. J. Oper. Res.*, vol. 245, no. 1, pp. 81–99, 2015. DOI: 10.1016/j.ejor.2015.01.049.

[28] X. Zuo, Y. Xiao, M. You, I. Kaku, and Y. Xu, "A new formulation of the electric vehicle routing problem with time windows considering concave nonlinear charging function," *J. Clean. Prod.*, vol. 236, p. 117 687, 2019. DOI: 10.1016/j.jclepro.2019.117687.

[29] S. Pelletier, O. Jabali, and G. Laporte, "50th Anniversary Invited Article—Goods Distribution with Electric Vehicles: Review and Research Perspectives," *Transp. Sci.*, vol. 50, no. 1, pp. 3–22, Feb. 2016. DOI: 10.1287/trsc.2015.0646.

[30] M. A. Hannan, M. M. Hoque, A. Hussain, Y. Yusof, and P. J. Ker, "State-of-the-Art and Energy Management System of Lithium-Ion Batteries in Electric Vehicle Applications: Issues and Recommendations," *IEEE Access*, vol. 6, pp. 19 362–19 378, 2018. DOI: 10.1109/ACCESS.2018.2817655.

[31] D. A. Pola, H. F. Navarrete, M. E. Orchard, R. S. Rabié, M. A. Cerda, B. E. Olivares, J. F. Silva, P. A. Espinoza, and A. Pérez, "Particle-filtering-based discharge time prognosis for lithium-ion batteries with a statistical characterization of use profiles," *IEEE Trans. Reliab.*, vol. 64, no. 2, pp. 710–720, 2015. DOI: 10.1109/TR.2014.2385069.

[32] A. Perez, R. Moreno, R. Moreira, M. Orchard, and G. Strbac, "Effect of Battery Degradation on Multi-Service Portfolios of Energy Storage," *IEEE Trans. Sustain. Energy*, vol. 7, no. 4, pp. 1718–1729, 2016. DOI: 10.1109/TSTE.2016.2589943.

[33] H. Rozas, "Prognostic Based Real-time Decision Making Approach for the Dynamic and Stochastic Shortest Path Problem for Electric Vehicles," PhD thesis, University of Chile, 2019.

[34] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*. USA: Prentice-Hall, Inc., 1974.

[35] K. Braekers, K. Ramaekers, and I. V. Nieuwenhuyse, "The vehicle routing problem : State of the art classification and review," *Comput. Ind. Eng.*, vol. 99, pp. 300–313, 2016. DOI: 10.1016/j.cie.2015.12.007.

[36] G. Laporte, "Fifty Years of Vehicle Routing," *Transp. Sci.*, vol. 43, no. 4, pp. 408–416, 2009. DOI: 10.1287/trsc.1090.0301.

[37] B. Eksioglu, A. V. Vural, and A. Reisman, "The vehicle routing problem: A taxonomic review," *Comput. Ind. Eng.*, vol. 57, no. 4, pp. 1472–1483, 2009. DOI: 10.1016/j.cie.2009.05.009. arXiv: arXiv:1011.1669v3.

[38] G. B. Dantzig and J. H. Ramser, "The Truck Dispatching Problem," *Manage. Sci.*, vol. 6, no. 1, pp. 80–91, 1959. DOI: 10.1287/mnsc.6.1.80.

[39] M. Gendreau, J.-Y. Potvin, O. Bräumlaysy, G. Hasle, and A. Løkketangen, "Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography," in *Veh. Routing Probl. Latest Adv. New Challenges*. Boston, MA: Springer US, 2008, pp. 143–169. DOI: 10.1007/978-0-387-77778-8_7.

[40] R. Conrad and M. Figliozzi, "The Recharging Vehicle Routing Problem," *Proc. 61st Annu. IIE Conf.*, Jan. 2011.

[41] S. Erdoğan and E. Miller-Hooks, "A Green Vehicle Routing Problem," *Transp. Res. Part E Logist. Transp. Rev.*, vol. 48, no. 1, pp. 100–114, 2012. DOI: https://doi.org/10.1016/j.tre.2011.08.001.

[42] O. Sassi, W. Ramdane Cherif, and A. Oulamara, *Vehicle Routing Problem with Mixed fleet of conventional and heterogenous electric vehicles and time dependent charging costs*, Oct. 2014.

[43] S. Shao, W. Guan, B. Ran, Z. He, and J. Bi, "Electric Vehicle Routing Problem with Charging Time and Variable Travel Time," *Math. Probl. Eng.*, vol. 2017, M. Pellicciari, Ed., pp. 1–13, 2017. DOI: 10.1155/2017/5098183.

[44] J. Barco, A. Guerra, L. Muñoz, and N. Quijano, "Optimal Routing and Scheduling of Charge for Electric Vehicles: A Case Study," *Math. Probl. Eng.*, vol. 2017, D. Quagliarella, Ed., p. 8 509 783, 2017. DOI: 10.1155/2017/8509783.

[45] R. Peltokangas and A. Sorsa, *Real-coded genetic algorithms and nonlinear parameter identification Riikka Peltokangas and Aki Sorsa*, 34. 2008.

[46] K. Ghoseiri and S. F. Ghannadpour, "Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm," *Appl. Soft Comput. J.*, vol. 10, no. 4, pp. 1096–1107, 2010. DOI: 10.1016/j.asoc.2010.04.001.

[47] H. Nazif and L. S. Lee, "Optimised crossover genetic algorithm for capacitated vehicle routing problem," *Appl. Math. Model.*, vol. 36, no. 5, pp. 2110–2117, 2012. DOI: 10.1016/j.apm.2011.08.010.

[48] K. C. Tan, L. H. Lee, Q. L. Zhu, and K. Ou, "Heuristic methods for vehicle routing problem with time windows," *Artif. Intell. Eng.*, vol. 15, no. 3, pp. 281–295, 2001. DOI: https://doi.org/10.1016/S0954-1810(01)00005-X.

[49] B. Ombuki, B. J. Ross, and F. Hanshar, "Multi-objective genetic algorithms for vehicle routing problem with time windows," *Appl. Intell.*, vol. 24, no. 1, pp. 17–30, 2006. DOI: 10.1007/s10489-006-6926-z.

[50] B. M. Baker and M. A. Ayechew, "A genetic algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 30, no. 5, pp. 787–800, 2003. DOI: https://doi.org/10.1016/S0305-0548(02)00051-5.

[51] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Comput. Methods Appl. Mech. Eng.*, vol. 191, no. 11-12, pp. 1245–1287, 2002. DOI: 10.1016/S0045-7825(01)00323-1.

[52] C. Fiori, K. Ahn, and H. A. Rakha, "Power-based electric vehicle energy consumption model: Model development and validation," *Appl. Energy*, vol. 168, pp. 257–268, 2016. DOI: 10.1016/j.apenergy.2016.01.097.

[53] G. Hiermann, R. F. Hartl, J. Puchinger, and T. Vidal, "Routing a mix of conventional, plug-in hybrid, and electric vehicles," *Eur. J. Oper. Res.*, vol. 272, no. 1, pp. 235–248, 2019. DOI: https://doi.org/10.1016/j.ejor.2018.06.025.

Appendix A

Development of Mass-dependent Energy Consumption Equations

The energy consumption when an EV departs from node i towards node j at time t_0, carrying a weight w, is denoted $E_{ij}(w, t_0)$. Calculating this value can be a very computationally expensive procedure, specially using shortest paths, because we must calculate it for each payload and for each time combination. Therefore, we develop a method to easily calculate $E_{ij}(w, t_0)$ for any valid payload w and time t_0. This is a two-stage procedure that consists of using known no-cargo energy consumption, and a realistic EV energy consumption model to include more payload.

First, we divide the times of the day into h equally-spaced periods of length ΔT. Second, we assume that, at the beginning of each period, the no-cargo energy consumption is available. That is, we know the values

$$E_{ij}(0, 0),\ E_{ij}(0, \Delta T), \ldots,\ E_{ij}(0, (h-1)\Delta T)$$

There are two ways to obtain the no-cargo energy consumption values, which depend if we consider linear arcs, or shortest paths. If the arc is a line connecting two nodes, we can use the realistic EV energy consumption below and replace the distance by the Euclidean distance. On the other hand, if using shortest paths, we use the methodology in Section 4.3.2 using the EV energy consumption model below. In both cases, we must consider the EV plus driver mass only.

To include additional payload, we use the realistic energy consumption model from [52]. The model states that the wheels consume an instant power

$$P_1(t) = v(t) \cdot \left[ma(t) + mg \cdot \cos\left(\theta(t)\right) \cdot \frac{C_r}{1000} \left(c_1 v(t) + c_2\right) + \frac{1}{2} \rho_{air} A_f C_D v^2(t) + mg \cdot \sin(\theta(t)) \right] \tag{A.1}$$

where t is the time, $v(t)$ is the EV velocity, $a(t)$ is the EV acceleration, and $\theta(t)$ is the road grade. Table A.1 details all remaining model parameters.

When the EV accelerates, $P_1(t)$ is positive. In that case, the battery delivers power. If the EV decelerates, $P_1(t)$ is negative, and the battery can recharge. Nonetheless, the real delivered and recovered energies are limited by the EV efficiencies (summarized in Table

Table A.1: Description of EV energy consumption model parameters

Description	Parameter
Mass (EV + driver + payload)	m
Rolling resistance coefficients	C_r
	c_1
	c_2
Air mass density	ρ_{air}
Frontal are of the EV	A_f
Gravitational acceleration	g
Aerodynamic EV drag coefficient	C_D
Driveline efficiency	$\eta_{Driveline}$
Electric motor efficiency	η_{Motor}
EV battery efficiency	$\eta_{Battery}$
Regenerative braking efficiency	η_{RB}

A.1). Thus, the real power is

$$P(t) = \begin{cases} \dfrac{P_1(t)}{\eta_{Driveline} \cdot \eta_{Motor} \cdot \eta_{Battery}} & \text{if } a(t) \geqslant 0 \\ \eta_{RB} \cdot P_1(t) & a(t) < 0. \end{cases} \tag{A.2}$$

The total energy the battery delivers is the integral of the instant power since the EV starts moving:

$$E(t) = \int_{t_{start}}^{t} P(\tau) d\tau \tag{A.3}$$

In the real-world, the EV may accelerate, decelerate, and even stop when traveling through an arc. Nonetheless, in our problem, we just worry about the time it departs from i and the time it arrives at j. Therefore, we can assume that the EV travels from i towards j with a constant velocity

$$v_{ij}(t_0) = \frac{d_{ij}}{t_{ij}(t_0)},$$

where d_{ij} is the length of the arc between i and j, and $t_{ij}(t_0)$ the travel time at time of the day t_0. Thus, the acceleration is zero throughout the arc. We also consider that the road grade is fixed, i.e. $\theta(t)$ is constant and has the value θ. Under this scenario, we can set the velocity and keep it constant throughout the arc. The latter yields to a constant instant power

$$P_{ij}(t_0) = v_{ij}(t_0) \cdot \left[mg \cdot \cos(\theta) \cdot \frac{C_r}{1000} \left(c_1 v_{ij}(t_0) + c_2 \right) + \frac{1}{2} \rho_{air} A_f C_D v_{ij}^2(t_0) + mg \cdot \sin(\theta) \right].$$

As the instant power is constant, the integral turns into the product between the travel

time and the instant power:

$$\begin{aligned} E_{ij}(t_0) &= t_{ij}(t_0) \cdot P_{ij}(t_0) \\ &= d_{ij} \cdot \left[mg \cdot \cos(\theta) \cdot \frac{C_r}{1000}(c_1 v_{ij}(t_0) + c_2) + \frac{1}{2}\rho_{air} A_f C_D v_{ij}^2(t_0) + mg \cdot \sin(\theta) \right] \\ &= m \cdot \alpha + \beta \end{aligned} \tag{A.4}$$

where

$$\begin{aligned} \beta &= \frac{1}{2} d_{ij} \rho_{air} A_f C_D v_{ij}^2(t_0) \\ \alpha &= d_{ij} g \cdot \cos(\theta) \cdot \frac{C_r}{1000}(c_1 v_{ij}(t_0) + c_2) + d_{ij} g \cdot \sin(\theta) \end{aligned}$$

Notice that d_{ij} and $v_{ij}^2(t_0)$ are known. Thus, we can calculate the value β. It may occur that θ is not known; however, the below-explained method does not require knowing θ.

We decompose the total mass into EV mass and driver (w_{EV}), and the payload mass (w):

$$m = w_{EV} + w.$$

Now, remember that the energy consumption that depends on the payload w is denoted $E_{ij}(w, t_0)$, and the no-cargo energy consumption $E_{ij}(0, t_0)$ is known. Therefore, we can use (A.4) to write α as

$$\alpha = \frac{E_{ij}(0, t_0) - \beta}{w_{EV}}. \tag{A.5}$$

Finally, to estimate the energy considering both the EV mass and the payload, replace (A.5) into (A.4) to find a definition of the energy consumption that depends on the payload and the no-cargo energy consumption:

$$E_{ij}(w, t_0) = E_{ij}(0, t_0) \cdot \left[1 + \frac{w}{w_{EV}} \right] - \frac{w}{w_{EV}} \cdot \beta \tag{A.6}$$

When the battery delivers an energy $E_{ij}(w, t_0)$, the battery SOC reduces by

$$e_{ij}(w, t_0) = \frac{E_{ij}(w, t_0)}{Q}, \tag{A.7}$$

where Q is the battery capacity.

Appendix B

Initial Population Algorithms

B.1 αGA

Algorithm 7 constructs an individual as follows. First, the algorithm orders customers according to their time window lower bounds. Second, it traverses the ordered customers and adds them to a vehicle's route until the weight limit is surpassed. Third, the created route is assigned to the vehicle, and a new one is created starting from the current customer. The algorithm repeats the process until it sets all customers. Charging operations and departure times are created randomly. The heuristic returns both a candidate individual and fleet size.

Algorithm 7: ConstructIndividual1-A

Input : Array with customers to visit ($\mathcal{A}$); maximum payload ($\bar{D}$)
Output: An individual candidate I, fleet size m

$\mathcal{A}^* \leftarrow$ Customers in $\mathcal{A}$ sorted in increasing order according to their time window lower bounds;
$R \leftarrow []$;
$r \leftarrow []$;
$W \leftarrow 0$;
for i in $\mathcal{A}^*$ **do**
    $d \leftarrow$ Requirement at i;
    if $W + d \leqslant \bar{D}$ **then**
        Append i to r;
        $W \leftarrow W + d$
    else
        Append r to R;
        $r \leftarrow []$;
        Append i to r;
        $W \leftarrow d$
    end
end
$m \leftarrow$ Routes in R;
C_block $\leftarrow$Create customer block with delimiters using routes from R;
CO_block $\leftarrow$Random CPB-1 block with $2m$ charging operations that will not occur;
DT_block $\leftarrow$List of size m with the lower time window bound of the first customers for each route in R;
$I \leftarrow$Concatenate(*C_block*, *CO_block*, *DT_block*);
return I, m

Algorithm 8 creates individual candidates by ordering customers according to their time window lower bound. In contrast to Algorithm 7, Algorithm 8 receives the fleet size and calculates how many customers each EV will receive. Thus, it does not check if the weight limit is surpassed. Charging operations and departure times are created randomly.

Algorithm 8: ConstructIndividual2-A

Input : Array with customers to visit ($\mathcal{A}$); fleet size (m)
Output: An individual candidate I

$\mathcal{A}^* \leftarrow$ Customers in $\mathcal{A}$ sorted in increasing order according to their time window lower bounds;
$l \leftarrow$Integer(len$\mathcal{A}/m$)
$C\_block \leftarrow []$;
for i in $1, \ldots, m$ **do**
    if $i == m$ **then**
        Append the rest of $\mathcal{A}^*$ to C_block;
    else
        $customers \leftarrow$ Pop l elements from $\mathcal{A}^*$;
        Append $customers$ to C_block;
        Append '|' to C_block;
    end
end
$CO_block \leftarrow$Random CPB-1 block with $2m$ charging operations that will not occur;
$DT_block \leftarrow$List of size m with the lower time window bound of the first customers for each route in R;
$I \leftarrow$Concatenate(C_block, CO_block, DT_block);
return I, m

B.2 βGA

Algorithm 9 turns the best individual found by αGA into an individual using βGA encoding. As the CPB-2 in the sub-individuals of βGA only permit a single CS between two nodes, we turn a charging path with multiple visits to CS into a single CS visit by only allowing the

EV to perform the last charging operation in the charging path with multiple visits.

Algorithm 9: ConstructIndividual-B

Input : Optimal individual αGA finds ($I^{*\alpha}$)
Output: A random individual I

$I \leftarrow []$;
$S, L, x_0 \leftarrow \Psi^{\alpha}(I^{*\alpha})$;
for S^i, L^i in S, L **do**
  $customer\_block \leftarrow []$;
  $CPB2 \leftarrow []$;
  $g \leftarrow 0$;
  for S^i_k, L^i_k in S, L **do**
    if $L^i_k > 0$ **then**
      if $g > 0$ **then**
        $CBP1[-2:] = [S^i_k, L^i_k]$;
      else
        Append $[S^i_k, L^i_k]$ to $CBP2$;
      end
      $g \leftarrow g + L^i_k$;
    else
      Append S^i_k to $customer\_block$;
      Append $[-1, \mathcal{U}(5, 20)]$ to $CBP2$;
      $g \leftarrow 0$;
    end
  end
  Append Concatenate($customer\_block$, CPB2, 0) to I;
end
return I

B.3 OnGA

OnGA constructs a candidate individual using the routes from the previous OnGA execution. Algorithm 10 does this procedure by creating the individual using old routes, starting from the critical node. Then, it modifies the CPB-1 according to the old recharging plan. The offset is always set to zero. We remark that this method assumes that the CPB-1 size is

enough to store all charging operations in the old recharging plan.

Algorithm 10: ConstructIndividual-On

Input : Ahead routes, starting right after the critical node (S, L); CPB-1 size (n^*)
Output: A constructed individual I

$I \leftarrow []$;
for S^i, L^i in S, L **do**
    CPB-1$\leftarrow []$;
    Traverse L^i to find each L^i_k representing a charging operation;
    Turn the charging operations into CPB-1 sub-blocks;
    Append the sub-blocks to CPB-1;
    if The CPB-1 has less sub-blocks than n^* **then**
        Fill with charging operations that will not occur;
    customer_block $\leftarrow$ Customers in S^i in the order they appear;
    Append Concatenate(*customer_block*, CPB-1, 0) to I;
end
return I