Developing Efficient Routing Algorithms for Sustainable City Logistics

A dissertation presented in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

by

Mehmet Anil Akbay

Supervisor:

Dr. Christian Blum

Tutor:

Dr. Felip Manyà

Institut d'Investigació en Intel.ligència Artificial, CSIC
Programa de Doctorat en Informàtica
Universitat Autònoma de Barcelona

February, 2025

ABSTRACT

Route planning for vehicles in freight distribution has long been a primary objective in logistics. Researchers and practitioners have devoted considerable effort to developing models and solution methods to optimize routes for a fleet of vehicles transporting goods from supply points to demand points. However, the global environmental crisis, mainly caused by rapid industrialization, population growth, and urbanization, has underscored the need for sustainable solutions for logistic operations. Rising concerns over environmental pollution, noise, traffic congestion, and population growth, especially in large cities, highlight the necessity to consider social and environmental issues alongside economic and operational efficiency in the design of logistics systems.

In this line, this thesis addresses the critical need for sustainability in urban logistics by exploring advanced solutions that integrate electric vehicles (EVs) and multi-echelon distribution systems into *Vehicle Routing Problems*. Situated at the intersection of *combinatorial optimization* and *sustainable urban logistics*, this work focuses on both algorithmic advancements and the development of comprehensive routing models.

The algorithmic contributions primarily involve enhancing and applying the *Construct, Merge, Solve & Adapt (CMSA)* algorithm, particularly by developing a self-adaptive variant known as *Adapt-CMSA*. This algorithm variant addresses the issue of parameter sensitivity in metaheuristics, where performance often heavily depends on specific parameter settings. *Adapt-CMSA* aims to reduce this sensitivity, ensuring robust performance across various problem sizes and complexities without the need for parameter re-tuning. Its effectiveness is first shown in the context of the *Minimum Positive Influence Dominating Set Problem*, where it outperformed the standard *CMSA* by dynamically adjusting its parameters, thereby enhancing efficiency and scalability.

From a practical standpoint, this thesis tackles the formulation of complex routing problems that mirror real-world scenarios in sustainable urban logistics. It introduces models for the *Electric Vehicle Routing Problems* and *Two-Echelon Electric Vehicle Routing Problems*, integrating critical constraints such as time windows, simultaneous pickup and deliveries, and partial recharging strategies. Additionally, the thesis goes beyond traditional objective functions considering distance minimization. In particular, we consider energy-minimization that is affected by several factors such as vehicle speed and load. By effectively employing a range of heuristic and metaheuristic approaches, the thesis provides

practical solutions to complex variants of the addressed problems.

Keywords: *Sustainable Logistics*, *Electric Vehicle Routing*, *Two-Echelon Freight Distribution*, *Time Windows*, *Simultaneous Pickup & Delivery*, *Partial Recharging*, *Minimum Dominating Set*, *Minimum Positive Influence Dominating Set*, *CPLEX*, *Variable Neighborhood Search*, *Construct Merge Solve & Adapt*, *Ant Colony Optimization*.

RESUMEN

La planificación de rutas para vehículos en la distribución de mercancías ha sido durante mucho tiempo un objetivo primordial en logística. Investigadores y profesionales han dedicado considerables esfuerzos a desarrollar modelos y métodos de resolución para optimizar rutas para flotas de vehículos que transportan bienes desde puntos de suministro a puntos de demanda. Sin embargo, la crisis ambiental global, profundizada por la rápida industrialización, el crecimiento de la población y la urbanización, ha subrayado la necesidad de soluciones sostenibles para tales operaciones logísticas. Las crecientes preocupaciones sobre la contaminación ambiental, el ruido, la congestión ocasionada por el tráfico y el crecimiento de la población, especialmente en las grandes ciudades, destacan la necesidad de considerar los problemas sociales y ambientales junto con la eficiencia económica y operativa en el diseño de los sistemas logísticos.

En esta línea, esta tesis aborda la necesidad crítica de sostenibilidad en la logística urbana explorando soluciones avanzadas que integran vehículos eléctricos (EVs) y sistemas de distribución de múltiples niveles en los *Problemas de Enrutamiento de Vehículos*. Situado en la intersección de la *optimización combinatoria* y la *logística urbana sostenible*, este trabajo se centra tanto en los avances algorítmicos como en el desarrollo de modelos de enrutamiento integrales.

Las contribuciones algorítmicas involucran principalmente la mejora y aplicación del algoritmo *Construir, Fusionar, Resolver y Adaptar (CMSA)*, particularmente desarrollando una variante auto-adaptativa conocida como *Adapt-CMSA*. Esta variante aborda el desafío de la sensibilidad a los parámetros en las metaheurísticas, donde el rendimiento a menudo depende en gran medida de configuraciones de parámetros específicos. *Adapt-CMSA* tiene como objetivo reducir esta sensibilidad, asegurando un rendimiento robusto en diversos tamaños y complejidades de problemas sin la necesidad de reajustar los parámetros. Su efectividad fue particularmente evidente cuando se aplicó al *Problema de Conjunto Dominante de Influencia Positiva Mínima*, donde superó al *CMSA* estándar al ajustar dinámicamente sus parámetros, mejorando así la eficiencia y la escalabilidad.

Desde un punto de vista práctico, esta tesis aborda la formulación de problemas complejos de enrutamiento que reflejan escenarios del mundo real en la logística urbana sostenible. Introduce modelos para los *Problemas de Enrutamiento de Vehículos Eléctricos* y *Problemas de Enrutamiento de Vehículos Eléctricos de Dos Niveles*, integrando restricciones críticas como ventanas de

tiempo, recogidas y entregas simultáneas, y estrategias de recarga parcial. Además, la tesis va más allá de las funciones objetivo tradicionales de minimización de distancias al considerar también la minimización de energía, afectada por varios factores como la velocidad del vehículo y su carga. Al emplear efectivamente una gama de enfoques heurísticos y metaheurísticos, la tesis proporciona soluciones prácticas a variantes complejas de los problemas abordados.

RESUM

La planificació de rutes per a vehicles en la distribució de mercaderies ha estat durant molt de temps un objectiu primordial en logística. Investigadors i professionals han dedicat considerables esforços a desenvolupar models i mètodes de resolució per a optimitzar rutes per a flotes de vehicles que transporten béns des de punts de subministrament a punts de demanda. No obstant això, la crisi ambiental global, profunditzada per la ràpida industrialització, el creixement de la població i la urbanització, han subratllat la necessitat de solucions sostenibles per a les operacions logístiques. Les creixents preocupacions sobre la contaminació ambiental, el soroll, la congestió deguda al trànsit i el creixement de la població, especialment en les grans ciutats, destaquen la necessitat de considerar els problemes socials i ambientals juntament amb l'eficiència econòmica i operativa en el disseny dels sistemes logístics.

En aquesta línia, aquesta tesi aborda la necessitat crítica de sostenibilitat en la logística urbana explorant solucions avançades que integren vehicles elèctrics (EVs) i sistemes de distribució de múltiples nivells en els *Problemes d'Enrutament de Vehicles*. Situat a la intersecció de la *optimització combinatòria* i la *logística urbana sostenible*, aquest treball se centra tant en els avenços algorísmics com en el desenvolupament de models d'enrutament integrals.

Les contribucions algorísmiques involucren principalment la millora i aplicació de l'algorisme *Construir, Fusionar, Resoldre i Adaptar (CMSA)*, particularment desenvolupant una variant auto-adaptativa coneguda com *Adapt-CMSA*. Aquesta variant aborda el repte de la sensibilitat als paràmetres en les metaheurístiques, on el rendiment sovint depèn en gran mesura de configuracions de paràmetres específics. *Adapt-CMSA* té com a objectiu reduir-ne aquesta sensibilitat, assegurant-ne un rendiment robust en diverses mides i complexitats de problemes sense la necessitat de reajustar-ne els paràmetres. La seva efectivitat va ser particularment evident en ser aplicat al *Problema del Conjunt Dominant de Influència Positiva Mínima*, on va superar el *CMSA* estàndard en ajustar dinàmicament els seus paràmetres, millorant-ne així l'eficiència i l'escalabilitat.

Des d'un punt de vista pràctic, aquesta tesi aborda la formulació de problemes complexos d'enrutament que reflecteixen escenaris del món real en la logística urbana sostenible. Introdueix models per als *Problemes d'Enrutament de Vehicles Elèctrics* i *Problemes d'Enrutament de Vehicles Elèctrics de Dos Nivells*, integrant restriccions crítiques com ara finestres de temps, recollides i entregues simultànies, i estratègies de recàrrega parcial. A més, la tesi va més enllà de les funcions objectiu tradicionals de minimització de distàncies en considerar també

la minimització d'energia, afectada per diversos factors com ara la velocitat del vehicle i la seva càrrega. En emprar de manera efectiva una gamma d'enfocs heurístics i metaheurístics, la tesi proporciona solucions pràctiques a variants complexes dels problemes abordats.

ACKNOWLEDGEMENT

As I reach the end of my PhD journey, I am filled with immense happiness and gratitude. This period has undoubtedly been both challenging and rewarding, and I am thrilled to have reached this milestone.

Doing a PhD is like fighting a battle against oneself. The challenges in the field are immense, but the realization that the true struggle is within brings profound insight. The journey was like climbing a very steep mountain; I felt the pain in my muscles, but the hope of reaching the beautiful scene at the top was one of the things that kept me going. However, this hope is not always sufficient to keep you on this challenging track. External support and motivation are essential, and I believe I have had plenty of it.

Firstly, I would like to express my heartfelt gratitude to my supervisor, Dr. Christian Blum. I always think that the most crucial aspect of a successful PhD is having a good supervisor, not only in terms of research skills but also on a personal level. In this regard, I consider myself extremely fortunate to have such a great guide and support beside me on this journey. Dr. Blum is an outstanding researcher, a fact that goes without saying, but he is also an exceptional person. He has been consistently supportive and helpful, not only with my PhD studies but also with my personal issues. Whenever I needed something, he was there, and I cannot adequately express how much value he has contributed to my life.

I also want to extend my sincere thanks to my tutor, Dr. Felip Manya. His positivity and support have been invaluable, always motivating me, which is one of the core necessities during the PhD process.

I am deeply grateful to my Master's supervisor, Assoc. Prof. Can Berk Kalayci, with whom I began my research journey. It was through him that I had the opportunity to meet Dr. Blum, which became a significant turning point in my academic career. I am very thankful for this pivotal moment and all the things I have learned from him.

I would like to thank the Republic of Turkiye Ministry of National Education for providing YLYS funding, which made this journey possible.

I am also grateful to the Spanish Ministry of Science and Innovation, which funded all the research in this thesis through project MCIN/AEI/10.13039/501100011033 (Grants PID2022-136787NB-I00 and TED2021-129319B-I00).

The Artificial Intelligence Research Institute (IIIA-CSIC) provided an outstanding working environment during my PhD studies. I am thankful for the physical resources and the friendly atmosphere. Sharing my time and the

atmosphere with my colleagues in the first and second-floor offices, along with all visiting PhD students and researchers, has been a wonderful experience.

I want to thank my family—my mother, father, and siblings—for their unwavering support. Their encouragement and belief in me have been a constant source of strength.

Finally, I wish to express my profound appreciation to my wife and daughter. Without them, my life would be incomplete. They are the most precious treasures of my life, illuminating my path with their unwavering love and support. Their presence constantly reminds me of how fortunate I am, even when I fail to see it. I am grateful for their patience and emotional support during this journey and, most importantly, for transforming my grayscale world into a vibrant, blooming garden of joy.

Thank you all.

CONTENTS

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Environmental pollution, an inevitable consequence of rapid industrialization and population growth, has become a critical global challenge. This escalating crisis is largely attributed to the uncontrolled exploitation of natural resources and the consequent release of pollutants into the environment. These activities, coupled with the expansion of urban areas and the industrial sector, have caused a significant deterioration of air, water, and soil.

One of the most concerning issues of this environmental degradation is air pollution, which is directly linked to the extensive use of fossil fuels across various sectors such as energy production, transportation, and manufacturing industries. The increasing presence of air pollutants not only deteriorates environmental quality but also poses severe health risks. Prolonged exposure to poor air quality has been associated with various health problems, from respiratory illnesses [85] to cardiovascular diseases [136], impacting the well-being and productivity of populations.

Beside the individual effects of air pollutants, there is also a significant global effect due to greenhouse gases such as carbon dioxide (CO_2), methane (CH_4), and nitrous oxide (N_2O). Emitted through human activities across various sectors, these gases escalate the crisis to a global level, contributing to global climate change and intensifying environmental and health challenges worldwide.

In a concerning development, 2022 witnessed a record high in greenhouse gas emissions, with no indication of a downward trend, see Figure 1.1. The World Meteorological Organization reported that the global average concentration of CO_2 reached levels 50% higher than those in the pre-industrial era [190]. Furthermore, the International Energy Agency's 2023 report indicated that global energy-related CO_2 emissions increased by 0.9% or 321 Mt in 2022, totaling over 36.8 Gt [97]. This rise indicates the urgency of the ongoing challenges in reducing emissions from energy combustion and industrial processes.

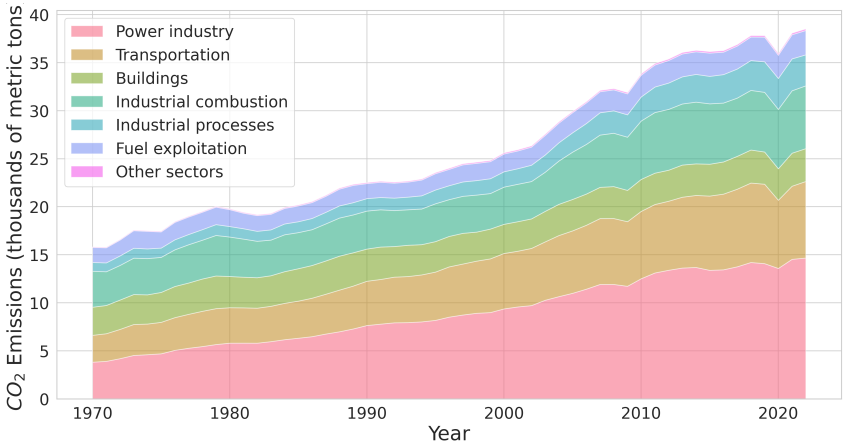


Fig. 1.1 Global CO_2 Emission by Sector Over Time [48].

The logistics sector, due to its vital role in the global economy, is one of the primary contributors to environmental pollution, especially in greenhouse gas emissions, see Figure 1.2. Logistic activities encompass various modes of transportation, such as air, marine, and road transport. According to the International Energy Agency, the transport sector was responsible for approximately 23% of global energy-related CO_2 emissions in 2015, with road transport alone accounting for 72% of these emissions. Aviation and maritime transport also contributed significantly, with 10% and 9.6% of the sector's emissions, respectively [97].

Recognizing environmental issues as complex and layered, spanning from local to the global impacts of pollution and greenhouse gas emissions, underscores the critical need for comprehensive solutions. Within this framework, *sustainability* has emerged as a pivotal concept. As the World Commission on Environment and Development outlines, sustainability entails the cautious use of resources to fulfil current needs without threatening the capacity of future generations to meet theirs [188]. This holistic approach reduces pollution, preserves biodiversity, and promotes social equity [77]. With a clear understanding of the need for eco-friendly practices across different sectors, the logistics industry, notable for its significant environmental footprint, is no exception in adopting sustainable strategies to lessen its adverse environmental effects [125]. This commitment is further supported by *the United Nations Sustainable Development Goals (SDGs)* which provide a global framework that includes targets specifically relevant to sustainable logistics practices (see Figure 1.3). These goals underscore the necessity of integrating environmental considerations with socio-economic development, highlighting areas such as responsible consumption and production, climate action, and decent work

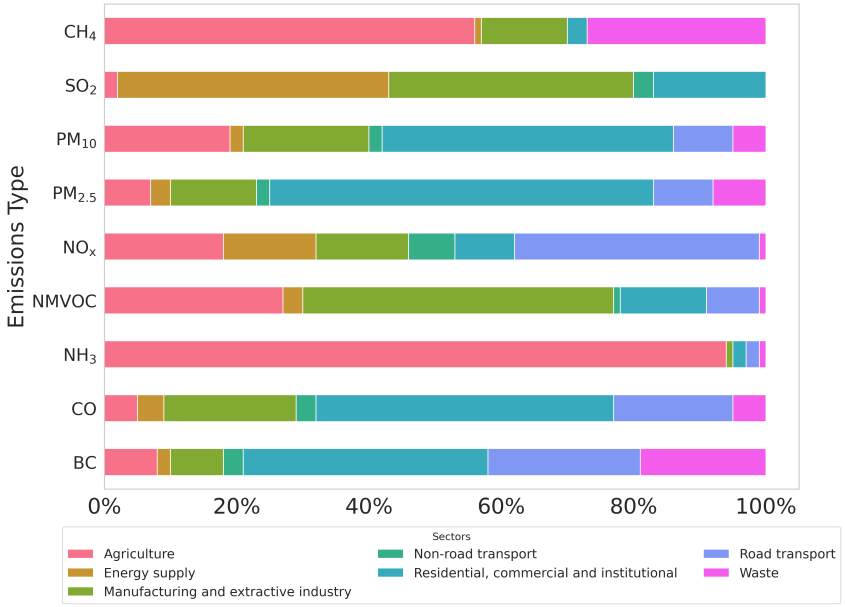


Fig. 1.2 Percentage Contribution of Various Sectors to EU-27 Emissions. The emission types represented on the y-axis include Black Carbon (BC), a component of fine particulate matter contributing to climate change; Carbon Monoxide (CO), a colorless and toxic air pollutant; Ammonia (NH_3), primarily from agricultural processes; Non-Methane Volatile Organic Compounds (NMVOCs), which contribute to ozone formation; Nitrogen Oxides (NO_x), involved in the formation of smog and acid rain; Particulate Matter with diameters of 2.5 micrometers and smaller ($PM_{2.5}$) and 10 micrometers and smaller (PM_{10}), affecting respiratory health; Sulfur Dioxide (SO_2), a precursor to acid rain; and Methane (CH_4), a potent greenhouse gas [63].

conditions and economic growth, which are directly applicable to the logistics sector [180].

As a critical aspect of this sustainability movement, sustainable logistics specifically targets the environmental impact of logistics activities [33]. It aims to minimize the ecological footprint of the logistics sector by implementing energy-efficient strategies, reducing waste, and promoting the use of renewable resources [132]. Beyond environmental considerations, sustainable logistics also encompasses social and economic aspects, such as ensuring fair working conditions and promoting local economies [33]. The ultimate goal of sustainable logistics is to strike a balance between meeting the demands of consumers, businesses, and the environment, ensuring that logistic activities contribute to long-term environmental, social, and economic well-being [132].

Sustainable city logistics, on the other hand, a more focused area within sustainable logistics, concentrates on urban areas where the environmental impact of transportation is particularly significant [175]. As urban populations

Fig. 1.3 The Sustainable Development Goals and their relevance to sustainable logistics [180].

continue to grow and the demand for goods and services in cities increases, the need for efficient and sustainable transportation solutions becomes even more critical [47]. In this line, sustainable city logistics aims to optimize the flow of goods and people within cities while minimizing the adverse effects on the environment, such as air pollution, noise pollution, and traffic congestion [175]. This can be achieved through various strategies, including the use of environmentally friendly vehicles, promoting multi-modal transport, implementing smart traffic management systems, and adopting multi-echelon distribution systems [47].

Two key approaches have been increasingly recognized in both theoretical and practical aspects of sustainable city logistics: firstly, the adoption of environmentally friendly vehicles, and secondly, the implementation of multi-echelon distribution systems, supplemented by strategic restrictions such as limited service times and reverse logistics practices.

Environmentally friendly vehicles, including electric vehicles (EVs), hybrid vehicles, and those utilizing alternative fuels, play a crucial role in reducing exhaust gas emissions, thereby contributing to the sustainability goals set by governments and international organizations [125].

Figure 1.4 underlines the pivotal role of electrification in the transport sector's contribution to CO_2 emissions reduction in the *Sustainable Development Scenario*. It demonstrates that the shift towards electrification, including the use of EVs, represents the most significant share of emission reductions compared to other actions such as demand avoidance, hydrogen usage, bioenergy, and other renewables. This underscores the importance of accelerating the adoption of EVs within the logistics sector. These vehicles not only offer long-term financial benefits through reduced maintenance and fuel costs but also assist logistics

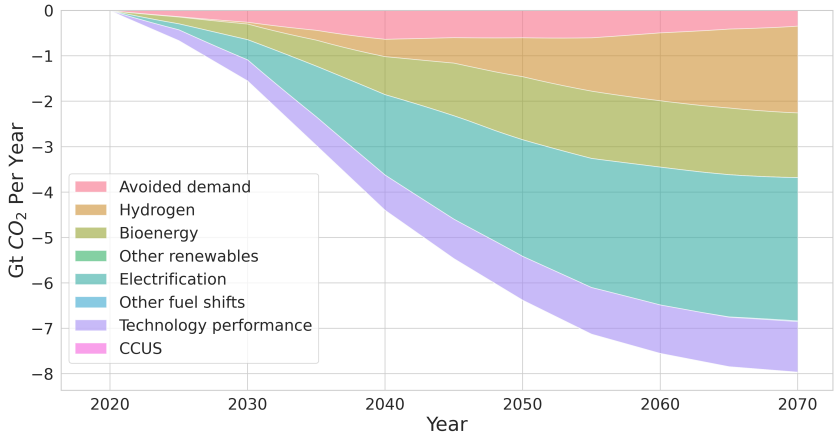


Fig. 1.4 CO_2 emissions reductions in the transport sector in the Sustainable Development Scenario relative to the Stated Policies Scenario [98].

companies in adhering to environmental regulations, fulfilling corporate social responsibility, and enhancing their public image [125]. However, challenges such as limited driving range, higher initial costs, restricted availability, and battery disposal issues remain key considerations in the broader adoption of these vehicles.

Complementing the adoption of environmentally friendly vehicles, multi-echelon distribution systems emerge as another essential component of sustainable city logistics, offering innovative strategies to address the complexities of urban freight movement. This logistic approach involves transporting goods through a series of stages, each utilizing various transportation modes and facilities strategically designed to enhance efficiency while reducing environmental impact. In this system, the supply chain is segmented into multiple 'echelons' or stages. At each stage, goods are consolidated at designated facilities, often referred to as intermodal transshipment hubs or 'satellites,' before being transferred to different vehicles for the next phase of delivery [127].

The significance of multi-echelon distribution in city logistics is profound. Segmenting the transportation process allows for a more effective consolidation of goods. This segmentation leads to substantial benefits such as diminished traffic congestion, lower air pollution levels, and reduced transportation costs, thereby aligning closely with the objectives of city logistics which prioritize the minimization of social, economic, and environmental impacts [127]. A critical aspect of this approach is its capability to decrease the presence of large trucks in urban centers, opting instead for smaller, eco-friendly vehicles for final delivery stages. Such a shift not only boosts the efficiency of goods movement but also markedly improves urban air quality and mitigates traffic congestion [127].

Nevertheless, the implementation of multi-echelon distribution systems entails its own set of challenges. It demands meticulous strategic planning, effective coordination among diverse stakeholders, and the establishment of suitable infrastructures like urban consolidation centers (UCCs). These UCCs serve as central hubs where goods from various sources are aggregated before their distribution across the city, thereby reducing the number of delivery vehicles and enhancing the overall efficacy of urban freight operations [45]. The success of these systems hinges on optimal transportation planning and management, necessitating advanced decision-making tools and technologies. The deployment of Intelligent Transportation Systems (ITS) and sophisticated fleet management practices becomes crucial for the control and synchronization of logistics activities within these systems [45].

Another sustainable logistics practice is the implementation of restrictions, such as limited service time for customers (time windows) and reverse logistics practices. Time windows impose constraints on when deliveries can be made, which can lead to more efficient route planning, reduced congestion, and lower emissions [175]. This practice also encourages better use of transportation resources and contributes to improved air quality in urban areas. However, time window restrictions can also pose challenges for logistics companies, as they must balance customer needs with the constraints imposed by local authorities.

Reverse logistics practices, which involve the collection, transportation, and processing of used, damaged, or obsolete products and materials, can also contribute to sustainable city logistics [157]. By facilitating the recycling, refurbishment, or disposal of products, reverse logistics helps reduce waste, conserve resources, and minimize the environmental impact of logistics activities. Moreover, it can provide economic benefits for companies by recovering value from returned products and reducing disposal costs. However, implementing reverse logistics practices can be complex and resource-intensive, requiring substantial investments in infrastructure, technology, and personnel [157].

In conclusion, sustainable city logistics encompasses a range of practices, including the use of environmentally friendly vehicles, multi-echelon distribution systems, time window restrictions, and reverse logistics practices. By adopting these strategies, logistics companies can reduce their environmental impact, enhance their sustainability performance, and contribute to a cleaner, greener urban environment.

Considering the urgent need to address environmental pollution and the significant role of road transport activities in contributing to this issue, it is essential to explore sustainable solutions within logistics. The concept of

sustainable city logistics is vital in this context, as it focuses on minimizing the environmental impact of transportation activities in urban areas. This thesis seeks to investigate electric vehicle routing problems and two-echelon electric vehicle routing problems as key components of sustainable city logistics, providing innovative ways to achieve a cleaner, greener future. By addressing variations of these problems, the thesis takes into account real-life constraints, such as time windows, simultaneous pickups and deliveries, partial battery charging and realistic transportation networks considering road junctions and road segments with different speed limitations for vehicles. The aim of this study is not only to contribute valuable insights to the field of sustainable logistics but also to develop solution approaches that can effectively solve the addressed problems.

1.2 Overview of Vehicle Routing Problems

The Vehicle Routing Problem (VRP) introduced by Dantzig and Ramser [50] represents a fundamental challenge in operations research and logistics. The problem involves determining the most efficient routes for a fleet of vehicles originating from a central depot or multiple distribution centers to a network of demand points or customers. The objective is generally minimizing travel distances, time, and overall logistical costs.

The VRP extends the principles of the well-known Traveling Salesman Problem (TSP). While the TSP seeks the shortest route for a single vehicle to visit each customer exactly once before returning to the origin, the VRP adds layers of complexity with multiple vehicles, often subject to loading capacity limits. This requires not just the determination of a singular optimal route but an entire set of efficient routes for the whole fleet. Figure 1.5 illustrates a typical VRP setup, including a central depot, customer locations, and routes for a fleet of vehicles.

Following the foundational work by Dantzig and Ramser [50], the field of VRPs has expanded significantly to include a variety of scenarios, reflecting the diverse challenges faced in real-world logistics and distribution systems. This evolution has led to the introduction of numerous VRP variants, each designed to address specific constraints and requirements in logistics planning.

Models such as the VRP with time windows (VRPTW) [173], the VRP with time limit (VRPTL) [177] and the Time-dependent VRP(TDVRP) [126] ensure that deliveries and services are scheduled within feasible time frames, accommodating the dynamic nature of traffic conditions and customer availability.

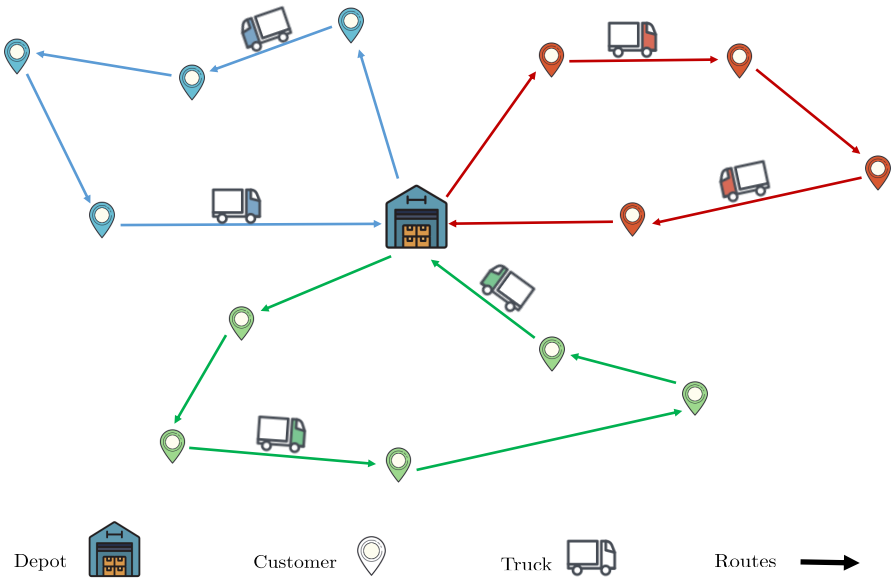


Fig. 1.5 Illustration of a Capacitated Vehicle Routing Problem. Icons by Icons8 (icons8.com).

Furthermore, the domain of VRPs has extended to involve various demand scenarios, notably through the Pickup and Delivery Problem (PDP) [165]. This category includes the VRP with backhauls (VRPB) [81], where deliveries are followed by pickups; the VRP with Clustered Backhauls (VRPCB) [145], emphasizing the geographical clustering of pickup points; and the VRP with simultaneous pickups and deliveries (VRPSPD) [179] where pickup and delivery requests are met simultaneously. Researchers have also worked on models like the Split-delivery VRP (SDVRP) [58, 59] and the VRP with divisable deliveries (VRPDD) [139], which allows customer demands to be fulfilled through multiple deliveries, thereby increasing the flexibility and efficiency of vehicle utilization.

The exploration of VRPs has not stopped at time and demand scenarios. The field also embraces models catering to diverse operational complexities. The Heterogeneous VRP (HVRP) [82] considers vehicles of varying capacities, while the Multi-depot VRP (MDVRP) [36] addresses organizations operating from multiple depots. The Periodic VRP (PVRP) [70] extends the planning horizon to include recurring customer visits. Addressing uncertainties, the Stochastic VRP (SVRP) [76] models randomness in demand, customer presence, and travel times, further subcategorized into variants like the VRP with stochastic demand (VRPSD) [23] and the VRP with stochastic travel times (VRPSTS) [114]. The Open VRP (OVRP) [162] introduces flexibility by allowing vehicles to end their routes without returning to the depot. Other specialized variants, such as the Green VRP (G-VRP) [62], the Dynamic VRP (DVRP) [115], and the Multi-compartment

VRP (MC-VRP) [42], highlight the field's adaptability to environmental concerns, real-time information changes, and specific cargo handling needs.

As the importance of sustainable systems and operations increases, the VRP literature is naturally moving towards the emergence of specialized VRP variants such as the Electric Vehicle Routing Problem (EVRP) [43], the Two Echelon Vehicle Routing Problem (2E-VRP) [147], and the Two Echelon Electric Vehicle Routing Problem (2E-EVRP) [29]. The following sections will present the integration of electric vehicles into routing strategies, the complexities of multi-tier distribution systems, and the synergy between these advanced approaches, highlighting the field's progression towards sustainable and efficient logistics solutions.

1.2.1 The Electric Vehicle Routing Problems

The Electric Vehicle Routing Problem (EVRP) is a variant of the classic VRPs that consider using EVs for goods or service delivery. As mentioned before, EVs have gained popularity in logistics and transportation due to the growing concern for reducing greenhouse gas emissions and promoting environmental sustainability [116].

The EVRP seeks the most effective routing strategies for delivering goods or services using EVs from a central warehouse to a network of customers. The primary objective remains to minimize total costs, such as travel distance, time and operational costs, as in the case of conventional VRPs. However, energy efficiency is also sometimes considered to be an objective due to the nature of the problem. Another crucial aspect of the EVRP involves managing the limited range of vehicles and strategically planning for recharging at available stations. En-route charging requirements of EVs add a layer of complexity to route planning, as it necessitates not only efficient routing to demand points but also timely and effective battery recharging plans.

Figure 1.6 illustrates an EVRP network, showing a central depot, various customer locations, charging stations, and the planned routes of electric vehicles. In this model, electric vehicles must consider both customer demands and the availability of charging stations along their routes. An EV departs from the depot with a fully charged battery. As it travels through demand points, its battery is consumed depending on various factors such as the distance travelled, load and speed. Therefore, it has to visit charging stations to charge the battery when needed.

The route optimization of rechargeable vehicles was first considered by Conrad and Filiozzi [43]. They presented a mathematical formulation of the problem, aiming to minimize the number of vehicles and the total cost related

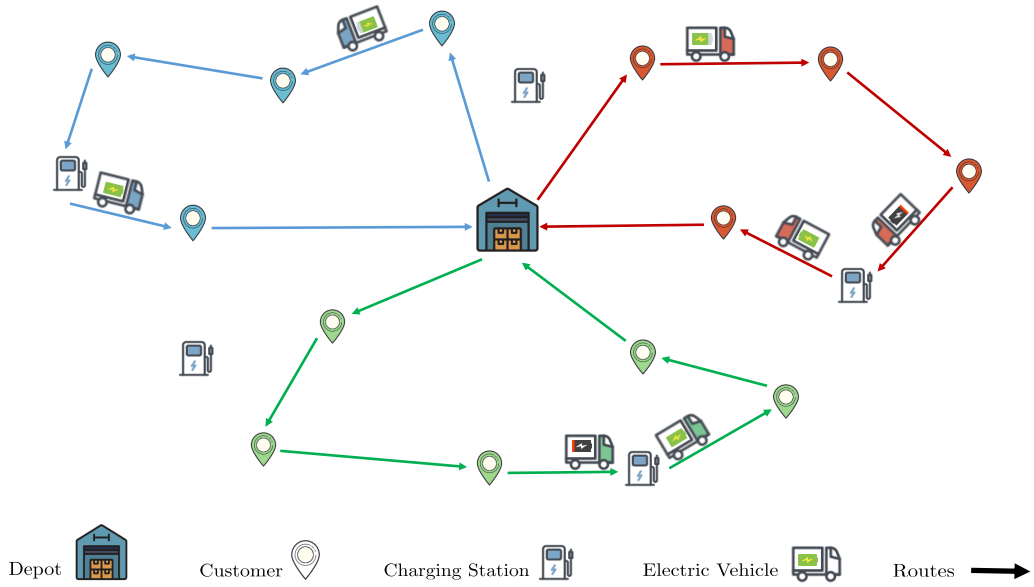


Fig. 1.6 Illustration of an Electric Vehicle Routing Problem. Icons by Icons8 (icons8.com).

to the traveled distance, the service time, and the vehicle recharging cost. After this pioneering work, several researchers presented EVRP variations together with different algorithmic solutions. Erdoğan et al. [62] formulated a green vehicle routing problem in terms of a mixed-integer programming model and proposed two contraction heuristics along with a customized improvement technique to solve large-sized problem instances. Schneider et al. [166] extended the EVRP by including time window constraints into the model. Moreover, they proposed a metaheuristic algorithm based on variable neighborhood search (VNS) and tabu search (TS). Felipe et al. [66] extended the green vehicle routing problem introduced by [62], considering several real-life assumptions such as partial recharging and multiple charging technologies with different charging speeds and costs. They developed a mathematical model and proposed several heuristic solution methods for problem instances of realistic size. In addition, Keskin and Çatay [104] introduced an EVRP with time windows and partial recharging. They proposed an adaptive large neighborhood search (ALNS) based solution approach and analyzed if the partial recharging of batteries improved the obtained solutions when compared to the full charging option. Furthermore, they also considered a fast charging option in [105] and proposed two different mathematical models as well as a matheuristic. Moreover, Montoya et al. [138] introduced a new model that takes into account the non-linear charging time of batteries. They reported that the time spent for charging batteries is non-linear, and ignoring this fact may cause the generation of infeasible and/or costly

solutions. Keskin et al. [106] considered also the fact that an EV might have to wait in a queue when visiting a charging station. They proposed a two-stage simulation-based heuristic using an LNS algorithm. Sadati and Çatay [160] recently introduced a multi-depot green vehicle routing problem and developed a mixed-integer linear programming model. They proposed a solution method based on VNS and TS and reported on the computational properties of the algorithm. Duman et al. [60] proposed exact and heuristic algorithms based on branch-and-price-and-cut and on column generation to solve the EVRP with TWs. Huerta-Rojo et al. [95] proposed an algorithm named ACOLS, a combination of ant colony optimization and local search, for the EVRP with time windows and partial recharging.

1.2.2 Two-Echelon Vehicle Routing Problems

Driven by an increasing focus on sustainable city logistics, the Two-Echelon Vehicle Routing Problem (2E-VRP) emerges as an extension of traditional VRPs. This variant is particularly relevant in the context of increasing urbanization and environmental awareness, as it introduces a multi-echelon distribution system designed to address the challenges of urban delivery networks [45, 147].

The 2E-VRP model structures the distribution process into two distinct levels. The first level — in other words, echelon — involves transporting goods from a central warehouse to intermediate transhipment facilities, known as satellites, typically located on the outskirts of urban areas. These satellites act as pivotal nodes for the collection and redistribution of goods. The transition of goods from the warehouse to satellites is generally handled by large trucks suitable for long-distance hauls and capable of carrying significant loads.

The second echelon shifts the focus to urban distribution challenges. At this stage, smaller vehicles, better suited to navigate city environments, take over to deliver goods from satellites to final customers within city centers. This phase is specifically designed to accommodate the constraints of city logistics, such as restricted entry points and narrow roads, while minimizing the environmental and social impact of delivery operations.

Figure 1.7 shows a typical 2E-VRP setup, illustrating the interaction between the central warehouse, satellite locations, and customer delivery points. Similar to the other variants of VRPs, the primary goal is to optimize transportation costs in both echelons while respecting the problem-related constraints.

After Crainic et al. [46] introduced the concept of two echelons in the context of the 2E-VRP as a new concept to the literature, this line of research developed into one of the most popular ones in the context of urban freight

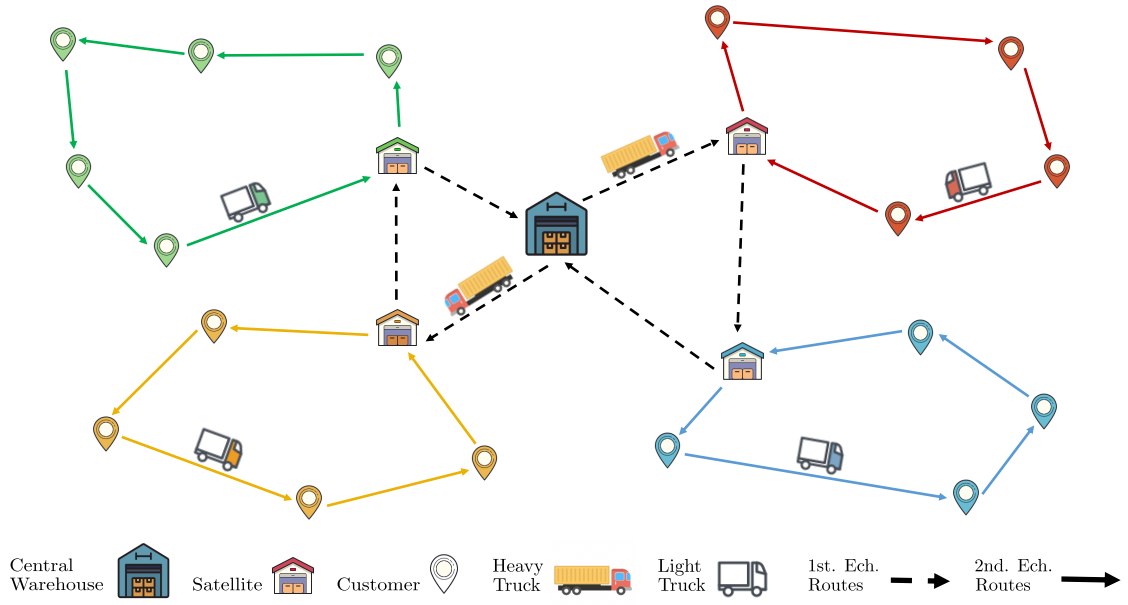


Fig. 1.7 Illustration of a Capacitated 2-Echelon Vehicle Routing Problem. Icons by Icons8 (icons8.com).

transportation [75]. The idea of developing sustainable cities and transportation systems further increases the interest in this field of research. Perboli et al. [147] proposed a mathematical model and math-based heuristics for the 2E-VRP. Various researchers proposed exact solution approaches such as branch-and-cut (see [100, 122]) and branch-and-price methods (see [51, 129]) as well as dynamic programming [14] to solve various extensions of the 2E-VRP. However, with growing instance size and problem complexity, researchers focused on approximate techniques to solve these problems. Grangier et al. [84] proposed a heuristic based on LNS for the 2E-VRP with time window and satellite synchronization constraints. Wang et al. [185] developed a matheuristic based on VNS and integer programming. Moreover, Belgin et al. [20] formulated the 2E-VRP with simultaneous pickup and delivery constraints as a two-index mixed-integer programming model and developed a hybrid metaheuristic combining variable neighborhood descent (VND) and local search.

1.2.3 Two-Echelon Electric Vehicle Routing Problems

The Two-Echelon Electric Vehicle Routing Problem (2E-EVRP) integrates the concepts of the 2E-VRP and the EVRP, as discussed in Sections 1.2.2 and 1.2.1 respectively. This hybrid model addresses the complexity of urban logistics by distributing goods through a two-tiered system that optimally utilizes both traditional large trucks and environmentally friendly EVs, each serving a specific logistical purpose.

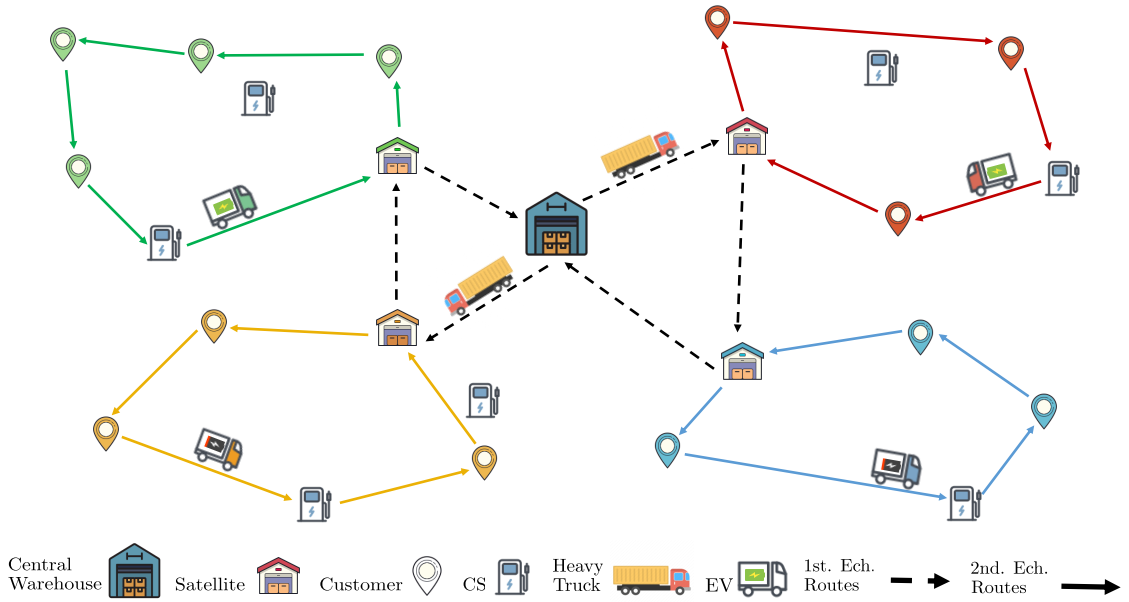


Fig. 1.8 Illustration of a Two-Echelon Electric Vehicle Routing Problem. Icons by Icons8 (icons8.com).

In the first echelon of the 2E-EVRP, goods are transported from a central warehouse to satellites using large conventional trucks. Transitioning to the second echelon, EVs, with their zero direct emissions and low noise levels, are deployed for the final delivery phase, bringing goods from the satellites to customers within urban areas. This approach aligns with the growing emphasis on reducing the environmental footprint of logistic activities especially in urban areas.

Figure 1.8 provides a visual representation of the 2E-EVRP setup that includes a central warehouse, satellites, charging stations and customers. It illustrates the flow of goods from the central warehouse to the satellites and then to the final customers, demonstrating the interplay between the conventional trucks in the first echelon and the EVs in the second echelon.

The 2E-EVRP introduces several unique challenges. In the first echelon, careful planning is essential for the efficient allocation of customers to satellites and the coordination of deliveries from the warehouse to these depots. The second echelon presents additional complexities, including the limited range of EVs, the necessity of accessible charging stations, and additional operational considerations like time windows for deliveries.

The objective of the 2E-EVRP is to minimize the total cost of the distribution process across both echelons. This includes a focus on factors such as travel distances, fleet costs, operational expenses, and, importantly, energy consumption. This holistic approach ensures that the distribution system is

not only operationally efficient but also minimizes its environmental impact, adhering to the principles of sustainable urban logistics.

The 2E-EVRP is regarded as a combination of two initially independent research lines: the one on EVRPs and the one on 2E-VRPs; see [43] and [46]. Works on EVRPS include [11, 113], while [172] is a recent example for work on 2E-VRPs. On the other hand, the related literature on 2E-EVRPs is still rather scarce. Jie et al. [101] were among the first to propose a 2E-EVRP with battery-swapping stations (BSS). A hybrid algorithm that combines column generation and LNS is proposed to solve the addressed problem. Breunig et al. [29] proposed a metaheuristic approach based on LNS and an exact mathematical programming algorithm that employs decomposition and pricing techniques to solve 2E-EVRP. Moreover, Cao et al. [32] investigated the design of a two-echelon reverse logistics network to collect recyclable waste utilizing a mixed fleet of electric and conventional vehicles. Instead of a single integrated mathematical model, the authors formulated the addressed problem as two distinct models, one for each echelon. Furthermore, Wu and Zhang [191] developed a branch and price algorithm to solve a 2E-EVRP. They tested the proposed solution approach on small and medium-sized instances containing up to 20 customers and two charging stations. Recently, Wang and Zhou [181] introduced a 2E-EVRP with time windows and battery-swapping stations. They developed a MILP model that minimizes transportation, handling, and fixed costs for the vehicles used in the first and second echelons, in addition to battery-swapping costs. However, the time spent on battery swapping is not considered.

1.3 Solution Approaches

Addressing EVRPs and 2E-VRPs presents significant challenges due to their complexity, including factors like vehicle capacity, customer demand, limited driving ranges of vehicles and en-route charging needs, and managing logistics across multiple distribution levels. The diversity of problem objectives and several constraints arising from real-life logistics scenarios further require tailored solution strategies.

This section provides an overview of various solution methods that have been used to solve VRPs in the past, ranging from exact algorithms to approximate techniques such as heuristics and metaheuristics.

1.3.1 Exact Solution Approaches

Exact solution approaches for VRPs are designed to identify an optimal solution by explicitly or implicitly exploring the complete space of valid solution. These methods are founded on mathematical rigor and incorporate advanced algorithmic techniques, including *Branch-and-Bound*, *Branch-and-Cut*, *Branch-and-Price*, and *Dynamic Programming*. Although these strategies ensure the identification of a best possible solution, their computational demand may render them impractical for larger or more complex VRP variants, such as the EVRP and the 2E-VRP.

Another technique, *Mixed Integer Linear Programming* (MILP), involves the addressed problem to be modeled with a linear objective function and linear constraints on both continuous and integer variables. This approach is widely favored for its applicability in constructing valid mathematical representations of combinatorial optimization problems. The advancement of computational tools, such as IBM ILOG CPLEX [8, 96] and Gurobi [87] has significantly enhanced the feasibility of solving problems formulated through MILP models.

Despite the precision these exact methods offer, their intensive computational requirements may limit their applicability to larger instances of complex or extensive VRP scenarios. The subsequent sections will present various exact solution approaches, detailing their methodologies, applications, and the challenges they face, especially in the context of complex routing problems.

1.3.1.1 Branch-and-Bound (B&B) Algorithms

The *Branch-and-Bound* (B&B) algorithm is a pivotal method in discrete optimization, especially suited for tackling integer and mixed-integer programming problems. B&B methodically explores the solution space through a tree of sub-problems, each node representing a part of the overall problem. This exploration is guided by two key processes: 'branching', which divides the problem into narrower sub-problems, and 'bounding', which establishes upper and lower limits for these sub-problems to eliminate unlikely candidates for optimal solutions.

The algorithm's effectiveness hinges on three core components: the *search strategy*, which determines the order in which subproblems in the tree are explored; the *branching strategy*, which determines the partitioning of the solution space; and the *pruning rules*, which help eliminate sub-optimal search areas.

Although B&B is theoretically capable of identifying optimal solutions, its computational demand escalates with the problem's complexity, posing a

significant challenge for large-scale instances of EVRPs and 2E-VRPs. The algorithm's computational feasibility often becomes strained as the number of sub-problems grows exponentially with the size of problem instances. Applications of B&B to VRPs can be found in [40, 179]

1.3.1.2 Branch-and-Cut (B&C) Algorithms

The *Branch-and-Cut* (B&C) method enhances the B&B by incorporating a "*cutting*" phase, which enhances the search process for finding optimal solutions in Integer Linear Programming (ILP) problems. Key to B&C's strategy is the generation of "*cuts*" or additional constraints, which are integrated to eliminate non-viable portions of the search space, focusing the search on feasible and promising areas. These cuts, derived from a polyhedral study of the problem's feasible region, specifically target and exclude fractional solutions from the linear programming relaxation of sub-problems, thereby tightening the LP relaxation and streamlining the path to the optimal solution [137].

However, implementing B&C is quite complex and computationally demanding. Generating effective cuts requires in-depth knowledge of the problem's polyhedral aspects and sophisticated algorithms for identifying impactful cuts. This complexity, combined with the significant computational resources needed, especially for large-scale VRP instances, limits B&C's practicality in certain scenarios.

Despite these challenges, B&C has demonstrated notable success in some applications, such as solving the two-echelon capacitated vehicle routing problem with grouping constraints more efficiently than general-purpose solvers like CPLEX [122]. Additionally, it has been used to address the GVRP by combining it with simulated annealing heuristics to achieve optimal solutions within reasonable time frames [199].

1.3.1.3 Branch-and-Price (B&P) Algorithms

The *Branch-and-Price* (B&P) method is a combination of B&B with column generation techniques. It starts by addressing a restricted version of the original problem, focusing on a manageable subset of variables, or columns. In VRP contexts, each of these variables often represents a potential vehicle route, with the initial set comprising a limited selection of feasible routes [164].

The strength of B&P lies in its dynamic column generation during the bounding process. New variables are systematically added based on their potential to enhance the current solution. This is achieved through the resolution

of a pricing problem, which seeks out routes with negative reduced costs, indicative of their ability to improve the incumbent solution. When fractional solutions arise, B&P branches to create sub-problems [65].

Unlike B&B and B&C, B&P introduces new variables only as necessary, thereby managing the problem's scale and complexity more effectively. This selective variable inclusion keeps the overall problem size manageable, enhancing computational performance. However, the method requires accurate management of both the pricing problem for route generation and the master problem for adjusting dual variables. The challenge is maintaining a balance in the number of generated columns, introducing neither too many nor too few at each stage. [133].

In recent VRP applications, B&P has demonstrated significant adaptability and efficiency. It has been successfully employed in solving complex EVRP and 2E-EVRP variants. By leveraging parallel column generation, B&P algorithms have shown a remarkable ability to handle the computational demands of these problems, often outperforming competing solvers in specific instances [34, 191]. Additionally, in addressing the EVRP with time windows, the combination of B&P with cutting-plane techniques has been instrumental for solving large-scale instances that were previously unresolved [60].

1.3.1.4 Dynamic Programming (DP)

Dynamic Programming (DP) is a cornerstone technique in operations research, particularly distinguished for its applicability to complex decision-making problems that exhibit overlapping sub-problems and a recursive structure [21]. The essence of DP lies in its systematic approach to problem-solving: it decomposes a large problem into smaller, manageable sub-problems, solves each of these once, stores the results, and reuses these solutions to address overarching problems.

Despite its efficiency across various combinatorial optimization problems, DP has limitations, particularly when applied to complex and large-scale problems such as EVRPs. These limitations primarily arise from the exponential increase in computational and storage demands associated with the rising dimensionality of the problem. Such growth can make DP impractical for large EVRP instances.

Within the context of EVRPs, DP is generally employed to identify the optimal strategy for EV charging station visits. Hiermann et al. [92] demonstrate this application through the use of a bi-directional labeling algorithm to address the Elementary Shortest Path Problem with Time Windows and Recharging Stations. This methodological approach is also echoed in the works of Roberti et al. [155]

and Desaulniers et al. [53], further underscoring the adaptability of DP in EVRP solutions. Additionally, Küçükoğlu et al. [112] illustrate how DP can be integrated within a hybrid simulated annealing and tabu search framework to efficiently generate charging operation plans

1.3.2 Construction Heuristics

Construction heuristics are simple, easy-to-implement algorithms for generating feasible (although not necessarily optimal) solutions to VRPs. They are typically used as a first step to obtain an initial solution, which can then be refined using local search or other optimization techniques.

Construction heuristics offer a more practical approach to solving VRPs, especially when dealing with large datasets or intricate constraints characteristic of EVRPs and 2E-VRPs. These heuristics build a solution incrementally, starting from an initially empty solution and gradually adding elements to it. Techniques such as the Greedy Heuristic, the Clarke-Wright Savings Algorithm, and the Sweep Algorithm and Insertion Heuristics are typical examples [173]. While they generally do not guarantee an optimal solution, construction heuristics are valued for their simplicity, speed, and ability to provide good quality solutions within a reasonable timeframe. This subsection explores various construction heuristics, their strategic application in VRP variants, and how they balance solution quality with computational efficiency.

1.3.2.1 Greedy Heuristic (Nearest Neighbor Algorithm)

This heuristic is a simple approach where the idea is to serve the closest unvisited customer next. Starting from the depot, the algorithm selects the nearest unvisited customer, adds it to the current route, and repeats this process, within loading capacity limits, until all customers are visited. A vehicle can usually only serve a subset of the customers due to capacity constraints. When the vehicle reaches its capacity or cannot accommodate any additional load, it returns to the depot to complete its tour. The process continues with the vehicle starting a new tour from the depot, targeting the next set of unvisited customers, until all customers have been served. The solution construction process concludes when the vehicle returns to the depot and no unvisited customer remains [159]. This approach is straightforward and easy to implement, but it tends to generate suboptimal solutions, especially for larger problems, as it does not consider the overall route configuration and may lead to routes that are considerably longer than necessary.

1.3.2.2 Clarke-Wright Savings Algorithm

Proposed by Clarke and Wright in 1964 [41], this algorithm calculates the "savings" for each pair of customers, which is defined as the cost saved by serving both customers on the same route rather than on separate routes. The algorithm then merges routes in decreasing order of savings, subject to capacity and other constraints. The Clarke-Wright Savings Algorithm is one of the most well-known heuristics for VRPs, but it tends to perform better on problems with rather symmetric costs.

1.3.2.3 Sweep Algorithm

Developed by Gillett and Miller [78], this heuristic is based on the geometric interpretation of the problem. It first orders the customers according to the angle they make with respect to the depot and a fixed axis. Then it "sweeps" a line around the depot, starting a new route whenever adding the next customer would exceed the vehicle's capacity. The Sweep Algorithm is intuitive and particularly useful when customers are dispersed in a circular manner around the depot, but it may perform poorly for other types of customer distributions [173].

1.3.2.4 Insertion Heuristics

These algorithms start with initial routes containing single customers and iteratively insert the unvisited customers into the routes where they cause the smallest increase in cost. Different versions exist, such as cheapest insertion (which minimizes the cost increase), farthest insertion (which inserts the farthest unvisited customer), and nearest insertion (which inserts the nearest unvisited customer). Insertion heuristics are more flexible than the Greedy and Savings algorithms and can perform well for various problem types, but they generally still lead to sub-optimal solutions and may not perform well for larger problem instances.

1.3.2.5 Randomized Heuristics

These algorithms combine deterministic construction rules with random choice, aiming to explore different parts of the solution space and avoid the basin of attraction of local optima. They often involve constructing multiple solutions and choosing the best one. Examples include randomized greedy algorithms [154], which select the next customer to serve with a probability that depends on the solution quality, and randomized insertion algorithms, which randomly select the next customer to insert. Randomized heuristics can often find good

solutions to complex problems, but they require careful tuning of their degree of randomness.

1.3.2.6 Two-Phase Heuristics

These algorithms, such as those proposed in [18], first partition the customers into clusters (phase one) and then determine the visiting order within each cluster (phase two). The *Route-First-Cluster-Second* heuristic constructs routes first and then assigns them to vehicles, while the *Cluster-First-Route-Second* heuristic first clusters the customers and then creates a route for each cluster. Two-phase heuristics can be quite effective, especially for larger problem instances, but their performance is usually sensitive to the method used to partition the customers and sequence the visits.

1.3.3 Metaheuristics

Metaheuristics [142] represent a sophisticated class of solution approaches, particularly effective for large-scale instances of complex VRPs such ass the EVRP and 2E-VRP. These methods are designed to explore the solution space extensively, employing strategies to escape from local optima. Techniques under this umbrella include Simulated Annealing, Tabu Search, Genetic Algorithms, and Ant Colony Optimization, each with unique mechanisms to balance the exploration and exploitation of the search process. *Metaheuristics* are favored for their flexibility and effectiveness in finding high-quality solutions to complex problems where exact methods are infeasible, and simple heuristics may fall short. This subsection delves into various metaheuristic techniques, analyzing their suitability for different VRP scenarios and highlighting their role in advancing the field of route optimization.

1.3.3.1 Simulated Annealing

Simulated Annealing (SA), introduced by Kirkpatrick et al. [107], is a metaheuristic inspired by the annealing process in metallurgy. This process involves heating and controlled cooling of materials to minimize structural defects. In the context of combinatorial optimization problems, SA starts with an initial feasible solution, often generated heuristically, and explores the solution space through random moves (local search operators). Its defining feature is the probabilistic acceptance of solutions with inferior objective function values compared to the current solution, based on a 'temperature' parameter. This parameter is initialized with a high value facilitating the occasional acceptance of sub-optimal solutions,

and gradually decreases over time. This reduction diminishes the likelihood of accepting solutions worse than the current one. This mechanism allows SA to escape from local optima, enabling a comprehensive exploration of the solution space.

A critical aspect of implementing SA in VRPs is the calibration of the temperature parameter. Precise calibration is essential, as rapid cooling can lead to premature convergence to sub-optimal solutions, while slow cooling might require excessive computational time.

SA has been widely applied in the context of combinatorial optimization problems, including various VRP variants [111]. However, within the scope of EVRPs and their variants, SA is often hybridized with other metaheuristics [121, 146, 166].

1.3.3.2 Tabu Search

Tabu Search (TS) [79] is another example of a formidable metaheuristic framework for addressing complex optimization problems. This method is distinguished from conventional heuristic approaches through its innovative use of memory structures to enhance problem-solving efficiency. More specifically, TS explores the solution space employing a *tabu list* to maintain a short-term memory of attributes of previously evaluated solutions. This mechanism prevents cyclic exploration and assists in overcoming local optima by prohibiting the algorithm from revisiting recently seen solutions.

Central to TS's methodology are two pivotal strategies: intensification, achieved by always moving to the best non-tabu neighbor of the current solution, and diversification, achieved by determining a suitable tabu list length [80].

One of the most notable strengths of TS is its adaptibility, allowing it to tackle complex and dynamic problem landscapes effectively. This adaptability makes TS particularly suitable for VRPs, where the solution space is often rugged with a huge number of local optima. However, the success of TS is heavily reliant on the careful configuration of its memory structures and strategic elements. Determining the optimal length of the *tabu list* and the right balance between intensification and diversification can be challenging and typically necessitates empirical fine-tuning.

The literature includes publications that present both standalone [1, 186] and hybrid implementations of TS. For example, hybrid approaches incorporate methods like Large Neighborhood Search with TS for solving variants of the EVRP [163].

1.3.3.3 Variable Neighborhood Search

Introduced by Hansen and Mladenović, [89] *Variable Neighborhood Search* (VNS) is –like TS– a *trajectory-based* metaheuristic approach. VNS is grounded in the principle that a local optimum concerning one neighborhood structure is not necessarily a local optimum for another one. It operates on the premise that the global optimum is a local optimum relating to all possible neighborhood structures. Therefore, unlike other local search algorithms, VNS employs multiple neighborhood structures rather than relying on a single one to avoid getting stuck in local optima.

The search process starts with an initial solution and iteratively explores neighborhoods of increasing distance to escape from local optima. This involves two main phases: *local search* and *shaking*. The local search phase intensifies the focus on promising areas of the search space, while the shaking phase expands the search to include more diverse and distant neighborhoods. This approach allows VNS to dynamically adapt its search strategy, balancing between intensification and diversification [88].

The algorithm's success largely depends on the strategic definition and adaptation of neighborhood structures, which can be intricate and specific to the problem at hand. The literature on VRP underscores VNS's adaptability and its capability to address large-scale and dynamic problems, making it a preferred method for these types of problems [99]. Furthermore, in EVRPs, VNS stands out as one of the most frequently utilized solution approaches [113].

1.3.3.4 Large Neighborhood Search

Large Neighborhood Search (LNS) [171] is a powerful metaheuristic that is particularly good at dealing with optimization problems for which large neighborhoods can be defined.

One of the most successful LNS variants in the VRP field is the one based on the partial destruction of the incumbent solution. Its key components are the utilized *destruction* and *repair* mechanisms. The process begins with an initial solution, possibly obtained through a heuristic approach. Then, the *destruction* phase involves systematically removing a part of the solution. For example, in VRPs, this could involve removing a group of customers from their planned routes or getting rid of entire routes. The subsequent *repair* phase involves reconstructing the destroyed solution by employing heuristics or exact methods. This iterative process enables LNS to explore broader solution landscapes, avoiding being stuck in local optima [148, 149].

LNS and its variants such as Adaptive LNS (ALNS) [158], are highly effective and adaptable metaheuristics that are well-suited for addressing complex optimization problems. Their application to VRPs shows their tremendous capacity for tailored problem-solving. LNS and ALNS have become very popular and successful metaheuristics in the field of many combinatorial optimization problems including EVRPs [128].

1.3.3.5 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are inspired by the principles of natural evolution and genetics [12]. EAs operate on a population of potential solutions, with the quality of each solution evaluated using a fitness function. The algorithms evolve the population over generations, using operators inspired by biological mechanisms such as selection, crossover (recombination), and mutation. The goal is to evolve solutions of increasing quality over time, ideally converging towards a global optimum or, at the very least, a high-quality solution.

Genetic algorithms (GAs) are the most renowned under the broader umbrella of EAs. Introduced by Holland in the early 1970s, GAs have since been widely applied to solve complex optimization problems, including VRPs [94]. GAs explicitly borrow concepts from biological genetics, operating on 'chromosomes' that represent solutions and 'genes' that represent solution components. The algorithm uses mechanisms such as 'crossover' (combining genes from two parent solutions to create offspring) and 'mutation' (randomly altering gene values) to explore the search space.

Several studies have been conducted on the use of GAs for VRPs. Notable among them is the work of [150], who proposed a GA with a specific crossover operator for VRPs with time windows. Moreover, some of the applications of GAs to EVRPs can be found in the following studies [102, 170, 198]

1.3.3.6 Particle Swarm Optimization

Particle Swarm Optimization (PSO), inspired by the social behavior of birds and fish, represents a swarm intelligence metaheuristic in the realm of optimization algorithms [103]. PSO involves a swarm of particles navigating through the solution space. Each particle embodies a potential solution characterized by its position and a velocity. These particles adjust their trajectories based on individual and collective experiences, effectively sharing information to guide their search. The standard PSO algorithm combines elements of both the particle's best-known position and the global best among the swarm, with a degree of

randomness to maintain diversity [13].

Like all other metaheuristic algorithms, PSO does not require gradient information. Its way of working is designed to avoid being trapped in local optima, making it highly effective for problems with complex landscapes. Moreover, its flexibility allows for easy adaptation to different types of optimization problems. However, PSO can face challenges, particularly in high-dimensional spaces where it may struggle with premature convergence or fail to explore the search space adequately. This limitation necessitates careful tuning of parameters and potentially hybridizing PSO with other algorithms to enhance its exploration capabilities [73].

Despite its origins in continuous optimization, PSO has been adapted for discrete problems such as VRPs in several studies [37, 196, 197]. Typically, these adaptations involve treating the search space as continuous and then discretizing the solutions, translating real-value positions into integer values to obtain routes. However, this approach only partially captures the unique aspects of the discrete nature of VRPs. To address this gap, Gong et al. [83] developed a discrete variant of PSO specifically for the VRP with time windows.

1.3.3.7 Ant Colony Optimization

Ant Colony Optimization (ACO) is a swarm intelligence algorithm inspired by the short path-finding behavior of ants. When searching for food, ants use pheromone trails to locate the short paths between food sources and their nest. ACO algorithms use this concept of stigmergic communication to discover efficient solutions in a decentralized approach by altering the environment [24, 57].

In ACO, each "ant" produces a potential solution within the problem space, marking its path with pheromone signals that guide subsequent ants towards promising solutions. ACO learns iteratively by updating the pheromone levels based on the solution's quality. By adopting a probabilistic construction of solutions, guided by both pheromone levels and heuristic information, ACO balances the exploration of new areas and the exploitation of known paths effectively [24].

Some of the successful implementations of ACO to EVRPs can be found in the following papers [130, 131, 195].

1.3.3.8 Hybrid Approaches

The development of optimization techniques has witnessed an exponential growth over recent decades, marked by an ever-increasing importance of heuristic and metaheuristic approaches. These methods range from single-solution-based algorithms to Swarm Intelligence and Evolutionary Algorithms. In addition, in the last 20 years, a notable shift has been observed towards hybrid metaheuristics, which integrate various algorithmic strategies to exploit the synergistic benefits of different approaches. This evolution is grounded in recognising the fact that no single optimization strategy can universally outperform others across all problem domains, as underscored by the No Free Lunch theorems [189]. Hybrid metaheuristics, therefore, aim to combine diverse algorithmic elements to enhance solution quality and efficiency [153].

Hybrid metaheuristics can be broadly categorized into two types: those involving the exchange of algorithmic components among different metaheuristics and those integrating metaheuristics with operations research (OR) or soft-computing and artificial intelligence (AI) techniques. The latter aims to leverage the strengths of classical methods in conjunction with metaheuristic strategies [26].

Within the EVRP domain [192], several studies exemplify the first category's approach. Schneider et al. [166] developed a hybrid metaheuristic that combines VNS and TS algorithms with a dynamic punishment mechanism for the EVRP-TW. Similarly, Schneider et al. [167] introduced an Adaptive VNS that merges VNS with an Adaptive LNS for the VRP with intermediate stops. Felipe et al. [66] solved the EVRP model with partial recharges by combining SA and iterated local search (ILS) algorithms. Hiermann et al. [92] addressed large-scale instances of EVRP-TW and recharging stations by combining adaptive large-scale neighborhood search with ILS. Keskin and Çatay [104] utilized the solution acceptance mechanism of simulated annealing within the Adaptive LNS framework for EVRP with partial recharges. Shao et al. [170] enhanced local search effectiveness in a hybrid GA by incorporating GA with local search strategies, crossover, mutation, and neighborhood-based gene modifications. Hiermann et al. [93] refined this approach with a hybrid genetic algorithm (HGA) for routing optimization, emphasizing local search for solution intensification and large neighborhood search as a mutation operator . Cortés-Murcia et al. [44] employed a hybrid approach that leverages VNS and ILS techniques for solving the EVRP-TW with partial recharges and satellite customers. Wang et al. [187] integrated the Clarke-Wright Savings Algorithm, the Sweep Algorithm, and the Multi-Objective PSO to solve the multi-depot green VRP.

Concerning the second category, combining metaheuristics and exact algorithms—known as matheuristics—have shown promise. This approach hybridizes exact solution algorithms with metaheuristics or employs exact optimization solvers within metaheuristic frameworks, merging the robustness of mathematical programming with the efficiency of metaheuristic algorithms. Froger et al. [71] developed a two-stage matheuristic for solving the EVRP with a nonlinear charging function and capacitated stations, starting with a pool of routes generated by ILS and then assembling these routes using a B&B algorithm. Çatay and Keskin [105] combined ALNS with an exact method for the EVRP-TW and fast recharges. Bruglieri et al. [30] proposed a three-phase matheuristic for the Time-effective EVRP with partial recharges, utilizing Mixed Integer Linear Programs in the first two phases to generate feasible solutions, followed by a VNS local Branching (VNSB) algorithm.

1.4 CMSA: Construct-Merge-Solve-Adapt

Despite the remarkable advancements in exact solution techniques such as DP and mathematical programming—think of popular, high-performing tools such as CPLEX[1] and Gurobi[2]—solving complex combinatorial optimization problems using pure exact techniques still may require overly long computation times, especially when considering large-sized problem instances. In those cases in which exact techniques and general-purpose solvers fail to deliver good enough solutions within a reasonable computation time, approximate techniques are used to obtain such solutions in much lower computation times. To take advantage of exact solvers, also in the context of large-sized problem instances, recent years have seen a sharp increase in the number of hybrid algorithms that combine approximate techniques and exact methods and tools. As explained in the previous section in detail, the resulting algorithms are often called *hybrid metaheuristics* or *matheuristics* [67]. One of the recent hybrid algorithms in this direction is "Construct, Merge, Solve & Adapt" (CMSA) [27].

At its core, the CMSA algorithm operates on the principle of reducing the original problem instance to smaller, more manageable sub-instances that potentially contain high-quality solutions to the original problem instance. This reduction enables the application of exact mathematical programming solvers to these sub-instances with a significantly reduced computational effort. In this line, CMSA consists of four main steps. Initially, a set of solutions is probabilistically

[1] `https://www.ibm.com/analytics/cplex-optimizer`
[2] `http://www.gurobi.com/`

constructed using, for example, heuristic methods. Next, components of these solutions are then merged to form a sub-instance. In other words, the search space is reduced before the exact solver is applied. Then, the exact solver is applied to possibly find a best solution to the current sub-instance. Finally, the sub-instance is adapted based on this solution, ensuring the algorithm's iterative refinement, leveraging feedback from the solver's output to inform future iterations, and progressively enhancing the solution quality. The algorithm then proceeds with the subsequent iteration until the stopping condition is met.

The CMSA algorithm has demonstrated its versatility and effectiveness across a multitude of combinatorial optimization problems. Notable applications include, but are not limited to, the maximum happy vertices problem [118], cooperative route planning for air-ground robots [10], and the strategic refueling and maintenance planning for nuclear power plants [61]. These implementations show CMSA's adaptability to various contexts, from logistics and transportation to energy management and software testing, underlining its utility in solving real-world problems with complex constraints and requirements.

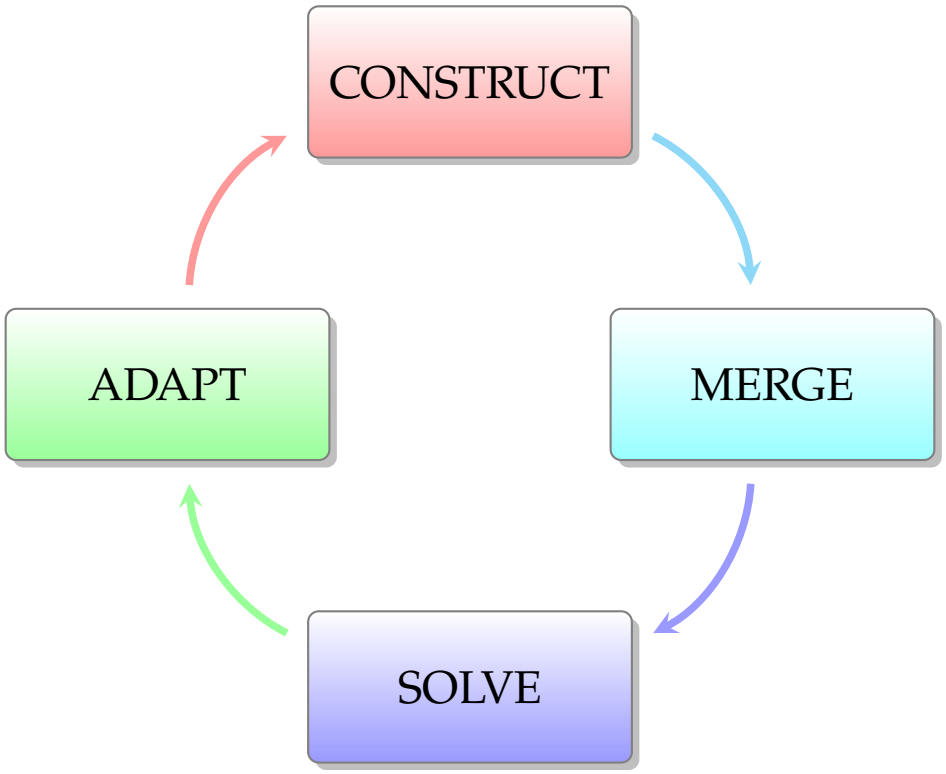


Fig. 1.9 CMSA Framework

Recent studies have further expanded the applicability of CMSA, addressing challenges such as taxi sharing optimization [22] and the multidimensional multi-way number partitioning problem [55]. These improvements show how the CMSA algorithm continues to evolve and adjust to work well across different

types of problems, confirming its importance as a key method in the range of hybrid metaheuristic strategies.

1.5 Thesis Contributions

This thesis is established at the intersection of two pivotal research areas within the field of combinatorial optimization, explicitly focusing on the algorithmic advancement of the solution approaches and the development of comprehensive routing models for sustainable urban logistics. These dual research pathways significantly contribute to both the theoretical framework and the practical applications of the algorithms for the route optimization of logistics distribution, particularly in contexts that prioritize environmental sustainability.

The algorithmic contributions primarily involve enhancing and applying the CMSA algorithm, particularly through the development of a self-adaptive variant known as Adapt-CMSA. This variant was developed in response to the recognized challenge of parameter sensitivity in metaheuristics, where performance for specific instances strongly depends on parameter settings. The introduction of Adapt-CMSA aims to reduce this sensitivity, ensuring robust performance across various problem sizes and complexities without the need for specific parameter re-tuning. This adaptation is crucial as it allows for a more consistent application across different scenarios, which was particularly evident in its application to the Minimum Positive Influence Dominating Set problem, where it outperformed the standard CMSA by adjusting its parameters dynamically, thereby enhancing efficiency and scalability.

From the modelling and application perspective, the thesis addresses the formulation of complex routing problems that reflect real-world scenarios in sustainable urban logistics. It introduces models for EVRPs and 2E-EVRPs, integrating critical constraints such as time windows, simultaneous pickups and deliveries, and scenarios allowing partial battery charging. Additionally, the thesis goes beyond traditional distance minimization objective functions to the one considering energy-minimization. Primarily employing CMSA and Adapt-CMSA, along with a range of heuristic and metaheuristic approaches like constructive heuristics, VNS, and ACO, the thesis effectively addresses and solves the addressed complex EVRP and 2E-EVRP variants.

In this line, the key contributions can be summarized as follows:

- The first major contribution of this work is the development of Adapt-CMSA. This advancement was inspired by our initial application of standard CMSA to the *Minimum Positive Influence Dominating Set* (MPIDS) problem—an

NP-hard yet comparatively simpler problem than EVRPs and 2E-EVRPs. This initial study allowed for an in-depth analysis of CMSA's performance and behavior on a simpler combinatorial optimization problem. The experimental analysis revealed that while standard CMSA outperformed existing metaheuristics and was competitive with CPLEX for small to medium-sized instances, its performance degraded on the largest problem instances. In these cases, even the generated sub-instances were too large for the ILP solver to solve efficiently within the allocated time. This highlights a critical limitation in scalability and parameter sensitivity. This observation led to the conceptualization of Adapt-CMSA, designed to dynamically adjust its parameters, thus reducing the need for extensive parameter tuning across different problem sizes. The development of Adapt-CMSA directly addressed these scalability issues, demonstrating enhanced performance and robustness compared to the standard version. In the experimental study on the MPIDS problem, Adapt-CMSA not only alleviated the limitations observed with standard CMSA but also showed remarkable adaptability to various instance sizes without requiring specific parameter adjustments. This feature makes it particularly suitable for more complex routing problems such as EVRPs and 2E-EVRPs, as evidenced by its subsequent applications across different variants of these problems throughout the thesis.

- Our second major contribution involves the application of Adapt-CMSA to the *Electric Vehicle Routing Problem with Time Windows, Simultaneous Pickup and Delivery, and Partial Battery Charging* (EVRP-TW-SPD-PR). Two distinct versions of Adapt-CMSA have been introduced. Both versions differ from each other in terms of the solution and sub-instance representation and in terms of the ILP model used for solving the sub-instances. The first version, referred to as Adapt-CMSA-STD, employs an assignment-type ILP model, which represents solutions as lists of edges between nodes. This model often struggles to provide good lower bounds [9]. The computational experiments also showed that it may fail to find feasible solutions within reasonable computation times, even for smaller sub-instances. In response to these limitations, we introduced a second variant of Adapt-CMSA, utilizing a set-covering ILP model (Adapt-CMSA-SETCOV). This model represents solutions as sets of valid vehicle tours, aiming to select the most promising tours from a set of feasible tours to cover all customer demands. Furthermore, both variants made use of two construction heuristics based on the Clarke-Wright Savings algorithm and the sequential

insertion algorithm to generate solutions. Comparative analyses showed that Adapt-Cmsa-SetCov consistently outperforms Adapt-Cmsa-Std in terms of best and average solution quality. To validate the robustness of our approach, Adapt-Cmsa-SetCov was also benchmarked against various VNS variants from existing literature on a reduced version of the addressed problem, the EVRP-SPD. The results demonstrated the superior performance of Adapt-Cmsa-SetCov across all evaluated problem instances.

- The third major contribution addresses the *Two-Echelon Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery* (2E-EVRP-SPD). This problem introduces a novel challenge by integrating simultaneous pickup and delivery operations within the two-echelon distribution framework, a complexity yet to be fully explored in the existing literature. The problem is formulated both by an assignment-type MILP model and a set-covering-based MILP model. Given its complexity, traditional solvers like CPLEX struggle even for smaller instances. To tackle this, Adapt-CMSA, which utilizes the set-covering-based MILP model for solving sub-instances, was employed to solve the addressed problem. Probabilistic versions of two construction heuristics based on Clarke-Wright Savings and sequential insertion algorithms were also implemented.

- The fourth key contribution centres on the application area of the *Two-Echelon Electric Vehicle Routing Problem with Time Windows* (2E-EVRP-TW). This contribution entails the development of two distinct versions of VNS. A node-based three-index MILP model was initially formulated to precisely define the problem. Despite the ability of CPLEX to handle smaller instances, the complexity of larger scenarios necessitated a more robust and efficient solution approach. The proposed VNS algorithms utilize an initial solution generation method based on the Clarke-Wright Savings Algorithm adapted to the 2E-EVRP-TW's problem-depended characteristics. Extensive neighbourhood search techniques also complemented this approach. In addition to the classical shaking and local search operators, the algorithm utilizes LNS operators known as "destroy and repair", resp. "removal and insertion", to enhance the performance of VNS. Additionally, to support the empirical evaluation, new problem-specific benchmark sets were generated, addressing the gap in available resources for this complex problem.

- The final significant contribution of this thesis is the development and analysis of the *Electric Vehicle Routing Problem with Road Junctions and Road*

Types (EVRP-RJ-RT). This novel problem variant extends the traditional scope of EVRPs by incorporating real-world complexities in the logistics network, such as varied road types and additional road junctions that impact electric vehicles' routing and energy consumption. Recognizing the environmental impact of road transport, this study not only enhances the realism of routing models but also contributes directly to sustainable urban logistics by optimizing the energy consumption of electric vehicles.

To effectively tackle this complex problem, we have formulated the EVRP-RJ-RT as a Mixed Integer Nonlinear Programming (MINLP) model, accommodating unique constraints that reflect the complexity of urban traffic and road layouts. To solve the EVRP-RJ-RT, two heuristic approaches were utilized. Initially, a modified Clarke-Wright Savings method was employed to construct initial feasible routes efficiently. Following this, an Ant Colony Optimization algorithm based on the MAX-MIN Ant System was applied to refine these solutions and explore the solution space more thoroughly. This metaheuristic approach not only demonstrated substantial improvements in solution quality over basic heuristics but also showed promise in handling the nonlinear aspects of the problem related to energy consumption and route feasibility in complex urban settings. Additionally, we have developed a set of new problem instances specifically designed to test the efficiency of the proposed solution approaches.

1.6 The Organization of This Thesis

This thesis report is organized as follows:

- Chapter 1 provides the foundational background and sets the stage for the ensuing discussion. It begins with a detailed overview of VRPs, emphasizing the distinctive challenges and characteristics of EVRPs and 2E-VRPs, as well as their amalgamation into the 2E-EVRPs. The chapter then outlines a range of existing solution approaches for VRPs, including exact solution methodologies and various heuristic and metaheuristic strategies, culminating in a discussion on hybrid approaches. This exposition lays the groundwork for understanding the complexities and nuances of the VRPs tackled in this thesis.

- Chapter 2 describes the algorithmic framework of CMSA and its self-adaptive variant, Adapt-CMSA, contextualized by means of binary integer linear programming (ILP) models of CO problems. This chapter

establishes a foundational understanding of the algorithmic strategies employed in later chapters, providing a general description of the methods and their application to combinatorial optimization problems modelled as binary ILPs, a category encompassing numerous NP-hard problems.

- Chapter 3 shows the application of CMSA and Adapt-CMSA to the Minimum Positive Influence Dominating Set (MPIDS) problem, offering insights into the comparative advantages of Adapt-CMSA over standard CMSA. This chapter also reflects on the broader applicability of Adapt-CMSA in various problem contexts, supported by a detailed experimental analysis.

- Chapter 4 focuses on applying Adapt-CMSA to the Electric Vehicle Routing Problem with Time Windows, Simultaneous Pickup and Delivery, and Partial Battery Charging (EVRP-TW-SPD-PR), a complex problem variant incorporating real-world constraints. The chapter discusses the development of two Adapt-CMSA variants, their performance comparison, and the challenges of solving such intricate problems with conventional methods.

- Chapter 5 extends the application of Adapt-CMSA to the Two-Echelon Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery (2E-EVRP-SPD), a problem characterized by its multi-tier distribution network. The chapter provides a comprehensive analysis of Adapt-CMSA's effectiveness in this context, juxtaposing it against both exact and heuristic solution methods.

- Chapter 6 explores the application of a VNS algorithm to the Two-Echelon Electric Vehicle Routing Problem with Time Windows (2E-EVRP-TW), presenting an innovative approach to a complex routing problem. The chapter offers insights into the algorithm's effectiveness and its comparative performance against other solution methods.

- Chapter 7 introduces an enhanced EVRP model (EVRP-RJ-RT), incorporating real-world complexities like road junctions and road types. The chapter details the application of construction heuristics and ACO to tackle this advanced model, highlighting the efficacy of these methods in managing the added layers of complexity.

- Chapter 8 concludes the thesis with a comprehensive summary of the findings and contributions, followed by an outlook section that sketches

the potential future directions and applications of the research presented. This chapter provides a holistic view of the thesis, tying together the various strands of research and suggesting pathways for further exploration in the field.

1.7 Publications Derived from this Thesis

Most of the results presented in this thesis have already been published, as indicated in the list below.

1. **Mehmet Anıl Akbay** and Christian Blum. Application of CMSA to the Minimum Positive Influence Dominating Set Problem. In *Proceedings of the 23rd International Conference of the Catalan Association for Artificial Intelligence, CCIA-2021, IOS Press*, Artificial Intelligence Research and Development, pages 17–26, 2021. (`https://doi.org/10.3233/FAIA210112`).

2. **Mehmet Anıl Akbay**, Albert López Serrano, and Christian Blum. A Self-Adaptive Variant of CMSA: Application to the Minimum Positive Influence Dominating Set Problem. In *International Journal of Computational Intelligence Systems*, vol 15, number 1, pages 1–13, 2022. (`https://doi.org/10.1007/s44196-022-00098-1`). [Impact Factor 2022: 2.9 (JCR)]

3. **Mehmet Anıl Akbay**, Can B. Kalayci, Christian Blum, and Olcay Polat. Variable Neighborhood Search for the Two-Echelon Electric Vehicle Routing Problem with Time Windows. *Applied Sciences*, vol 12, number 3, pages 1014, 2022. (`https://doi.org/10.3390/app12031014`). [Impact Factor 2022: 2.7 (JCR)]

4. **Mehmet Anıl Akbay**, Can B. Kalayci, Christian Blum. Application of CMSA to the Electric Vehicle Routing Problem with Time Windows, Simultaneous Pickup and Deliveries, and Partial Vehicle Charging. In *Proceedings of the Metaheuristics International Conference, Lecture Notes in Computer Science*, pages 1–16, 2022. (`https://doi.org/10.1007/978-3-031-26504-4_1`).

5. **Mehmet Anıl Akbay**, Can B. Kalayci, Christian Blum. Application of Adapt-CMSA to the Two-Echelon Electric Vehicle Routing Problem with Simultaneous Pickup and Deliveries. *Proceedings of EvoCOP 2023 – 23rd European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoSTAR), Lecture Notes in Computer Science*, pages 16-33, 2023. (`https://doi.org/10.1007/978-3-031-30035-6_2`). [CORE B conference]

6. **Mehmet Anıl Akbay**, Can B. Kalayci, Christian Blum. Application of Adapt-CMSA to the Electric Vehicle Routing Problem with Simultaneous Pickup and Deliveries. *Extended Abstract of EUROCAST 2024 – 19th International Conference on Computer Aided Systems Theory*, pages 16-18, 2024. [Conference website: `https://eurocast2024.fulp.ulpgc.es/`]

7. **Mehmet Anıl Akbay**, Can B. Kalayci, Christian Blum. Application of Adapt-CMSA to the Electric Vehicle Routing Problem with Simultaneous Pickup and Deliveries. *Selected to be included in Proceedings of the 19th International Conference on Computer Aided Systems Theory, Lecture Notes in Computer Science*. [Conference website: `https://eurocast2024.fulp.ulpgc.es/`]

8. **Mehmet Anıl Akbay**, Christian Blum, Michella Saliba. The Electric Vehicle Problem with Road Junctions and Road Types: An Ant Colony Optimization Approach *Proceedings of GECCO 2024 – The Genetic and Evolutionary Computation Conference*. (`https://doi.org/10.1145/3638529.3653997`). [CORE A conference]

9. **Mehmet Anıl Akbay** and Christian Blum. Two Examples for the Usefulness of STNWeb for Analyzing Optimization Algorithm Behavior. In *Proceedings of the Metaheuristics International Conference, Lecture Notes in Computer Science*, 2024. (`https://doi.org/10.1007/978-3-031-62922-8_25`).

10. (2nd. Reviewing Round) **Mehmet Anıl Akbay**, Christian Blum, Can B. Kalayci. CMSA Based on Set Covering Models for Packing and Routing Problem. Submitted to *Annals of Operations Research*. [Impact Factor 2022: 4.8 (JCR)]

CHAPTER 2

CMSA: GENERAL DESCRIPTION OF THE ALGORITHMIC FRAMEWORK

2.1 Introduction

This chapter provides a general description of the algorithmic framework of CMSA and the self-adaptive variant of CMSA developed in this thesis, henceforth labeled Adapt-CMSA. The description in this chapter is provided in the context of *CO problems* that can be modeled in terms of *binary ILPs*. For the description of both CMSA variants, we assume to be tackling a CO problem which can be modeled in terms of an ILP where $f()$ is the objective function to be minimized, and $x_i \in \{0,1\}$ $(i = 1,\dots,n)$ is the set of binary decision variables used to model the objective function and the constraints of the problem. Note that many NP-hard CO problems fall into this category of problems. Examples include the well-known traveling salesman problem (TSP) and the quadratic assignment problem (QAP), just to name two emblematic problems.

2.2 CMSA: The Baseline Algorithm

In the general case as described above, we introduce a solution component c_i^0 and a solution component c_i^1 for each binary variable x_i, $i = 1,\dots,n$. Hereby, c_i^0 corresponds to $x_i = 0$, while c_i^1 corresponds to $x_i = 1$. Moreover, $C = \{c_1^0,\dots,c_n^0,c_1^1,\dots,c_n^1\}$ is the complete set of $2n$ solution components. Any *candidate solution* S is a subset of C with $|s| = n$. In addition, it is required that S contains exactly one of the two components c_i^0 and c_i^1 for each $i = 1,\dots,n$. Finally, a candidate solution S is a valid solution if it fulfils all the constraints of the tackled problem.

2.2.1 Standard CMSA

Algorithm 2.1 Pseudo-code of standard CMSA

input 1: input graph G and the set of solution components C
input 2: values for CMSA parameters $n_{\rm a}$, age$_{\rm max}$, and $t_{\rm ILP}$
input 3: values for solution construction parameters $d_{\rm rate}$, $l_{\rm size}$
$S_{\rm bsf}$:= GenerateGreedySolution(C)
$C' := S_{\rm bsf}$
$age[c] := 0$ for all $c \in C$
while CPU time limit not reached **do**
  for $i := 1, \ldots, n_{\rm a}$ **do**
    S := ProbabilisticSolutionConstruction(C, $l_{\rm size}$, $d_{\rm rate}$)
    for all $c \in S$ **and** $c \notin C'$ **do**
      $age[c] := 0$
      $C' := C' \cup \{c\}$
    end for
  end for
  $S'_{\rm opt}$:= SolveSubinstance(C', $t_{\rm ILP}$)
  if $f(S'_{\rm opt}) < f(S_{\rm bsf})$ **then** $S_{\rm bsf} := S'_{\rm opt}$ **end if**
  Adapt(C', $S'_{\rm opt}$, age$_{\rm max}$)
end while
output: $S_{\rm bsf}$

Algorithm 2.1 provides the pseudo-code of a standard CMSA for binary optimization problems. Note that all functions in the pseudo-code are indicated with a special font as, for example, in GenerateGreedySolution(C). This function is used to initialize the best-so-far solution $S_{\rm bsf}$ with the solution generated by a greedy algorithm. This is done at the start of the algorithm. Moreover, the sub-instance C', which is solved by an ILP solver at each iteration, is initialized to $S_{\rm bsf}$. Note that, alternatively, $S_{\rm bsf}$ might be initialized to NULL and C' to the empty set. Each solution component $c \in C$ maintains a so-called age value $age[c]$. These age values are all initialized to zero. Note that the purpose of the age value of a solution component c is to count the number of consecutive CMSA iterations for which c forms part of C', without being included in the ILP-solution to the reduced problem instance generated on the basis of C'. At each iteration, CMSA iterates through four algorithmic steps. In the *construct* step, $n_{\rm a}$ valid solutions to the tackled problem are probabilistically constructed in function ProbabilisticSolutionConstruction(C, $l_{\rm size}$, $d_{\rm rate}$). In the *merge* step, those solution components that (1) are found in at least one of the constructed solutions from the *construct* step, and (2) do currently not form part of C', are added to C' and their age value is set to zero. Next, the *solve* step first generates a reduced problem instance on the basis of C', which is done by adding—for all $i = 1, \ldots, n$—the following constraints to the original ILP model of the tackled problem:

1. If $c_i^0 \in C'$ and $c_i^1 \notin C'$: add constraint $x_i = 0$ to the ILP model
2. If $c_i^0 \notin C'$ and $c_i^1 \in C'$: add constraint $x_i = 1$ to the ILP model

Note that, the more of these constraints are added to the original ILP, the smaller is the search space of the resulting sub-instance. Afterwards the extended ILP is solved in function SolveSubinstance(C', t_{ILP}), for example, by the application of an ILP solver with a CPU time limit of t_{ILP} seconds. Note that a variable x_i is only free in the extended ILP, if both solution components c_i^0 and c_i^1 form part of C'. Note also that the output of function SolveSubinstance(C', t_{ILP}) is—due to the computation time limit—not necessarily an optimal solution to the extended ILP. In those cases in which $f(S'_{\mathrm{opt}}) < f(S_{\mathrm{bsf}})$, the output of function SolveSubinstance(C', t_{ILP}) is set as S_{bsf}. Finally, in the *adapt* step, sub-instance C' is adapted in function Adapt(C', S'_{opt}, age$_{\mathrm{max}}$) depending both on S'_{opt} and on the age values of the solution components. This is done by increasing the age values of all components in $C' \setminus S'_{\mathrm{opt}}$ by one, and by re-initialising the age values of all components in S'_{opt} to zero. The final action in the *adapt* step consists in removing all those components from C' whose age value has reached the maximum allowed age of age$_{\mathrm{max}}$. This is done in order to prevent components that never appear in S'_{opt} to slow down the ILP solver in subsequent iterations. Figure 2.1 graphically summarizes the algorithm structure and components.

2.3 Self-Adaptive CMSA

Overly high sensitivity to changes in parameter values is a recognized problem in research on metaheuristics [176]. A metaheuristic is generally said to be *parameter sensitive* if (1) the algorithm performance for specific instances or instance groups strongly depends on the parameter values and if (2) the required parameter values for different instances or instance groups are rather different from each other. Unfortunately, such high sensitivity to parameter values was observed in some applications of CMSA in the literature, including the preliminary application to the NP-hard CO problem known as the *minimum positive influence dominating set* (MPIDS) problem [2]. Therefore, a self-adaptive variant of CMSA, named Adapt-CMSA, is developed with the aim of obtaining an algorithm less sensitive to parameter values. The obtained results from the experiments conducted in Chapter 3 demonstrate that Adapt-CMSA has several advantages over standard CMSA in the context of the MPIDS problem. Firstly, Adapt-CMSA does not require specific parameter tuning for subsets of the considered benchmark set. After a single parameter tuning session, Adapt-CMSA

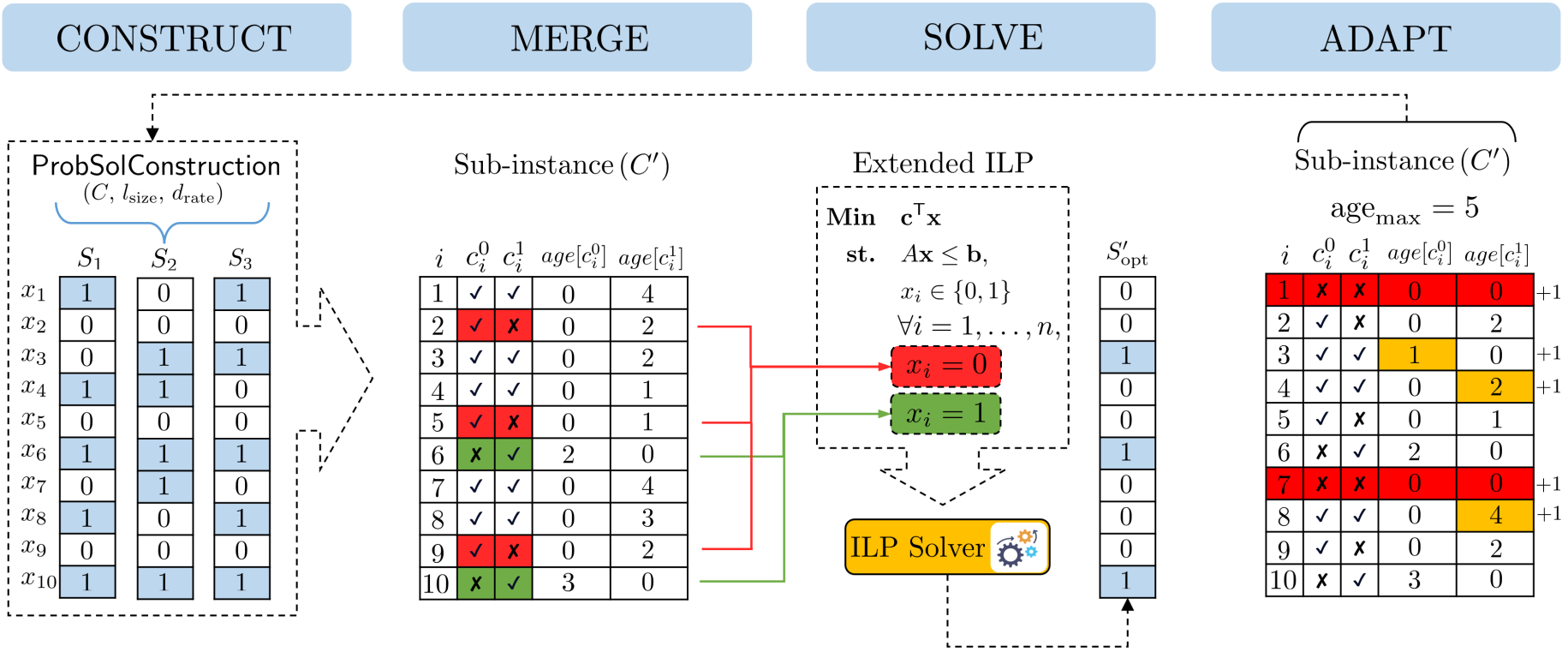


Fig. 2.1 Graphical abstract of CMSA for binary optimization problems.
Initially, in the "Construct" phase, three example solutions (S_1, S_2, S_3) are generated probabilistically. These solutions are combined into a sub-instance in the "Merge" phase, based on two components per variable (c_i^0, c_i^1) with so-called age values. Note that solution components included in the sub-instance are indicated by ✓, while excluded components are marked with ✗. The "Solve" phase constructs an Extended ILP model based on the solution components in the sub-instance (variables corresponding to the solution components marked red are fixed to value zero while the ones marked with green are fixed to value one). S'_{opt} is then determined using an ILP solver within a computational time limit. In the final "Adapt" phase, age values of solution components in $C' \setminus S'_{\text{opt}}$ are increased, while those in S'_{opt} reset to zero. Components reaching age_{max} are removed from the sub-instance, see lines colored red.

Algorithm 2.2 Pseudo-code of self-adaptive CMSA: Adapt-CMSA

1: **input 1:** input graph G and the set of solution components C
2: **input 2:** values for CMSA parameter t_{prop}, t_{ILP}
3: **input 3:** values for solution construction parameters α^{LB}, α^{UB}, α_{red}
4: $S_{\text{bsf}} :=$ GenerateGreedySolution(C)
5: $n_{\text{a}} := 1$; $\alpha_{\text{bsf}} := \alpha^{\text{UB}}$; $C' := S_{\text{bsf}}$
6: **while** CPU time limit not reached **do**
7: **for** $i := 1, \ldots, n_{\text{a}}$ **do**
8: $s :=$ ProbabilisticSolutionConstruction(C, S_{bsf}, α_{bsf})
9: $C' := C' \cup s$
10: **end for**
11: $(S'_{\text{opt}}, t_{\text{solve}}) :=$ SolveSubinstance(C', t_{ILP}) {This function returns two objects: (1) the obtained solution (S'_{opt}), (2) the required computation time (t_{solve})}
12: **if** $t_{\text{solve}} < t_{\text{prop}} \cdot t_{\text{ILP}}$ and $\alpha_{\text{bsf}} > \alpha^{\text{LB}}$ **then** $\alpha_{\text{bsf}} := \alpha_{\text{bsf}} - \alpha_{\text{red}}$ **end if**
13: **if** $f(S'_{\text{opt}}) < f(S_{\text{bsf}})$ **then**
14: $S_{\text{bsf}} := S'_{\text{opt}}$
15: $n_{\text{a}} := 1$
16: **else**
17: **if** $f(S'_{\text{opt}}) > f(S_{\text{bsf}})$ **then**
18: **if** $n_a = 1$ **then** $\alpha_{\text{bsf}} := \min\{\alpha_{\text{bsf}} + \frac{\alpha_{\text{red}}}{10}, \alpha^{\text{UB}}\}$ **else** $n_a = 1$ **end if**
19: **else**
20: $n_{\text{a}} := n_{\text{a}} + 1$
21: **end if**
22: **end if**
23: $C' := S_{\text{bsf}}$
24: **end while**
25: **output:** S_{bsf}

performs well across the entire benchmark set, containing instances of varying sizes. Secondly, Adapt-CMSA significantly outperforms standard CMSA in the context of large networks, where even specialized tuning does not enable CMSA to compete with Adapt-CMSA. We expect a similar advantage of Adapt-CMSA over standard CMSA in most applications where standard CMSA shows high parameter sensitivity.

The pseudo-code of self-adaptive CMSA (Adapt-CMSA) is provided in Algorithm 2.2. The first noticeable difference to standard CMSA is the absence of the age values. This is because Adapt-CMSA works with a fixed maximum age of one, that is, after each iteration all solution components apart from those that form part of the best-so-far solution S_{bsf} are removed from the sub-instance C' (see line 23). Another difference can bee seen in function ProbabilisticSolutionConstruction(C, S_{bsf}, α_{bsf}) for the probabilistic generation of solutions at each algorithm iteration (see line 8). Note that this latter function receives, apart from the set of all possible solution components (C), the currently

best-so-far solution S_{bsf} and a parameter α_{bsf} (where $0 \leq \alpha_{\text{bsf}} < 1$) as input. This parameter biases the construction of new solutions towards the best-so-far solution S_{bsf}. More specifically, the higher the value of α_{bsf}, the higher will be the similarity of the solutions constructed in ProbabilisticSolutionConstruction(C, S_{bsf}, α_{bsf}) to S_{bsf}.

The dynamic change of the value of α_{bsf} is one of the aspects that is handled in a self-adaptive way in Adapt-CMSA. First of all, Adapt-CMSA requires a lower bound α^{LB} and an upper bound α^{UB} for the value of α_{bsf} as input. Moreover, the step size α_{red} for the reduction of α_{bsf} must also be given as input. Adapt-CMSA starts with setting α_{bsf} to the highest possible value α^{UB}; see line 5.[1] In case the resulting ILP can be solved in a computation time t_{solve} which is below a proportion t_{prop} of the maximally possible computation time t_{ILP}, the value of α_{bsf} is reduced by α_{red}; see line 12. The rationale behind this step is the following one. In case the resulting ILP can be solved easily, the search space of the ILP is too small due to a rather low number of free variables. In order to have more free variables in the ILP, the solutions constructed in ProbabilisticSolutionConstruction(C, S_{bsf}, α_{bsf}) should be more different to S_{bsf}, which can be achieved by reducing the value of α_{bsf}.

The second aspect, which is handled in a self-adaptive way in Adapt-CMSA, is the number of solution constructions per iteration (n_a); see lines 13-22. The algorithm starts with a value of $n_a = 1$; see line 5. Moreover, in case the solution of the reduced ILP (S'_{opt}) improves over the best-so-far solution S_{bsf}, n_a is set back to one; see line 15. If, however, the solution of the reduced ILP (S'_{opt}) is strictly worse than the best-so-far solution S_{bsf}, the corresponding sub-instance was clearly too large and/or complex in order to be solved by the ILP solver within t_{ILP} seconds. In this case, if $n_a = 1$ the value of α_{bsf} is slightly increased (by $\frac{\alpha_{\text{red}}}{10}$); resp. n_a is set back to one, otherwise. In the remaining case ($f(S'_{\text{opt}}) = f(S_{\text{bsf}})$), n_a is incremented by one; see line 20. This is done because the sub-instance did not contain a better solution than S_{bsf}. At the same time, the sub-instance was solved within the allowed computation time of t_{ILP} seconds, which means that the size of the sub-instance should be increased.

Finally, note that functions SolveSubinstance(C', t_{ILP}) are exactly the same in both version of CMSA (standard CMSA and Adapt-CMSA).

The Figure 2.2 graphically summarizes the algorithm structure and components of the Adapt-CMSA and highlights its difference from the standard CMSA.

[1] Remember that this means that solutions constructed in this way will be more similar to S_{bsf} than with lower values of α_{bsf}.

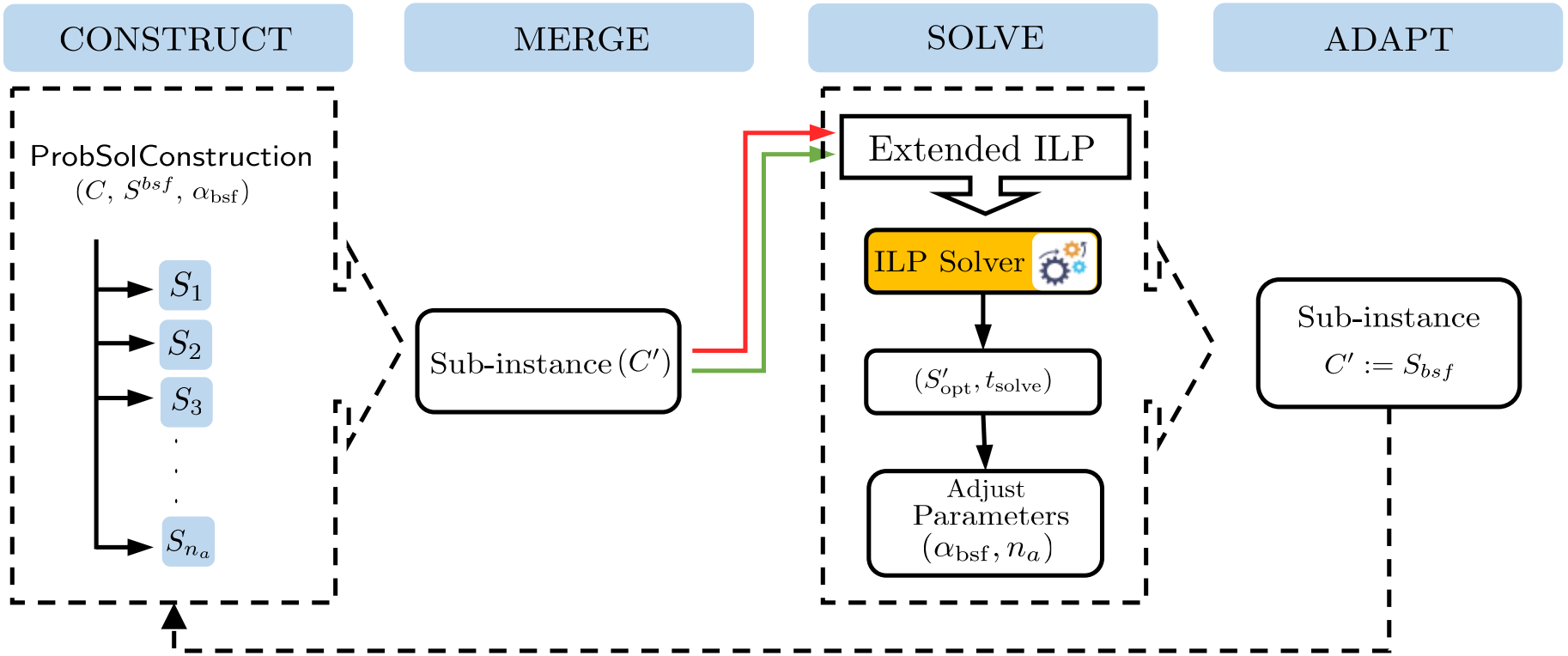


Fig. 2.2 Graphical abstract of Adapt-CMSA for binary optimization problems.
This diagram highlights the adaptive solution construction, where variable algorithm parameters adjust dynamically based on problem complexity and performance of the ILP solver. In the "Construct" phase, n_{a} solutions $(S_1, S_2, S_3, \ldots, S_{n_a})$ are generated probabilistically, influenced by the parameter α_{bsf} to bias solutions towards the best-so-far solution S_{bsf}. The process of creating the sub-instance, extending the ILP model, and solving it aligns with the Standard CMSA approach. After deriving S'_{opt}, the parameters α_{bsf} and n_{a} are updated based on the solver's output. The "Adapt" phase removes solution components except for those in S_{bsf}.

CHAPTER 3

APPLICATION of ADAPT-CMSA TO THE MINIMUM POSITIVE INFLUENCE DOMINATING SET PROBLEM

3.1 Introduction

This chapter demonstrates the application of CMSA and Adapt-CMSA to the so-called *Minimum Positive Influence Dominating Set* (MPIDS) problem [182, 183]. The content shown in this chapter was also presented in our papers [2, 4] that were published in the proceedings of *CCIA 2021: The International Conference of the Catalan Association for Artificial Intelligence* (`https://doi.org/10.3233/FAIA210112`) and in the International Journal of Computational Intelligence Systems (`https://doi.org/10.1007/s44196-022-00098-1`). The obtained results show that Adapt-CMSA has several advantages over standard CMSA in the context of the MPIDS problem. First, Adapt-CMSA does not require specific parameter tuning for subsets of the considered benchmark set. After applying parameter tuning once, Adapt-CMSA works very well for the whole benchmark set containing instances of very different sizes. Second, Adapt-CMSA clearly outperforms standard CMSA in the context of large networks for which even a specialized tuning does not enable CMSA to compete with Adapt-CMSA. We would expect a similar advantage of Adapt-CMSA over standard CMSA in most applications in which standard CMSA shows a high parameter sensitivity.

3.2 The Minimum Positive Influence Dominating Set Problem

The MPIDS problem is an NP-hard combinatorial optimization problem with applications in the context of social networks. Each vertex in such a network represents an individual—that is, a person—and edges indicate relationships, respectively interactions, between those individuals. The background of the MPIDS problem is that information propagated in social networks can have a significant, either positive or negative, impact on the respective parts of society. From social norms theory, it is known that the behavior of individuals can be

affected by the perception of others' thoughts and behaviors [69]. This makes it possible to exploit the relationships among people in social networks in order to obtain great benefits for both the economy and society. The aim of the MPIDS problem is to identify a small subset of influential individuals (or key individuals) in order to accelerate the spread of positive influence in a social network [86, 123]. Alternative applications of the MPIDS problem can be found in e-learning software [184], online business [151], drinking, smoking, and other drug-related problems [182].

From an algorithmic point of view, the efforts of the research community initially focused on the development of well-working greedy heuristics [28, 38, 64, 144, 152, 183]. In fact, until 2021, the best available approach was a greedy method from [28]. The development of successful metaheuristic approaches seemed much harder. This is shown by the results of the first two metaheuristics—an ILP-based memetic algorithm [119] and an algorithm based on swarm intelligence [120]—whose results are inferior to the greedy approach from [28]. The first metaheuristic that was able to improve over [28] is the iterated carousel approach from [169]. Finally, the currently best metaheuristic is the following one: a negative learning ant colony optimization approach from [168].

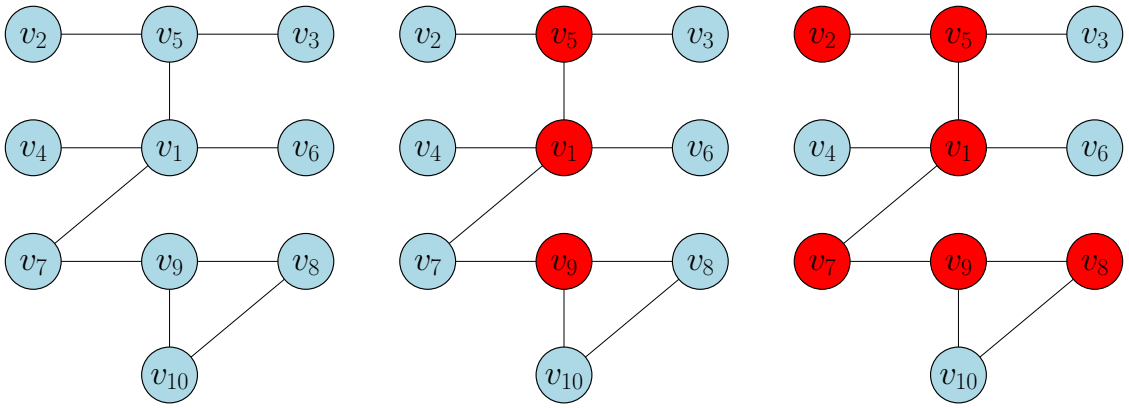
3.2.1 ILP Model for the MPIDS

In technical terms, the MPIDS problem can be described as follows. Given a simple,[1] connected, undirected graph $G = (V, E)$, the problem requires finding a subset s^* of V of minimum cardinality such that the following two conditions are fulfilled:

1. s^* is a dominating set of G. Remember that a subset $s \subseteq V$ of the vertices of an undirected graph G is called a dominating set, if and only if each vertex $v \in V$ forms either part of s or has at least one neighbor that forms part of s.
2. At least half of the neighbors of each vertex $v \in V$ form part of s^*.

Note that the MPIDS problem can easily be stated in terms of an ILP. The

[1] A simple graph does neither contain self-loops nor parallel edges



(a) A simple connected undirected graph

(b) A dominating set

(c) A positive influence dominating set

Fig. 3.1 An illustrative example of the MPIDS problem. Red vertices form part of the solution.

model is based on a binary variable x_i for each vertex $v_i \in V$.

$$\textbf{Minimize} \quad \sum_{i=1}^{n} x_i \tag{3.1}$$

$$\textbf{Subject to} \quad \sum_{v_j \in N(v_i)} x_j \geq \left\lceil \frac{deg(v_i)}{2} \right\rceil \quad \forall v_i \in V \tag{3.2}$$

$$x_i \in \{0, 1\} \tag{3.3}$$

Hereby, $N(v_i)$ is the neighborhood of v_i in input graph G, and $deg(v_i)$ is the degree of vertex v_i, where $deg(v_i) := |N(v_i)|$. Equation (3.2) ensures that a feasible solution contains at least half of the neighbors of each vertex $v_i \in V$.

In the context of the CMSA algorithms outlined in the next section, a valid solution $s \in V$ is represented as a set S of solution components that contains component c_i^1 for each $v_i \in s$ and component c_i^0 for each $v_i \in V \setminus s$. Moreover, the objective function value of a CMSA-solution S is defined via its corresponding set solution s, that is, $f(S) := |s|$. Note that $s := V$ is a trivial solution to the problem.

3.3 Application to the MPIDS problem

This section outlines the application of CMSA and Adapt-CMSA to the MPIDS problem by highlighting the core differences of both approaches.

First of all, the only problem-dependent part of both standard CMSA and Adapt-CMSA is the construction of feasible solutions.

At each iteration of the algorithm, solutions are constructed in ProbabilisticSolutionConstruction(C, l_{size}, d_{rate}) in the case of CMSA, respectively in function ProbabilisticSolutionConstruction(C, S_{bsf}, α_{bsf}) in the case of Adapt-CMSA. Both functions make use of the solution construction mechanism of the greedy procedure from [28]. They only differ in the way in which this procedure is made probabilistic. For the following discussion, remember that a vertex $v \in V$ is called *covered* with respect to a (partial) solution s if and only if at least half of its neighbors form part of s. In the opposite case, v is labeled as *uncovered*.

The solution construction mechanism utilized by both functions is shown in Algorithm 3.1. First, each solution s to be constructed is initialized by a set $s_{\text{par}} \subset V$ of nodes that must form part of an optimal solution; see line 3. Note that s_{par} is obtained by the application of a pre-processing procedure described in [28]. Then, at each step of the solution construction mechanism, the following is done. First, the set of all uncovered vertices with respect to (partial) solution s is determined; see line 5. This set is labeled U. Second, a node $v \in U$ such that $deg(v) \leq deg(v')$ for all $v' \in U$ is selected (line 6). Third, nodes from $N(v) \setminus s$ are iteratively added to s while $|N(v) \cap s| < \left\lceil \frac{deg(v)}{2} \right\rceil$; see lines 7-10. Hereby, exactly one vertex from $N(v) \setminus s$ is selected by function ChooseFrom($N(v) \setminus s$) at each entry of the while loop.

In standard CMSA, function ChooseFrom($N(v) \setminus s$) is implemented as follows. At first, a candidate list L is created. This list includes all vertices $v' \in N(v) \setminus s$. Each vertex v' in L is characterised by its *cover degree* $cov\_deg(v')$, which is the number of uncovered adjacent vertices of v'. Note that vertices in L are sorted according to a non-increasing cover degree value. Then, a uniform random number r is generated from the interval $[0, 1]$. If $r \leq d_{\text{rate}}$ (where d_{rate} is the so-called determinism rate) the vertex with the highest cover degree is selected and added to s. Otherwise, a vertex is selected randomly from the restricted candidate list which contains the first l_{size} vertices of L. Hereby, l_{size} is the size of the restricted candidate list. All vertices in the restricted candidate list have an equal probability $\frac{1}{l_{\text{size}}}$ of being selected.

In contrast, in Adapt-CMSA, function ChooseFrom($N(v) \setminus s$) is implemented in the following way. First, each vertex $v_i \in N(v) \setminus s$ such that $c_i^1 \in S_{\text{bsf}}$—that is, v_i forms part of the best-so-far solution—obtains a value $q(v_i) := (cov\_deg(v_i) + 1) \cdot \alpha_{\text{bsf}}$, while all other vertices $v_j \in N(v) \setminus s$ receive a value $q(v_j) := (cov\_deg(v_j) + 1) \cdot (1 - \alpha_{\text{bsf}})$. A vertex $\hat{v}$ is then chosen from $N(v) \setminus s$ according to the following probabilities:

$$\mathbf{p}(v') := \frac{q(v')}{\sum_{v'' \in N(v) \setminus s} q(v'')} \quad \forall\, v' \in N(v) \setminus s \tag{3.4}$$

Algorithm 3.1 Solution construction procedure (CMSA and Adapt-CMSA)

CMSA input: solution construction parameters d_{rate}, l_{size}
Adapt-CMSA input: solution construction parameter α_{bsf}
$s := s_{\text{par}}$ {$s_{\text{par}} \subset V$ is obtained from a pre-processing procedure}
while s is not a valid solution **do**
  Let $U \subseteq V$ be the set of uncovered vertices
  Choose $v \in U$ such that $deg(v) \leq deg(v')$ for all $v' \in U$
  while $|N(v) \cap s| < \left\lceil \frac{deg(v)}{2} \right\rceil$ **do**
    $\hat{v} :=$ ChooseFrom($N(v) \setminus s$)
    $s := s \cup \{\hat{v}\}$
  end while
end while
output: CMSA-solution S corresponding to the constructed solution s

In other words, the higher the value of parameter $\alpha_{\text{bsf}} \in [0, 1]$, the stronger the bias towards the best-so-far solution S_{bsf}. This bias does not exist in standard CMSA.

Figures 3.2 and 3.3 illustrate the general framework of both standard CMSA and Adapt-CMSA, respectively.

3.4 Experimental Evaluation

All experiments reported in the following were performed on a cluster of machines with Intel® Xeon® 5670 CPUs with 12 cores of 2.933 GHz and a minimum of 32 GB RAM. Note that CPLEX version 20.1 was used in one-threaded mode both in a standalone manner and within CMSA and Adapt-CMSA for solving the respective sub-instances. Two sets of experiments were performed. A comprehensive experimentation in the context of a new set of 800 scale-free networks is described in Section 3.4.1. The second set of experiments makes use of small, medium, and large problem instances that were already used in the related literature; see Section 3.4.2.

3.4.1 Experiments regarding scale-free networks

In order to be able to compare CMSA and Adapt-CMSA on a controlled set of benchmark instances with different features, we generated the following set of 800 scale-free networks by using the `igraph` software package [49]. An undirected network is said to be scale-free—or, equivalently, to follow a power law distribution—if the statistical distribution of the degrees of its nodes is as

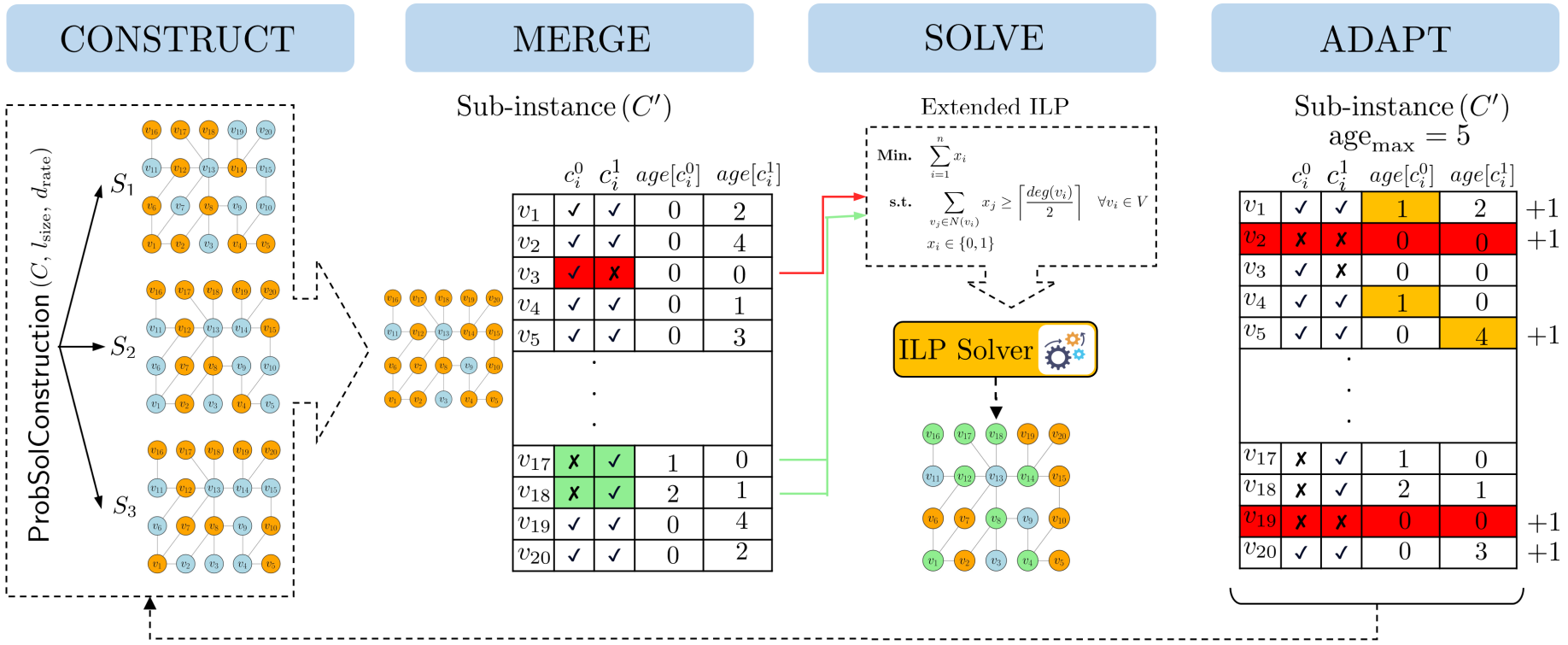


Fig. 3.2 Graphical abstract of CMSA for MPIDS

In the construct step, three example solutions are generated probabilistically for the MPIDS problem, with each node included in the respective solution colored orange. In the merge phase, a sub-instance is created by combining these solutions, categorizing nodes into three groups based on two solution components per node (c_i^0, c_i^1) with corresponding age values. Note that solution components included in the sub-instance are indicated by ✓, while excluded components are marked with ✗. Consequently, each node is categorized into one of three groups: nodes always included in those solutions created in the construct phase, nodes never included in these solutions, and the rest. This categorization results in formulating an extended ILP model by fixing the variable values to 1 for the first group (marked in green) and to 0 for the second group (marked in red) in the solve phase. The extended ILP model is then solved within a computational time limit to determine S'_{opt}. Finally, in the "Adapt" phase, the age values of solution components in $C' \setminus S'_{\text{opt}}$ are increased (highlighted in orange), while those in S'_{opt} are reset to zero. Components reaching age_{max}, set to 5 in this example, are subsequently removed from the sub-instance (colored red).

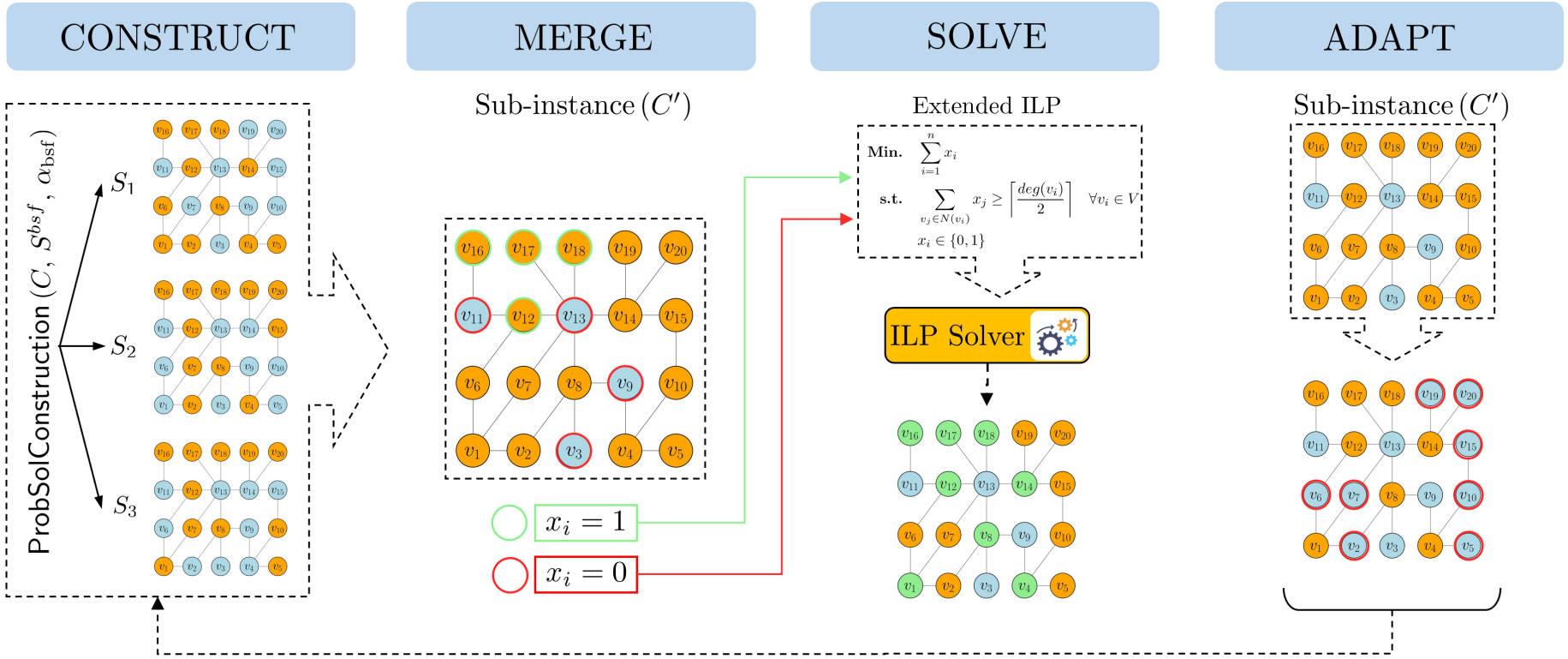


Fig. 3.3 Graphical abstract of Adapt-CMSA for MPIDS

Similar to the Standard CMSA, three example solutions are generated probabilistically in the "Construct" phase, with selected nodes in each solution highlighted in orange. In the subsequent "Merge" phase, solution components from these examples are combined to form a sub-instance. Note that no age value is required since age_{max} is set to 1 for Adapt-CMSA. The creation of the extended ILP model follows the same principle as in standard CMSA: nodes not included in any solution during the "Construct" phase (marked in red) are excluded from the model by fixing the corresponding variable to 0, while nodes included in all solutions (marked in green) are mandatorily included in S'_{opt} by fixing the variable to 1. This model is then solved by an ILP solver within a restricted time limit to obtain S'_{opt}, with green nodes indicating those included in S'_{opt}. Finally, in the "Adapt" phase, the sub-instance is modified by retaining only the nodes in S'_{opt} and removing all others (with removed nodes marked in red).

follows:

$$P(k) \sim k^{-\lambda} \quad , \tag{3.5}$$

where $P(k)$ is the probability that a given node has exactly k neighbours. Specifically, we used the random power-law graph generator function called `igraph_static_power_law_game` to generate networks differing in the following parameters:

- Number of nodes, $|V| \in \{1.000, 10.000, 50.000, 100.000, 250.000, 500.000, 750.000, 1.000.000\}$.
- Number of edges: $l \in \{5, 10, 20, 30\}$, where $l \cdot |V|$ is the number of edges
- Exponent for the power law exponential distribution, $\lambda \in \{2, 2.25, 2.5, 2.75, 3\}$. Note that parameter λ establishes the pace at which the probability of having highly connected nodes decreases.

Note that five networks were generated for all combinations of the number of nodes, the number of edges, and the exponent for the power law exponential distribution. This makes a total of 800 networks. Neither of the generated networks has self-loops or multiple edges between a pair of nodes. Note that, following the suggestion in the `igraph` package, the `finite_size_correction` mechanism [39] for the generation of the networks was used. Our reason for choosing power-law, scale-free networks for the comparison between CMSA and Adapt-CMSA is that they are generally accepted models for social networks [15, 72].[2]

For the purpose of parameter tuning, we separately generated one graph for each combination of $|V| \in \{50.000, 100.000, 500.000, 1.000.000\}$, $l \in \{5, 30\}$ and $\lambda \in \{2, 3\}$. That is, 16 graphs were used for parameter tuning purposes. In particular, we used the scientific tuning software irace [124] for fine-tuning the parameters of CMSA and Adapt-CMSA. The parameters of CMSA (together with their domains allowed for tuning) are the following ones:

1. Number of solution constructions per iteration, $n_{\mathrm{a}} \in \{1, 2, \ldots, 19, 20\}$.
2. Upper limit for the age values, $\mathrm{age}_{\max} \in \{1, 2, \ldots, 9, 10\}$.
3. Time limit for CPLEX per call, $t_{\mathrm{ILP}} \in \{1, 2, \ldots, 49, 50\}$ CPU seconds.

[2]Note that, due to the size and the number of the generated graphs, this instance set is very large and consumes a large amount of memory (more than 17 GB). It can be obtained by contacting the author of this thesis.

4. Determinism rate for solution construction, $d_{\text{rate}} \in [0.0, 0.99]$.
5. Length of the restricted candidate list, $l_{\text{size}} \in \{2, 3, \ldots, 9, 10\}$.

The parameters of Adapt-CMSA, together with their domains, are the following ones:

1. Time limit for CPLEX per call, $t_{\text{ILP}} \in \{1, 2, \ldots, 49, 50\}$ CPU seconds.
2. Lower bound for the bias towards the best-so-far solution, $\alpha^{\text{LB}} \in [0.6, 0.99]$.
3. Upper bound for the bias towards the best-so-far solution, $\alpha^{\text{UB}} \in [0.6, 0.99]$.
4. Step size for the reduction of this bias, $\alpha_{\text{red}} \in [0.01, 0.1]$.
5. Parameter used for determining when to reduce the bias, $t_{\text{prop}} \in [0.1, 0.8]$.

Note that in the case of numerical parameters, the precision of irace was fixed to two positions behind the comma. Both for the tuning of CMSA and Adapt-CMSA irace was applied with a budget of 3.000 algorithm applications. The time limit for each problem instance was set to $|V|/100$ CPU seconds. The outcome of the tuning runs can be summarised as follows:

- **CMSA parameter values:** $n_{\text{a}} = 1$, $\text{age}_{\text{max}} = 1$, $t_{\text{ILP}} = 38$, $d_{\text{rate}} = 0.53$, $l_{\text{size}} = 2$.
- **Adapt-CMSA parameter values:** $t_{\text{ILP}} = 44$, $\alpha^{\text{LB}} = 0.85$, $\alpha^{\text{UB}} = 0.97$, $\alpha_{\text{red}} = 0.05$, $t_{\text{prop}} = 0.13$.

Note that both parameter settings indicate that we are dealing with very large graphs. In the case of CMSA, for example, $n_{\text{a}} = 1$ and $\text{age}_{\text{max}} = 1$ are set in this restrictive way because, otherwise, the size of the sub-instances would be too large to be solved by CPLEX. Similarly, the parameters of Adapt-CMSA are characterised by a strong bias towards the best-so-far solution in order to keep the sub-instance as small as possible, while still being able to find improving solutions.

With the final parameter settings as provided above, both CMSA and Adapt-CMSA were applied exactly once to each of the 800 problem instances. The computation time limit was the same as the one chosen for tuning, that is, $|V|/100$ CPU seconds. The results are shown in a summarised way in the graphic of Figure 3.4. The graphic is composed of $4 \times 8 = 24$ sub-graphics for each combination of $|V| = n$ (rows) and l (columns). Each sub-graphic shows—for

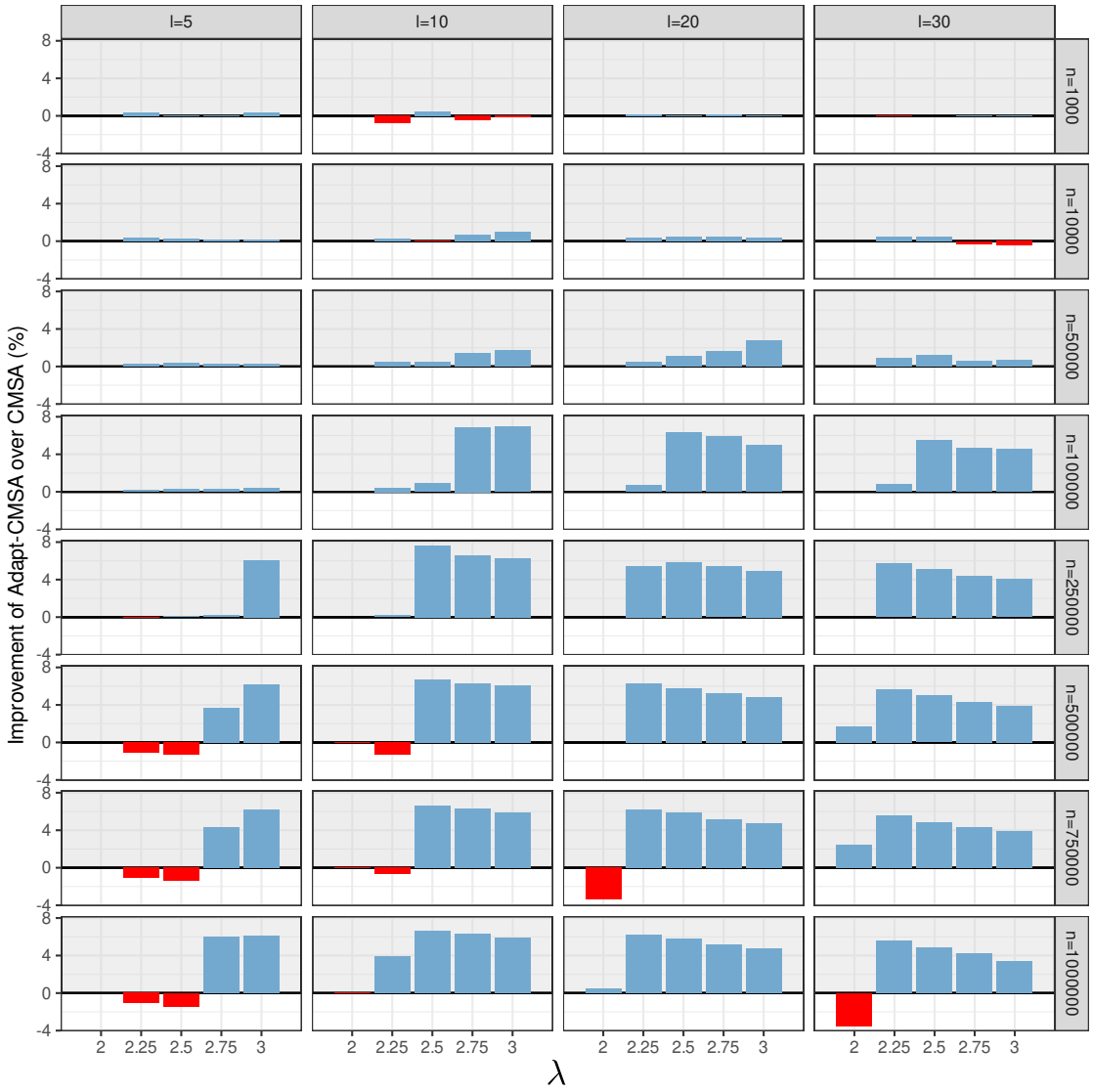


Fig. 3.4 Average improvement of Adapt-CMSA over standard CMSA (in percent)

all five values of λ—the average improvement of Adapt-CMSA over CMSA (in percent). Note that those cases in which Adapt-CMSA improves over CMSA are additionally marked by bars in blue color, while bars in red color indicate the cases in which CMSA is better than Adapt-CMSA.

The following observations can be made. First, for smaller graphs (up to 50.000 nodes) not much difference between the two algorithms can be observed. However, starting from 100.000 nodes, Adapt-CMSA clearly outperforms CMSA. This holds especially with a growing number of nodes and a growing number of edges. Interestingly, for the smallest values of λ—that is, for $\lambda \in \{2, 2.25\}$ in the case of $l = 5$, respectively for $\lambda = 2$ in the case of of the remaining values of l—CMSA often seems to have a slight advantage over Adapt-CMSA. In other words, the advantage of Adapt-CMSA over CMSA is higher for graphs with less

nodes with high degrees. Nevertheless, these results provide a strong indication for the general superiority of Adapt-CMSA over CMSA.

3.4.2 Experiments regarding instances from the literature

In our second set of experiments, we compare CMSA and Adapt-CMSA to the standalone application of CPLEX and to the best metaheuristic from the literature [169] (ICG). This is done in the context of 17 social networks that are partially used in the related literature on the MPIDS problem. These networks are of small and medium size, and contain between 34 and 36.692 nodes and between 788 and 198.050 edges. In addition, CPLEX and both CMSA variants were applied to 10 larger social networks from the SNAP library that contain between 37.700 and 1.134.890 nodes and between 2.289.003 and 3.387.388 edges (`https://snap.stanford.edu/data/`).

While CPLEX was applied exactly once to each of these 27 problem instances, both CMSA and Adapt-CMSA were applied 10 times to each instance. A computation time limit of 2 hours was given to each CPLEX run. On the contrary, much less time was given to the CMSA variants. In the case of the 17 small/medium size problem instances we allowed a computation time of $|V|/10$ CPU seconds for each run. A relatively shorter computation time of $|V|/100$ CPU seconds was allowed for the application to the large instances from the SNAP library. The main reason for this difference is that $|V|/100$ seconds would have been a very short computation time for most of the small and medium size instances.

In a first experiment we applied both CMSA and Adapt-CMSA with the parameter values from the previous section to all 27 instances. The obtained results are shown in numerical form in Table 3.1 (small/medium size instances) and Table 3.2 (large instances). These tables have the following structure. The first column contains the instance name, and the second column provides information about the quality of the best solutions known to date. Columns with heading 'q' report on the quality of the best solutions found by the four approaches, and columns with heading 'avg' provide the respective average solution quality. Furthermore, columns with heading '$\overline{t(s)}$' indicate the average computation times of CMSA and Adapt-CMSA to find the best solutions in each run. Note that the information about average computation times was not provided in [169] for ICG. The authors, however, state that they chose a computation time limit of $|V| \cdot 30/1000$, because this assured convergence of their algorithm in the case of all considered problem instances. In other words, ICG would not profit from a

Table 3.1 Numerical results for small to medium size instances.

Network	best	CPLEX		ICG		CMSA			Adapt-CMSA		
	known	q	gap (%)	q	avg	q	avg	$\overline{t(s)}$	q	avg	$\overline{t(s)}$
Karate	15	**15**	0.00	n.a.	n.a.	**15**	15.00	0.004	**15**	15.00	0.004
Dolphins	30	**30**	0.00	**30**	30.0	**30**	30.00	0.02	**30**	30.00	0.02
Football	63	**63**	0.00	64	64.0	**63**	63.70	3.06	**63**	63.80	3.07
Jazz	79	**79**	0.00	n.a.	n.a.	**79**	79.00	1.66	**79**	79.00	0.23
CA-AstroPh	6736[a]	6740	0.30	6808	6812.95	6751	6753.60	1311.32	**6738**	6739.40	1329.03
CA-GrQc	2587	**2587**	0.00	**2587**	2587.50	**2587**	2587.00	19.08	**2587**	2587.00	1.87
CA-HepPh	4718	**4718**	0.01	4743	4746.85	4726	4727.50	604.52	**4718**	4718.20	482.04
CA-HepTh	4471	**4471**	0.00	4481	4483.10	4474	4474.70	306.65	**4471**	4471.00	17.60
CA-CondMat	9584	**9584**	0.06	9625	9627.85	9593	9595.00	1773.06	9585	9586.20	1048.08
Email-Enron	11682	**11682**	0.00	11737	11740.65	11692	11693.40	1815.34	**11682**	11682.80	690.15
ncstrlwg2	2994	**2994**	0.00	n.a.	n.a.	2995	2995.00	20.92	**2994**	2994.20	213.07
actors-data	3092	**3092**	0.24	n.a.	n.a.	3099	3100.90	763.51	3093	3093.70	597.38
ego-facebook	1973	**1973**	0.00	**1973**	1973.25	1975	1975.00	5.553	**1973**	1973.00	95.42
socfb-Brandeis99	1397[a]	**1400**	1.41	1443	1445.05	1427	1428.90	324.00	1414	1416.30	340.99
socfb-nips-ego	1398	**1398**	0.00	n.a.	n.a.	**1398**	1398.00	0.04	**1398**	1398.00	0.03
socfb-Mich67	1327[a]	**1329**	1.56	n.a.	n.a.	1342	1344.60	241.66	1340	1342.70	288.82
soc-gplus	8244	**8244**	0.00	n.a.	n.a.	8250	8251.20	956.37	**8244**	8244.00	2.425
average		**3552.88**				3558.59	3559.56		3554.35	3554.96	

[a]: these best-known results were obtained by [2] (CA-AstroPh) and [168] (socfb-Brandeis99, socfb-Mich67)

Table 3.2 Numerical results for large SNAP networks.

Network	Best	CPLEX		CMSA			Adapt-CMSA		
	known	q	gap (%)	q	avg	$\overline{t(s)}$	q	avg	$\overline{t(s)}$
musae_git	9752	**9752**	0.00	9793	9796.90	356.24	9757	9758.00	361.02
loc-gowalla_edges	67617	**67617**	0.07	67723	67727.90	1902.18	67690	67695.20	1915.43
gemsec_facebook_artist	15194	**15194**	1.20	15319	15330.70	494.05	15256	15259.50	484.12
deezer_HR	22699	54573	95.68	22567	22605.10	541.96	**22338**	22354.50	523.39
com-youtube	351281	**351281**	0.00	351960	351972.50	11134.93	351422	351431.20	11053.78
com-dblp	120492	**120492**	0.08	120640	120647.00	3082.43	120566	120576.30	3086.46
Amazon0302	130378	262111	97.50	128913	128939.80	2606.64	**128587**	128624.40	2605.34
Amazon0312	180853	400727	95.41	183113	183113.00	1.27	**174495**	174832.50	3994.77
Amazon0505	183114	410236	95.19	185310	185310.00	1.32	**176882**	177175.90	4100.25
Amazon0601	179964	403394	96.94	182279	182279.00	1.405	**173509**	173929.10	4030.06
average		209537.70		126761.70			**124050.20**		

higher computation time limit. Finally, the gap (in percent) between the solution obtained by CPLEX and the best lower bound is indicated in the column with heading 'gap(%)'. Note that when the gap is zero, CPLEX was able to prove optimality. The best result for each instance is shown in bold font. Furthermore, in case the best solution known so far was improved, the respective result is underlined. Finally, in those cases in which none of the algorithms was able to reach the currently best known solution, we provide at the bottom of the table an indication of the algorithm that obtained the respective best known solution.

The following observations can be made. First, CPLEX performs strongly for small and medium size instances. Apart from instance `CA-AstroPh`, CPLEX obtains all best known solutions. Only in six out of 17 cases, CPLEX is not able to prove optimality of these results. The performance of Adapt-CMSA is very similar to the one of CPLEX. In one case (instance `CA-AstroPh`) Adapt-CMSA outperforms CPLEX both in terms of best-performance and in average-performance. On the downside, in four other cases (instances `CA-CondMat`, `actors-data`, `socfb-Brandeis99` and `socfbMich67`) the results of Adapt-CMSA fall slightly short of those of CPLEX. On the other side, Adapt-CMSA clearly outperforms CMSA, which (with the parameter setting for scale-free networks) only matches the results of Adapt-CMSA for six out of 17 problem instances. Finally, note that both CMSA and Adapt-CMSA clearly outperform the most recent metaheuristic from the related literature (ICG). Concerning the large instances from the SNAP library—see Table 3.2—we can state that the standalone application of CPLEX clearly starts to fail with a growing problem instance size. In fact, in five out of 10 cases—see the ones with an optimality gap of more than 95%—CPLEX is only able to provide the trivial solution that simply contains all network nodes. In addition, it can also be observed that the standard CMSA approach fails for instances `Amazon0312`, `Amazon0505` and `Amazon0601`. In these three cases, standard CMSA is not able to improve over the initial solutions provided by the greedy approach. Adapt-CMSA, on the other side, works very well also for these large-size SNAP networks. In fact, Adapt-CMSA is able to obtain new best known solutions in five out of 10 cases. Moreover, in those five cases in which CPLEX still works fine, the results of Adapt-CMSA are only slightly worse than those of CPLEX. Therefore, a first conclusion of this work is that Adapt-CMSA appears to be a CMSA variant that does not require to be specifically tuned for subsets of the considered benchmark set. It shows a high performance over the whole range of benchmark instances with one single parameter value set.

Table 3.3 Improvement of CMSA after specific tuning for small and medium size instances.

Network	best	CMSA			CMSA (special tuning)		
	known	q	avg	$\overline{t(s)}$	q	avg	$\overline{t(s)}$
Karate	15	**15**	15.00	0.004	**15**	15.00	0.01
Dolphins	30	**30**	30.00	0.02	**30**	30.00	0.03
Football	63	**63**	63.70	3.06	**63**	63.70	4.81
Jazz	79	**79**	79.00	1.66	**79**	79.00	0.36
CA-AstroPh	6736	6751	6753.60	1311.32	**6736**	6737.30	1181.44
CA-GrQc	2587	**2587**	2587.00	19.08	**2587**	2587.00	1.57
CA-HepPh	4718	4726	4727.50	604.52	**4718**	4718.10	332.57
CA-HepTh	4471	4474	4474.70	306.65	**4471**	4471.00	14.08
CA-CondMat	9584	9593	9595.00	1773.06	**9584**	9584.10	1001.96
Email-Enron	11682	11692	11693.40	1815.34	**11682**	11682.00	1471.90
ncstrlwg2	2994	2995	2995.00	20.92	**2994**	2994.00	26.65
actors-data	3092	3099	3100.90	763.51	**3091**	3092.30	521.21
ego-facebook	1973	1975	1975.00	5.553	**1973**	1973.00	25.26
socfb-Brandeis99	1397	1427	1428.90	324.00	**1406**	1407.90	215.55
socfb-nips-ego	1398	**1398**	1398.00	0.04	**1398**	1398.00	0.04
socfb-Mich67	1327	1342	1344.60	241.66	**1335**	1338.30	198.67
soc-gplus	8244	8250	8251.20	956.37	**8244**	8244.20	808.90
average		3558.59	3559.56		3553.29	3553.82	

In a second experiment, we aimed at studying the change of performance of standard CMSA when specifically tuned for small and medium size problem instances on one side, and for large SNAP network on the other side. Again, we used irace for the purpose or parameter tuning. For small and medium size instances, the budget given to irace consisted of 1000 algorithm applications, and instances CA-AstroPh and socfb-Brandeis99 were used for tuning. The same budget was used for the tuning run concerning large SNAP networks. In this case, instances Amazon0505 and Amazon0601 were used for tuning. The outcome of these two tuning experiments was the following one:

- **Small/med. size instances:** $n_{\mathrm{a}} = 1$, $\mathrm{age}_{\mathrm{max}} = 3$, $t_{\mathrm{ILP}} = 16$, $d_{\mathrm{rate}} = 0.09$, $l_{\mathrm{size}} = 8$.
- **Large SNAP networks:** $n_{\mathrm{a}} = 1$, $\mathrm{age}_{\mathrm{max}} = 1$, $t_{\mathrm{ILP}} = 39$, $d_{\mathrm{rate}} = 0.95$, $l_{\mathrm{size}} = 10$.

Clearly, the parameter settings for small and medium size instances result in much larger sub-instance sizes than those for large SNAP networks. With these new parameter settings we repeated the experiments of CMSA. The results, in comparison to the CMSA results obtained with the previous parameter values, are shown in Tables 3.3 and 3.4.

The new CMSA results improve substantially in the case of small and medium size problem instances (Table 3.3). In fact, CMSA is now able to generate for 14

Table 3.4 Results of CMSA after specific tuning for the large SNAP networks.

Network	Best	CMSA			CMSA (special tuning)		
	known	q	avg	$\overline{t(s)}$	q	avg	$\overline{t(s)}$
musae_git	9752	**9793**	9796.90	356.24	9890	9896.40	359.88
loc-gowalla_edges	67617	**67723**	67727.90	1902.18	67993	68008.30	1957.53
gemsec_facebook_artist	15194	**15319**	15330.70	494.05	15511	15526.70	498.52
deezer_HR	22699	**22567**	22605.10	541.96	22727	22744.50	538.43
com-youtube	351281	**351960**	351972.50	11134.93	352225	352245.80	11246.08
com-dblp	120492	**120640**	120647.00	3082.43	120970	120977.20	3131.91
Amazon0302	130378	**128913**	128939.80	2606.64	130386	130422.20	2600.53
Amazon0312	180853	183113	183113.00	1.27	**180438**	181974.30	2372.35
Amazon0505	183114	185310	185310.00	1.32	**182160**	183670.60	1527.61
Amazon0601	179964	182279	182279.00	1.405	**179584**	181282.40	2197.99
average		126761.70	126772.19		**126188.40**	126674.84	

out of 17 problem instances the best known solutions. In one case—see instance actors-data—CMSA is even able to generate a new best known solution of value 3091. With the specialised parameter setting, CMSA is now even able to perform slightly better, on average, than Adapt-CMSA (compare to Table 3.1). The results for large SNAP instances, however, show that specialized tuning does not help in this case. Even though the results of CMSA improve over the original ones from Table 3.2 in the case of those problem instances that were used for tuning (Amazon0505 and Amazon0601), they become worse for seven out of 10 problem instances. Moreover, even in those cases in which CMSA is able to improve with a specialised parameter setting, the results are still clearly inferior to those of Adapt-CMSA.

3.5 Conclusions

One of the occasional disadvantages of CMSA is the need for repeated parameter tuning for subsets of the considered benchmark set. For dealing with this problem, we proposed in this thesis a self-adaptive variant of CMSA, called Adapt-CMSA, that adjusts its parameters on the fly in order to be able to solve problem instances of very different sizes without the need of re-tuning. Experiments were performed in this chapter in the context of the minimum positive influence dominating set (MPIDS) problem.

Based on the obtained results we can say that Adapt-CMSA has several advantages over standard CMSA in the context of the MPIDS problem. First, Adapt-CMSA does not need to be specifically tuned for subsets of the considered benchmark set. After one single tuning run, Adapt-CMSA works very well for the whole benchmark set, which contains instances of very different sizes. Second, Adapt-CMSA clearly outperforms standard CMSA in the context of large

networks for which even a specialised tuning does not enable CMSA to compete with Adapt-CMSA.

CHAPTER 4

ADAPT-CMSA FOR THE EVRP-TW-SPD AND PARTIAL BATTERY CHARGING

4.1 Introduction

Once demonstrated the advantages of Adapt-CMSA over CMSA in the context of a simple-to-express problem in Chapter 3, this chapter presents the application of Adapt-CMSA to the *Electric Vehicle Routing Problem with Time Windows, Simultaneous Pickup and Delivery and Partial Battery Charging* (EVRP-TW-SPD). The content shown in this chapter was also presented in our paper [5] that was published in the proceedings of *MIC 2022: 14th. Metaheuristics International Conference* (`https://doi.org/10.1007/978-3-031-26504-4_1`). Further developments and applications of Adapt-CMSA were detailed in additional works. Notably, an extended abstract discussing the application of Adapt-CMSA to the Electric Vehicle Routing Problem with Simultaneous Pickup and Deliveries was presented at *EUROCAST 2024 – 19th International Conference on Computer Aided Systems Theory* (`https://eurocast2024.fulp.ulpgc.es/`). Additionally, the research on Adapt-CMSA based on Set Covering Models for Packing and Routing Problems is currently submitted to the Annals of Operations Research journal and is under review (2nd round).

The tackled problem first of all considers time window (TW) constraints for the delivery of goods to customers. Note that time windows can be used to control the visiting times of the customers, which might be regulated by local jurisdictions, but also by the customers themselves. It also considers simultaneous pickup and delivery (SPD) constraints regarding customer deliveries. When dealing with SPD constraints in the context of vehicle routing problems, it is important to note that each customer's demand may consist of two distinct requirements: (1) delivering goods to the demand point, known as the *"delivery demand"*, and (2) collecting goods from the demand point, known as the *"pickup demand"*. It is necessary to satisfy both demands simultaneously when a vehicle visits a particular customer. This practice is commonly associated with

reverse logistics. Finally, in conventional EVRP models, electric vehicle batteries are assumed to be fully charged upon visiting a charging station. However, in this chapter we make use of a more realistic scenario by allowing for partial recharging. This departure from the conventional approach is noteworthy, as it better reflects the real-world operational conditions of electric vehicles, as opposed to the less realistic expectation of complete charging.

Through the formulation of the addressed problem as a mixed integer linear programming (MILP) model, suitable for solution by general-purpose solvers like CPLEX or Gurobi, our computational experiments highlight the increasing complexity when considering TW and SPD constraints, in conjunction with the limited driving range of electric vehicles. The complexity encountered, even when solving small-sized instances to optimality, presents significant challenges for CPLEX 20.1, often resulting in valid solutions with considerable optimality gaps. This shows the need for more effective algorithms, such as Adapt-CMSA, capable of deriving good-quality solutions across both small and large problem instances.

Building on the findings from the computational experiments detailed in the previous chapter, the effectiveness of Adapt-CMSA in handling complex problems becomes increasingly apparent. Remember that, unlike standard CMSA, Adapt-CMSA does not require specific parameter tuning for different subsets of the considered benchmark set. Once the parameter tuning is applied, Adapt-CMSA demonstrates robust performance across the entire benchmark set, which contains instances of varying sizes. Furthermore, as in the previous application of Adapt-CMSA in the context of MPIDS, Adapt-CMSA significantly outperforms standard CMSA, especially in the context of large networks where even specialised tuning fails to make CMSA competitive with Adapt-CMSA. This suggests that Adapt-CMSA's advantage over standard CMSA might extend to most applications where standard CMSA exhibits high parameter sensitivity.

In this line, in order to tackle large-sized problem instances two variants of Adapt-CMSA have developed. The first variant, henceforth denoted by Adapt-Cmsa-Std, utilizes a matrix form representation for solutions and sub-instances considered by the algorithm. Moreover, the MILP formulation from Section 4.2 is used to solve sub-instances. On the contrary, in the second variant, henceforth denoted by Adapt-Cmsa-SetCov, each generated solution and each considered sub-instance is represented by a set of vehicle routes. Furthermore, a set-covering-based MILP model is utilized to derive a solution to the sub-instance. The performances of both approaches are compared in the context of problem

instances of different types and sizes from the literature.

4.2 Problem Description and Mathematical Model

This section provides a technical description of the EVRP-TW-SPD under the assumption of partial recharging (PR) along with its corresponding MILP formulation. The presented model is an extension of the model for the EVRP-TW-PR proposed by [104], which—in turn—is a modified variant of the model for the EVRP-TW proposed in [166]. In particular, we extended the model further in order to consider SPD constraints. We adopt the same notation as in the previous works to maintain consistency with the existing literature. In this context, the problem involves a set of N customers, denoted by $V = \{1, \ldots, N\}$, and a set of charging stations, denoted by F. To allow for multiple visits to any charging station, we defined a set $F^{'}$ that contains multiple copies of each charging station from F. The depot is represented by nodes 0 and $N + 1$, where node 0 is the starting point and node $N + 1$ is the ending point for each route. Note that both 0 and $N + 1$ refer to the same depot. The set $V^{'} = V \cup F^{'}$ contains all customers and dummy charging stations, with the sub-indexes 0, $N + 1$, or both, indicating the inclusion of the respective instances of the depot. Based on the above notations, we define the following sets:

1. $F_0^{'} := F^{'} \cup \{0\}$
2. $V_0^{'} := V^{'} \cup \{0\}$
3. $V_{N+1}^{'} := V^{'} \cup \{N + 1\}$
4. $V_{0,N+1}^{'} := V^{'} \cup \{0\} \cup \{N + 1\}$

Following established sets and notations, the EVRP-TW-SPD can be defined on a complete, directed graph $G(V_{0,N+1}^{'}, A)$. $A = \{(i, j)|i, j \in V_{0,N+1}^{'}, i \neq j\}$ is the set of arcs where each arc has a corresponding distance d_{ij} and travel time t_{ij}. The energy consumed per unit distance traveled by an electric vehicle is denoted by a constant h. A fleet of electric vehicles with identical loading capacity C and battery capacity Q is stationed at a depot to satisfy delivery demand $q_i > 0$ and pickup demand $p_i > 0$ of customers simultaneously. Each vertex $i \in V_{0,N+1}^{'}$ is only allowed to be visited within a time window $[e_i, l_i]$ that is defined by the earliest and latest possible visiting times allowed. Moreover, each customer $i \in V$ has a service time s_i, which refers to the time an electric vehicle spends visiting a customer. When an EV visits a charging station, its battery is charged with a constant charging rate of $g > 0$.

The following decision variables are used to formulate the MILP model of the problem. The binary decision variable x_{ij} takes a value of 1 if the arc (i, j) is included in the route and 0 otherwise. The starting time of the service for each customer visited by the electric vehicle is monitored by the decision variable τ_i. Moreover, to keep track of the battery's state of charge upon arrival and departure at each vertex $i \in V'_{0,N+1}$, the decision variables y_i and Y_i are employed, respectively. Furthermore, the remaining cargo to be delivered to the customers of the route and the amount of cargo already collected (picked up) at the previously visited customers are represented by the variables u_{ij} and v_{ij}, respectively. Therefore, the MILP model, henceforth denoted as $\texttt{ILP}^{\text{EVRP}}_{\text{std}}$, can be presented as follows.

$$\textbf{Min} \quad \sum_{i \in V'_0, j \in V'_{N+1}} d_{ij} x_{ij} + \sum_{j \in V'_{N+1}} M x_{0j} \tag{4.1}$$

$$\sum_{j \in V'_{N+1}, i \neq j} x_{ij} = 1 \qquad \forall i \in V \tag{4.2}$$

$$\sum_{j \in V'_{N+1}, i \neq j} x_{ij} \leq 1 \qquad \forall i \in F' \tag{4.3}$$

$$\sum_{i \in V'_0, i \neq j} x_{ij} - \sum_{i \in V'_{N+1}, i \neq j} x_{ji} = 0 \qquad \forall j \in V' \tag{4.4}$$

$$\tau_i + (t_{ij} + s_i) x_{ij} - l_0(1 - x_{ij}) \leq \tau_j \qquad \forall i \in V_0, j \in V'_{N+1}, i \neq j \tag{4.5}$$

$$\tau_i + t_{ij} x_{ij} + g(Y_i - y_i) - (l_0 + gQ)(1 - x_{ij}) \leq \tau_j \qquad \forall i \in F', \forall j \in V'_{N+1}, i \neq j \tag{4.6}$$

$$e_j \leq \tau_j \leq l_j \qquad \forall j \in V'_{0,N+1} \tag{4.7}$$

$$0 \leq u_{0j} \leq C \qquad \forall j \in V'_{N+1} \tag{4.8}$$

$$v_{0j} = 0 \qquad \forall j \in V'_{N+1} \tag{4.9}$$

$$\sum_{i \in V'_0, i \neq j} u_{ij} - \sum_{i \in V'_{N+1}, i \neq j} u_{ji} = q_j \qquad \forall j \in V' \tag{4.10}$$

$$\sum_{i \in V'_{N+1}, i \neq j} v_{ji} - \sum_{i \in V'_0, i \neq j} v_{ij} = p_j \qquad \forall j \in V' \tag{4.11}$$

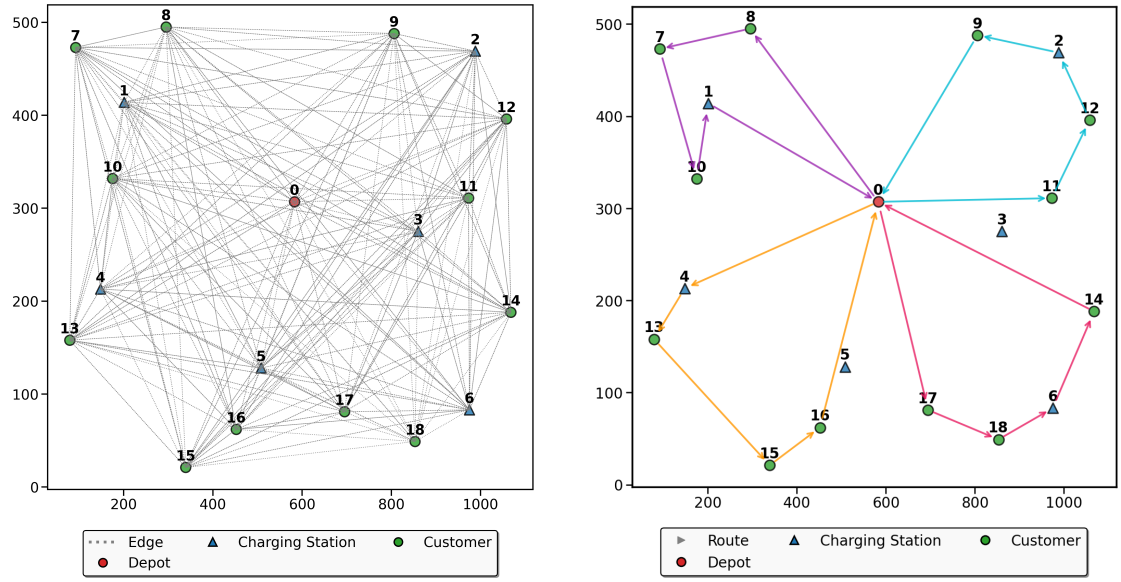
$$u_{ij} + v_{ij} \leq C x_{ij} \qquad \forall i \in V'_0, j \in V'_{N+1}, i \neq j \tag{4.12}$$

$$0 \leq y_j \leq y_i - (h d_{ij}) x_{ij} + Q(1 - x_{ij}) \qquad \forall i \in V, \forall j \in V'_{N+1}, i \neq j \tag{4.13}$$

$$0 \leq y_j \leq Y_i - (h d_{ij}) x_{ij} + Q(1 - x_{ij}) \qquad \forall i \in F'_0, \forall j \in V'_{N+1}, i \neq j \tag{4.14}$$

$$y_i \leq Y_i \leq Q \qquad \forall i \in F'_0 \tag{4.15}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V'_0, j \in V'_{N+1}, i \neq j \tag{4.16}$$



(a) EVRP Instance **(b)** Solution

Fig. 4.1 Illustration of an EVRP instance and its solution. (a) presents a map showing the locations of a depot, twelve customers, and six charging stations based on Cartesian coordinates. Gray dashed lines indicate a fully connected graph connecting each pair of nodes. (b) shows a valid solution to the given instance on the same map, with four distinct tours represented by arrows with different colors. All routes begin and end at the depot, passing through various customers and charging stations.

The distance-based objective function from [104] is extended in order to prioritize solutions that utilize fewer vehicles, even if the total distance traveled in such cases is greater than in other solutions. This is done by introducing an additional cost parameter $M > 0$ per vehicle utilized. Note that the number of vehicles used in a solution corresponds to the variables on outgoing arcs of the depot (0) that have a value of 1. In this line, the objective function (4.1) minimizes the total travel and vehicle cost. Constraints (4.2) ensure that each customer is visited by an electric vehicle, while constraints (4.3) allow vehicles to visit a charging station only when required. Constraints (4.4) guarantee that each vehicle that visits a particular node must also depart from the corresponding node. The arrival and departure times are calculated using constraints (4.5) and (4.6), which consider the service and battery charging times. Constraints (4.7) permit vehicles to visit each node within the corresponding time windows. At the same time, constraints (4.5)–(4.7) prevent sub-tours. Constraints (4.8)–(4.12) ensure that the delivery and pickup demands of customers are simultaneously met. Finally, constraints (4.13)–(4.15) are related to the battery state of charge. For an example instance together with a solution, see Figure 4.1.

4.3 Set-Covering Based ILP Model of the EVRP-TW-SPD

Assignment-type ILP models such as the one presented above for the EVRP-TW-SPD generally do not allow to derive good lower bounds (see, for example, [9]). In addition, experiments reported in [5] showed that finding any feasible solution to the corresponding model within reasonable execution times for CPLEX becomes difficult, even in the context of small-sized sub-instances of the original problem instances.

The EVRP-TW-SPD can be modeled in terms of a set-covering-based ILP in the following way. Let $\mathcal{T}$ be the set of all possible (and feasible) tours, where a tour is defined as the trip of one single vehicle returning to the depot from which it originally left. Each tour $T_r \in \mathcal{T}$ is evaluated by the total distance traveled d_r, that is, the sum of the distances of all arcs on the tour. Finally, let $\mathcal{T}_i \subset \mathcal{T}$ be the set of tours that serve customer $i \in V$. With these definitions, the set-covering-based ILP model for the EVRP-TW-SPD, henceforth denoted as $\texttt{ILP}^{\text{EVRP}}_{\text{setcov}}$, can be stated as follows.

$$\textbf{min} \sum_{T_r \in \mathcal{T}} d_r x_r + M \sum_{T_r \in \mathcal{T}} x_r \tag{4.17}$$

$$\textbf{s.t.} \sum_{T_r \in \mathcal{T}_i} x_r \geq 1 \qquad \forall\, i \in V \tag{4.18}$$

$$x_r \in \{0,1\} \qquad \forall\, T_r \in \mathcal{T} \tag{4.19}$$

The objective function minimizes the total travel and vehicle costs and constraints (4.18) ensure that each customer is visited at least once. Note that the set-covering-based formulation is generally used as a post-optimization method in the VRP literature [156]. In contrast to this, our results will show that CMSA provides a suitable algorithmic framework for iteratively applying heuristics and exact components.

4.4 Application of standard Adapt-CMSA to the EVRP-TW-SPD

We first develop an Adapt-CMSA version based on the assignment-type ILP model—that is, model $\texttt{ILP}^{\text{EVRP}}_{\text{std}}$—to the EVRP-TW-SPD. This version of Adapt-CMSA is henceforth labeled Adapt-Cmsa-Std. In the context of Adapt-Cmsa-Std, the complete set of solution components consists of a component c_{ij} for each arc a_{ij} from $A = \{(i,j) | i,j \in V'_{0,N+1}, i \neq j\}$. Consider the following example. The vector $\mathbf{I}$ comprises all the node indexes for a small

problem instance involving three charging stations and five customers. Nodes indexed with 0 and 6 denote the depot.

$$\mathbf{I} = (\underbrace{0,}_{\text{depot}} \underbrace{1,\ 2,\ 3,\ 4,\ 5,}_{\text{customers}} \underbrace{6,}_{\text{depot}} \underbrace{7,\ 8,\ 9}_{\text{charging stations}})$$

Now consider a solution consisting of two tours T_1 and T_2, where $T_1 =<0 \rightarrow 9 \rightarrow 1 \rightarrow 4 \rightarrow 6>$ and $T_2 =<0 \rightarrow 2 \rightarrow 8 \rightarrow 3 \rightarrow 7 \rightarrow 5 \rightarrow 6>$. In the context of Adapt-CMSA-STD, this solution is represented by $S = \{c_{0,9}, c_{9,1}, c_{1,4}, c_{4,6}, c_{0,2}, c_{2,8}, c_{8,3}, c_{3,7}, c_{7,5}, c_{5,6}\}$, that is, a solution S in Adapt-CMSA-STD is kept in terms of the list of solution components representing the arcs used in any of the tours of S.

4.5 The Adapt-Cmsa-Std Algorithm

The pseudo-code presented in Algorithm 4.1 is common to Adapt-CMSA-STD and Adapt-CMSA-SETCOV. It describes the general algorithmic framework of Adapt-CMSA for the EVRP-TW-SPD. First, a feasible solution is generated by calling function GenerateGreedySolution() to initialize the best-so-far solution S_{bsf}. More precisely, this function applies an insertion heuristic which is further explained in Section 4.5.1. Following that, in lines 4 and 5, parameters α_{bsf}, n_a, and l_{size} are given initial values. How these variables are managed within the algorithm will be explained below.

During each iteration of Adapt-CMSA-STD, a sub-instance C' of the original problem instance is created. Similar to the solution representation, a sub-instance is also a set of solution components, that is, $C' \subseteq C$, where C' is initialized to the best solution found so far (S_{bsf}) at the beginning of each iteration. Subsequently, a probabilistic solution construction process shown in lines 8–12 probabilistically generates n_a solutions using function ProbabilisticSolutionConstruction(S_{bsf}, α_{bsf}, l_{size}). This function takes two additional parameters apart from S_{bsf}. These are the parameter α_{bsf} ($0 \leq \alpha_{\text{bsf}} < 1$), which biases the creation of new solutions towards the best-so-far solution, and the parameter l_{size}, which determines the number of options considered at each solution construction step. Remember that higher values of α_{bsf} lead to an increase of similarity between the constructed solutions and S_{bsf}. On the contrary, a higher value of l_{size} results in the construction of more diverse solutions and, consequently, contributes to forming a larger sub-instance.

After constructing a solution S by calling the above-mentioned function in line 9 of Algorithm 4.1, each tour of S undergoes a local search process, as indicated in line 10. This local search procedure applies well-known intra-route

Algorithm 4.1 Pseudo-code of Adapt-CMSA for the EVRP-TW-SPD

1: **input 1:** values for CMSA parameters t_{prop}, t_{ILP}
2: **input 2:** values for solution construction parameters α^{LB}, α^{UB}, α_{red}
3: $S_{\text{bsf}} :=$ GenerateGreedySolution()
4: $\alpha_{\text{bsf}} := \alpha^{\text{UB}}$
5: Initialize$(n_{\text{a}}, l_{\text{size}})$
6: **while** CPU time limit not reached **do**
7: $C' := S_{\text{bsf}}$
8: **for** $i := 1, \ldots, n_{\text{a}}$ **do**
9: $S :=$ ProbabilisticSolutionConstruction(S_{bsf}, α_{bsf}, l_{size})
10: LocalSearch1(S)
11: **for** all $c \in S$ **and** $c \notin C'$ **do** $C' := C' \cup \{c\}$ **end for**
12: **end for**
13: $(S'_{\text{opt}}, t_{\text{solve}}) :=$ SolveSubinstance(C', t_{ILP}) {This function returns two objects: (1) the obtained solution (S'_{opt}), (2) the required computation time (t_{solve})}
14: LocalSearch2(S'_{opt})
15: **if** $t_{\text{solve}} < t_{\text{prop}} \cdot t_{\text{ILP}}$ and $\alpha_{\text{bsf}} > \alpha^{\text{LB}}$ **then** $\alpha_{\text{bsf}} := \alpha_{\text{bsf}} - \alpha_{\text{red}}$ **end if**
16: **if** $f(S'_{\text{opt}}) < f(S_{\text{bsf}})$ **then**
17: $S_{\text{bsf}} := S'_{\text{opt}}$
18: Initialize$(n_{\text{a}}, l_{\text{size}})$
19: **else**
20: **if** $f(S'_{\text{opt}}) > f(S_{\text{bsf}})$ **then**
21: **if** $n_a = n^{\text{init}}$ **then** $\alpha_{\text{bsf}} := \min\{\alpha_{\text{bsf}} + \frac{\alpha_{\text{red}}}{10}, \alpha^{\text{UB}}\}$ **else** Initialize$(n_{\text{a}}, l_{\text{size}})$ **end if**
22: **else**
23: Increment$(n_{\text{a}}, l_{\text{size}})$
24: **end if**
25: **end if**
26: **end while**
27: **output:** S_{bsf}

operators such as *relocation*, *swap*, and *two_opt* in sequential order. Moreover, the best-improvement strategy is adopted in the context of the applied operators. The so-called *relocation* operator sequentially extracts each node from its existing position within a route and repositions it at an alternative location inside the same route. On the other hand, the *swap* operator works by interchanging the positions of a pair of selected nodes belonging to the same route. Lastly, the *two_opt* neighborhood explores every feasible combination of choosing two non-adjacent nodes in the same route and then reverses the arrangement of the nodes situated between the chosen pair of nodes.

Upon the application of local search, the so-called *merge step* is executed in line 11 in the same way as in any other CMSA algorithm. After probabilistically constructing n_{a} solutions and forming the sub-instance C', the sub-instance is

solved by first generating a corresponding ILP model based on model $\texttt{ILP}_{\text{std}}^{\text{EVRP}}$ and then solving the model with a CPU time limit of t_{ILP} seconds with CPLEX in function SolveSubinstance(C', t_{ILP}). In order to generate this model, the following constraints are added to $\texttt{ILP}_{\text{std}}^{\text{EVRP}}$:

$$x_{ij} = 0 \quad \text{for all } c_{ij} \in C \setminus C' \tag{4.20}$$

In other words, if an arc a_{ij} has not been used in any of the solutions that were merged into C', using this arc is forbidden by fixing the value of x_{ij} to zero. It is important to note that incorporating more constraints into the original ILP reduces the search space of the resulting ILP model, thereby facilitating CPLEX's ability to generate a high-quality solution or even the optimal one for the corresponding sub-instance. However, note that—due to the employed CPU time limit for each application of CPLEX—the output of function SolveSubinstance(C', t_{ILP}), denoted as S'_{opt}, is not necessarily an optimal solution to the sub-instance. In any case, S'_{opt} is subject to the application of a local search method different from the one described before. In particular, this local search procedure utilizes inter-tour neighborhoods such as *exchange (1,1)* and *shift (1,0)*. The *exchange (1,1)* neighborhood investigates all potential two-customer swaps not part of the same tour, whereas the *shift (1,0)* neighborhood examines every option for removing a customer from its existing tour and placing it at any possible location in other tours. As is done within LocalSearch1(S), operators used by LocalSearch2(S) employ the best-improvement search strategy.

The self-adaptive nature of Adapt-CMSA-STD can be found in the dynamical adjustment of the values of parameters α_{bsf}, n_{a}, and l_{size}. In other words, for this application of Adapt-CMSA an additional parameter, l_{size} is handled in a dynamic way. As already explained in Chapter 2, the value of the dynamic parameter α_{bsf} is bounded from below by α^{LB} and from above by α^{UB}. Both α^{LB} and α^{UB} are input parameters of the algorithm. Moreover, the value of a step size parameter α_{red} is employed for systematically reducing the α_{bsf}'s value, if needed. Initially, the value of α_{bsf} is set to the highest possible value, α^{UB}, as shown in line 4.[1] If the resulting ILP is solved within a computation time t_{solve} that is below a proportion t_{prop} of the maximum possible computation time t_{ILP}—that is, if $t_{\text{solve}} \leq t_{\text{prop}} \cdot t_{\text{ILP}}$—$\alpha_{\text{bsf}}$'s value is reduced by α_{red}, as seen in line 15. The reasoning behind this step is as follows. If the resulting ILP can be easily solved to optimality for the respective sub-instance, the search space is too small, owing to

[1] Remember that solutions constructed with a high value of α_{bsf} will be rather similar to the best-so-far solution S_{bsf}.

a relatively low number of free variables. To increase the number of free variables in the ILP, solutions produced in ProbabilisticSolutionConstruction(S_{bsf}, α_{bsf}, l_{size}) should differ more from S_{bsf}, which is achievable by lowering the value of α_{bsf}.

The adjustment of parameters n_{a} and l_{size} follows a similar scheme as the one described above. Their initial values are set as follows: $n_{\text{a}} := n^{\text{init}}$ and $l_{\text{size}} = l_{\text{size}}^{\text{init}}$, which is done in the Initialize(n_{a}, l_{size}) function. This function can be called under three distinct circumstances: (1) at the beginning of the algorithm (line 5), (2) when solution S'_{opt} is strictly better than S_{bsf} (line 18), and (3) when solution S'_{opt} is strictly worse than S_{bsf} while n_{a} concurrently exceeds n^{init} (line 21). On the other hand, when S'_{opt} and the S_{bsf} have the same objective function value, the algorithm has the capacity to create larger sub-instances, leading to an increase in the values of the three parameters in Increment($n_{\text{a}}, l_{\text{size}}$) function. Specifically, n_{a} is increased by n^{inc}, and l_{size} is increased by $l_{\text{size}}^{\text{inc}}$.

Figure 4.2 graphically summarizes the structure and components of Adapt-CMSA-STD for EVRP-TW-SPD.

4.5.1 Probabilistic Solution Construction

The call of function ProbabilisticSolutionConstruction(S_{bsf}, α_{bsf}, l_{size}) invokes the execution of one of two heuristics: either (1) a version of the Clarke-Wright Savings Algorithm [41], or (2) a variant of the insertion algorithm. The choice of a heuristic is done uniformly at random. Both heuristics exclusively generate feasible solutions. In the following, both construction algorithms and their variants are described in detail.

Probabilistic Clarke-Wright Savings Algorithm. Similar to the original Clarke-Wright approach, our algorithm variant begins by generating a set of direct routes, denoted as $R = \{(0 \rightarrow i \rightarrow (N+1)) \mid i \in V\}$. Next, the algorithm initializes a savings list L consisting of pairs of nodes (i, j), where i and j represent customers and charging stations. The savings value σ_{ij} for each pair is computed using the following equation:

$$\sigma_{ij} := d_{0i} + d_{0j} - \lambda d_{ij} + \mu |d_{0i} - d_{0j}| \tag{4.21}$$

Here, λ and μ are the so-called *route shape* and *asymmetry scaling* parameters, respectively. The route shape parameter λ prioritizes the selection of nodes based on their distance from each other [193]; parameter μ, on the other hand, scales the asymmetry between nodes i and j [143]. Well-working values for these

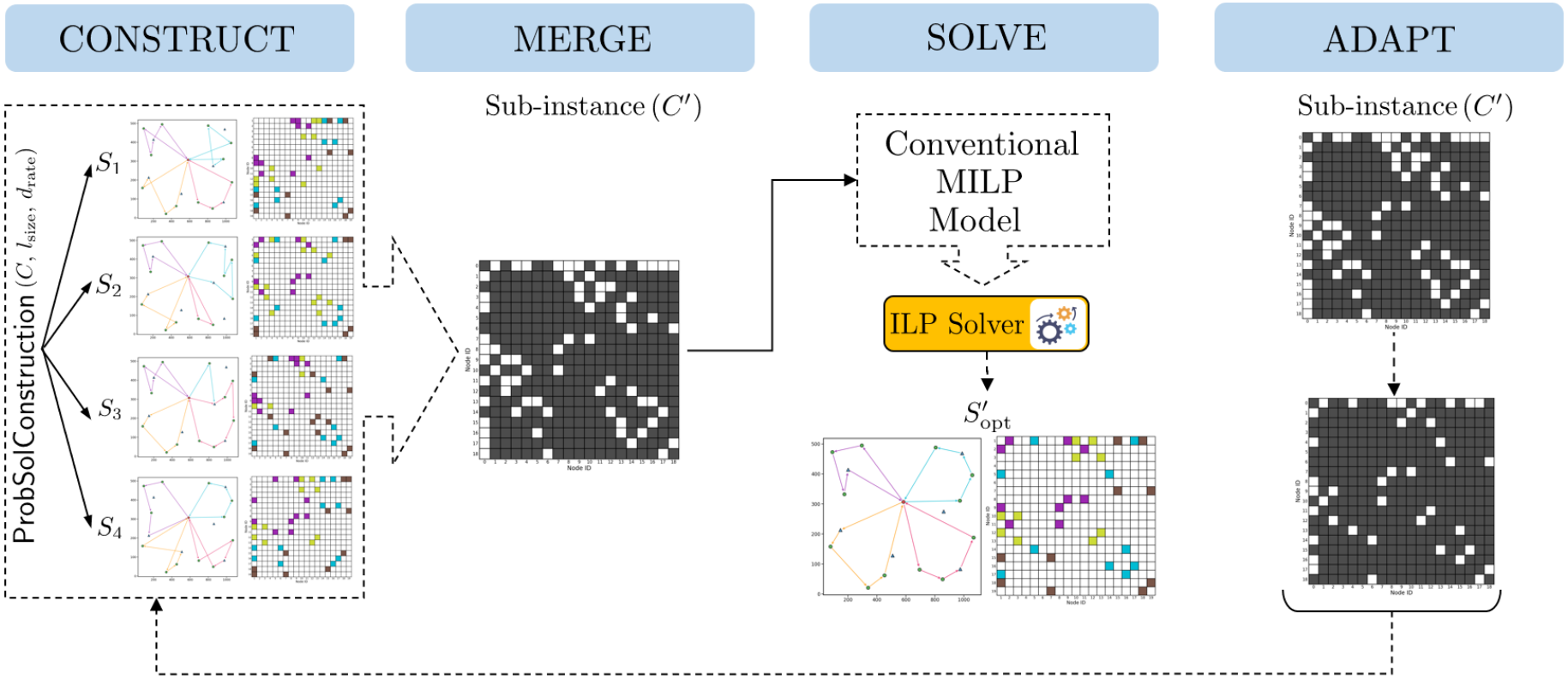


Fig. 4.2 Graphical abstract of Adapt-CMSA-STD for EVRP-TW-SPD

This graphical abstract illustrates the Adapt-CMSA-STD for EVRP-TW-SPD. Four example solutions are probabilistically generated in the "Construct" step, each shown on a map and corresponding binary matrices. Note that the colourful cells in these matrices indicate edges involved in the respective solution ($x_{ij} = 1$), while white cells denote unselected edges ($x_{ij} = 0$). During the "Merge" step, solution components of these solutions are combined to form a sub-instance, illustrated by a two-dimensional binary matrix where white cells represent edges that are part of at least one solution, and grey cells indicate those that never appeared in any solution. Subsequently, in the "Solve" step, the sub-instance is solved using an assignment-type MILP model by an ILP solver to yield S'_{opt}. Finally, the "Adapt" step updates the sub-instance based on S'_{opt}, retaining only the edges present in S'_{opt} in the sub-instance.

parameters are identified through a parameter tuning procedure described in Section 4.7.2. It is important to note that L only includes pairs of nodes (i, j) that satisfy two conditions: (1) i and j are part of two different tours, and (2) both i and j must be adjacent to the depot in the tour of which they form part. Moreover, solution construction will not only be influenced by the savings values of node pairs (i, j) but also by the fact whether or not arc a_{ij} appears in the current best-so-far solution S_{bsf}. For this purpose, an additional value, q_{ij}, is calculated for each entry $(i, j) \in L$:

$$q_{ij} := \begin{cases} (\sigma_{ij} + 1) \cdot \alpha_{\text{bsf}} & \text{if } c_{ij} \in S_{\text{bsf}} \\ (\sigma_{ij} + 1) \cdot (1 - \alpha_{\text{bsf}}) & \text{otherwise} \end{cases} \tag{4.22}$$

The algorithm performs the following sequence of steps until the savings list L is empty.

1. After computing q_{ij} for all entries in L, the list is sorted in non-increasing order with respect to the q_{ij} values, and a reduced list L_r is created, containing the first l_{size} elements of L.

2. Next, an entry (i, j) is chosen from L_r with respect to the following probabilities:

$$\mathbf{p}(ij) := \frac{q_{ij}}{\sum_{(i',j') \in L_r} q_{i'j'}} \quad \forall\, (i, j) \in L_r \tag{4.23}$$

   Note that the higher the value of α_{bsf}, where $0 \leq \alpha^{\text{LB}} \leq \alpha_{\text{bsf}} \leq \alpha^{\text{UB}} \leq 1$, the higher the probability of selecting arcs that are part of the best-so-far solution S_{bsf}.

3. Then, the chosen tours corresponding to nodes i and j are merged. The merging process is determined by one of the following four possible cases, depending on the direct connection of nodes i and j to the depot:

   (a) **Case 1:**

   - $T_1 :< 0 \rightarrow i \rightarrow \ldots \rightarrow N+1 >$, $T_2 :< 0 \rightarrow j \rightarrow \ldots \rightarrow N+1 >$
   - Merging: Reverse T_1, $\mathsf{rev}(T_1)$, and concatenate with T_2
   - Result: $T_m :< 0 \rightarrow \ldots \rightarrow i \rightarrow j \rightarrow \ldots \rightarrow N+1 >$

   (b) **Case 2:**

   - $T_1 :< 0 \rightarrow i \rightarrow \ldots \rightarrow N+1 >$, $T_2 :< 0 \rightarrow \ldots \rightarrow j \rightarrow N+1 >$
   - Merging: Reverse both T_1 and T_2, $\mathsf{rev}(T_1)$, $\mathsf{rev}(T_2)$, and concatenate
   - Result: $T_m :< 0 \rightarrow \ldots \rightarrow i \rightarrow j \rightarrow \ldots \rightarrow N+1 >$

(c) **Case 3:**

- $T_1 :< 0 \rightarrow \ldots \rightarrow i \rightarrow N+1 >$, $T_2 :< 0 \rightarrow j \rightarrow \ldots \rightarrow N+1 >$
- Merging: Concatenate T_1 and T_2
- Result: $T_m :< 0 \rightarrow \ldots \rightarrow i \rightarrow j \rightarrow \ldots \rightarrow N+1 >$

(d) **Case 4:**

- $T_1 :< 0 \rightarrow \ldots \rightarrow i \rightarrow N+1 >$, $T_2 :< 0 \rightarrow \ldots \rightarrow j \rightarrow N+1 >$
- Merging: Reverse T_2, $\mathsf{rev}(T_2)$, and concatenate with T_1
- Result: $T_m :< 0 \rightarrow \ldots \rightarrow i \rightarrow j \rightarrow \ldots \rightarrow N+1 >$

Depending on the positions of nodes i and j in the tour, it may be required to reverse one or both of the tours selected to ensure a direct connection from i to j. In such a case, the reversed form of tour T_1 is represented by $\mathsf{rev}(T_1)$. Subsequently, the feasibility of the merged tour T_m is checked in terms of vehicle loading capacity and time windows. If the obtained route violates vehicle capacity and/or time window constraints, it is deemed infeasible and eliminated from the savings list. A new candidate is then chosen following the procedure already outlined above. In the event that the merged tour is not feasible due to battery constraints, a charging station is inserted into the tour. Determining the optimal charging station location involves identifying the first node in the tour to which the electric vehicle has arrived with a negative battery level. Then, a charging station is inserted between this node and the previous node. After determining the insertion position, the charging station that results in the minimum amount of increase in the overall tour distance is selected and placed into the predetermined position. If the tour remains infeasible, then the same procedure is applied to the previous arcs. In those cases in which the infeasibility persists even after attempting to insert charging stations, the merged tour is discarded, and the associated nodes are taken out of the savings list. Then, the next candidate, pair of nodes, is selected from the savings list following the procedure described above. The tour merging process is repeatedly executed until the saving list is exhausted. Once the merging phase is complete, some of the previously added charging stations may no longer be necessary. Therefore, redundant charging stations are first identified and then removed from the constructed tours.

4. Finally, the savings list L is updated as described above.

As a last step, the final set of tours is converted into its corresponding set of

solution components.

Probabilistic Insertion Algorithm. Our second constructive heuristic operates by inserting customers into available tours in a sequential manner until all customers are visited. The first customer to be inserted into the tour is chosen based on the distance from the depot or the latest possible visiting time. In particular, the initial tour is established by inserting the customer with either the greatest distance from the depot or the earliest deadline. We then produce a cost list that outlines all possible insertion points for each unvisited customer, along with the associated costs. To determine the cost of inserting a customer at a particular point, we use the following equation, which calculates the cost of inserting customer i between nodes j and k

$$c(j,i,k) = d_{ji} + d_{ik} - d_{jk} \tag{4.24}$$

Then, q_{jik} is calculated for each entry $(j,i,k) \in L$ as follows:

$$q_{jik} := \begin{cases} (c(j,i,k)+1) \cdot (1-\alpha_{\text{bsf}})(1-\alpha_{\text{bsf}}) & \text{if } c_{ji} \in S_{\text{bsf}} \text{ and } c_{ik} \in S_{\text{bsf}} \\ (c(j,i,k)+1) \cdot (\alpha_{\text{bsf}})^2 & \text{if } c_{ji} \notin S_{\text{bsf}} \text{ and } c_{ik} \notin S_{\text{bsf}} \\ (c(j,i,k)+1) \cdot \alpha_{\text{bsf}}(1-\alpha_{\text{bsf}}) & \text{otherwise} \end{cases} \tag{4.25}$$

Subsequently, the selection of an entry (j,i,k) from the generated list is carried out based on the probabilities calculated using Eq. (4.25). If the capacity of the vehicle permits, the customer is added to the corresponding location in the tour. In addition, if the insertion is infeasible in terms of battery restrictions, a charging station is inserted into the tour using the process described in the Clarke-Wright Savings Algorithm. In situations where the insertion of a customer results in the vehicle exceeding its load capacity or battery capacity (even after charging station insertion) or causes a time window violation, a new tour is initiated, which includes only the respective customer.

After inserting all of the customers and a complete solution is derived, the obtained set of tours is transformed into the corresponding set S of solution components.

4.6 The Adapt-Cmsa-SetCov Algorithm

The ILP model for solving sub-instances in this variant of Adapt-CMSA is model $\texttt{ILP}^{\text{EVRP}}_{\text{setcov}}$ from Section 4.3. In this case, the complete set of solution components

C consists of a component c_r for each valid tour $T_r \in \mathcal{T}$ (see Section 4.3), that is, $C := \{c_r \mid T_r \in \mathcal{T}\}$. Any subset $S \subset C$ such that each customer $i \in V$ is served by exactly one tour of S is a valid solution to the EVRP-TW-SPD problem instance.

The probabilistic solution construction process in Adapt-CMSA-SETCOV works in exactly the same way as in Adapt-CMSA-STD. Just that the solutions returned consist of solution components that directly correspond to tours (instead of arcs as in the case of Adapt-CMSA-STD).

Another difference is—as mentioned above—the ILP model used to solve sub-instances. In fact, given a sub-instance C', the corresponding ILP model is obtained by replacing each occurrence of $\mathcal{T}$ with C', that is, the model only considers those tours as eligible tours that appear sub-instance C'. However, before returning the solution, it is checked for duplicate occurrences of customers, that is, all redundant customers are initially identified. Afterward, the benefit of removing each redundant customer, which is directly related to the distance of the respective customer with the adjacent nodes, is computed. Subsequently, redundant customers, beginning with the one with the greatest benefit, are removed until each customer appears in a single tour only.

Figure 4.3 illustrates the details of Adapt-CMSA-SETCOV for EVRP-TW-SPD.

4.7 COMPUTATIONAL EXPERIMENTS

The experiments were conducted on the same machines as the ones for the MPIDS problem, that is, on a cluster of machines equipped with Intel® Xeon® 5670 CPUs having 12 cores of 2.933 GHz and at least 32 GB of RAM. Moreover, sub-instances in both Adapt-CMSA variants were solved using CPLEX version 20.1 in one-threaded mode. Furthermore, the ILP models representing complete problem instances were solved using CPLEX version 20.1 in standalone mode.

4.7.1 Generation of the problem instances for EVRP-SPD-TW

The algorithm's performance was evaluated utilizing the EVRP-TW problem instances derived by [166] from the classical VRPTW instances by [173]. This dataset includes a total of 92 instances, consisting of 36 small-sized instances and 56 large-sized instances. Small-sized instances include 5, 10, and 15 customers, while large-sized instances contain 100 customers and 21 charging stations. These instances are organized into three distinct groups based on the spatial distribution of customer locations: clustered instances (marked by the prefix "c"), randomly distributed instances (prefix "r"), and a hybrid of random and clustered distributions (prefix "rc"). Each group further contains two sub-classes (type 1,

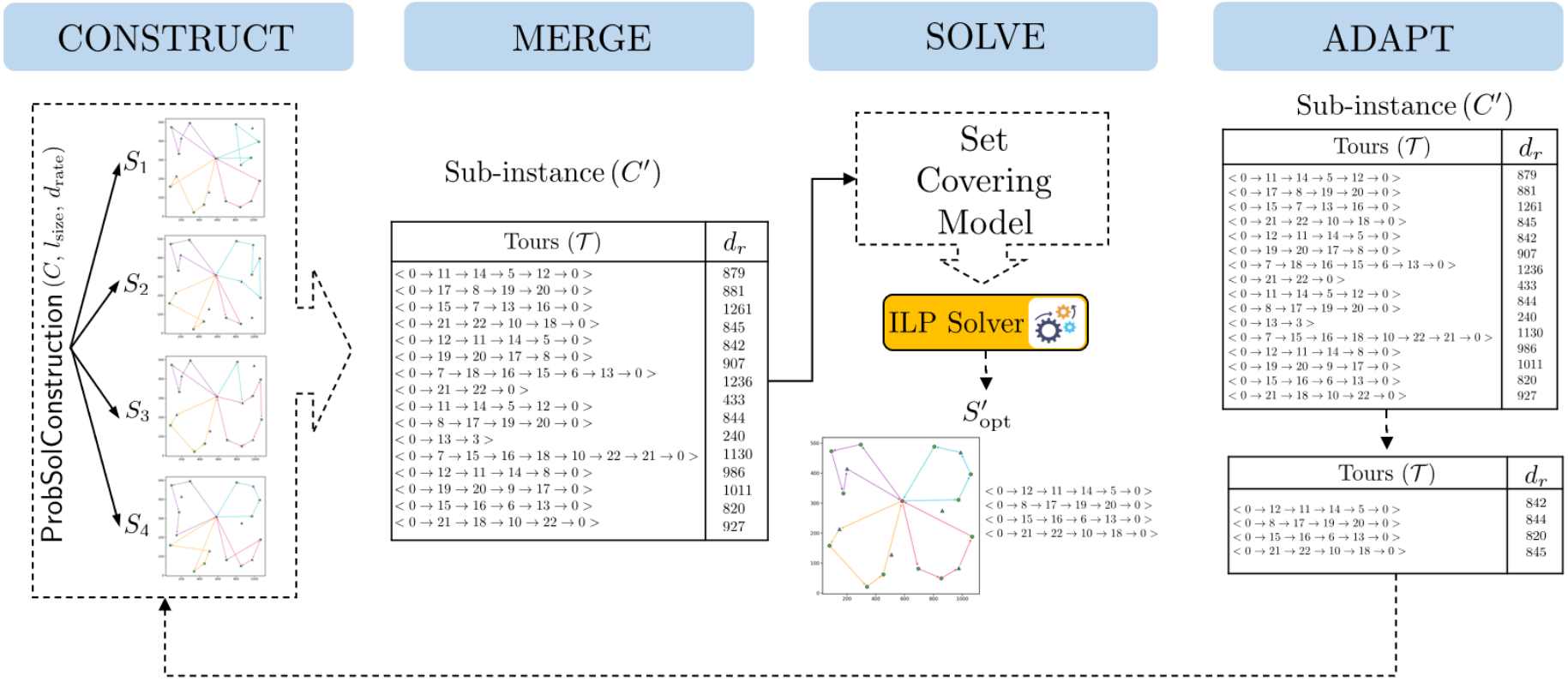


Fig. 4.3 Graphical abstract of Adapt-CMSA-SETCOV for EVRP-TW-SPD

Similar to the Adapt-CMSA-STD, four example solutions are probabilistically generated in the "Construct" step. A noticeable difference is that in this variant, solution components are complete tours undertaken by each vehicle. Thus, each solution is represented as a set of routes instead of a set of edges. During the "Merge" step, components from these solutions are combined to form a sub-instance, which includes all routes from the generated solutions. In the "Solve" step, a set covering-based ILP model is employed to select the most promising set of routes such that every customer is covered by at least one of the selected routes. After deriving S'_{opt}, the "Adapt" step updates the sub-instance by removing routes that are not part of S'_{opt}.

Table 4.1 Parameters, their domains, and the chosen values as determined by irace.

Parameter	Domain	Adapt-CMSA-STD	Adapt-CMSA-SETCOV	Description
λ	$[1, 2]$	1.99	1.38	route shape parameter (C&W alg.)
μ	$[0, 1]$	0.23	0.58	asymmetry scaling (C&W alg.)
$l_{\text{size}}^{\text{init}}$	$\{3, 5, 10, 15, 20, 50, 100, 200\}$	100	10	initial list size value
$l_{\text{size}}^{\text{inc}}$	$\{3, 5, 10, 15, 20, 50, 100, 200\}$	15	20	list size increment
n^{init}	$\{1, 3, 5, 10, 50, 100, 200, 300, 500\}$	1	10	initial nr. of constructed solutions
n^{inc}	$\{1, 3, 5, 10, 50, 100, 200, 300, 400\}$	1	50	increment for the nr. of constr. solutions
t_{ILP}	$\{5, 7, 10, 15, 20, 25, 30, 35, 40\}$	40	20	CPLEX time limit (seconds)
α^{LB}	$[0.6, 0.99]$	0.92	0.75	lower bound for α_{bsf}
α^{UB}	$[0.6, 0.99]$	0.98	0.86	upper bound for α_{bsf}
α_{red}	$[0.01, 0.1]$	0.07	0.07	step size reduction for α_{bsf}
t_{prop}	$[0.1, 0.8]$	0.17	0.23	control parameter for bias reduction

respectively type 2) which differentiate instances based on factors such as time windows, vehicle load, and battery capacity.

Schneider's modifications primarily involved integrating charging stations and adjusting battery capacities to ensure instance feasibility. Specifically, one charging station was positioned at the depot, with the remaining stations distributed randomly, yet in such a way that every customer could be reached using at most two charging stations. The battery capacity was determined as the maximum of (1) the need to travel 60% of the average route length of the best-known solution and (2) twice the battery capacity required to traverse the longest arc from a customer to a charging station. These changes also caused the creation of new time windows, as the original ones from Solomon became infeasible due to added constraints related to charging times.

Since Schneider's instances only provided a single demand type per customer, we adapted these to fit the requirements of our EVRP-SPD-TW model. This adaptation involved separating the combined delivery and pickup demands. We applied the method described by [161] to calculate a ratio $\rho_i = \min\{\frac{x_i}{y_i}, \frac{y_i}{x_i}\}$ using the Cartesian coordinates (x_i, y_i) of each customer $i \in V$. The delivery demand q_i was then computed by multiplying the original demand δ_i by ρ_i, and the pickup demand p_i was obtained by subtracting q_i from δ_i. These modified instances are available at: `https://github.com/manilakbay/EVRP-TW-SPD-Instances`, accessed on (25/04/2024).

4.7.2 Parameter Tuning

We employed the scientific tuning software irace [124] to derive well-working parameter values for Adapt-CMSA-STD and Adapt-CMSA-SETCOV. The tuning process was conducted using six instances, namely `r107`, `r205`, `rc101`, `rc104`, `rc105`, and `rc205`. The budget of irace—that is, the number of algorithm runs

Table 4.2 Computational results for small-sized instances with 5 customers.

Instance	CPLEX				Adapt-CMSA-STD				Adapt-CMSA-SETCOV			
name	m	best	time	gap(%)	m	best	avg.	time	m	best	avg.	time
c101C5	2	2257.75	0.61	0	2	2257.75	2257.75	0.03	2	2257.75	2257.75	0.017
c103C5	1	1175.37	0.58	0	1	1175.37	1175.37	0.77	1	1175.37	1175.37	0.975
c206C5	1	1242.56	0.82	0	1	1242.56	1242.56	0.04	1	1242.56	1242.56	0.006
c208C5	1	1158.48	0.12	0	1	1158.48	1158.48	0.01	1	1158.48	1158.48	0.001
r104C5	2	2136.69	0.03	0	2	2136.69	2136.69	0.10	2	2136.69	2136.69	0.011
r105C5	2	2156.08	0.04	0	2	2156.08	2156.08	0.01	2	2156.08	2156.08	0.001
r202C5	1	1128.78	0.08	0	1	1128.78	1128.78	12.81	1	1128.78	1128.78	0.001
r203C5	1	1179.06	0.04	0	1	1179.06	1179.06	0.32	1	1179.06	1179.06	0.068
rc105C5	2	2233.77	3.10	0	2	2233.77	2233.77	0.13	2	2233.77	2233.77	0.061
rc108C5	2	2253.93	0.27	0	2	2253.93	2253.93	0.01	2	2253.93	2253.93	0.003
rc204C5	1	1176.39	0.36	0	1	1176.39	1176.39	0.10	1	1176.39	1176.39	0.015
rc208C5	1	1167.98	0.17	0	1	1167.98	1167.98	0.74	1	1167.98	1167.98	0.037
average	1.42	1605.57	0.52		1.42	1605.57	1605.57	1.25	1.42	1605.57	1605.57	0.100

allowed for tuning—was set to 2500, and the time limit per instance was fixed to 900 CPU seconds. Moreover, the precision of irace was fixed to two positions behind the comma for numerical parameters. Table 4.1 presents a summary of the parameters, their domains, and the final values selected for the experimentation.

It is worth highlighting that the obtained values for n^{init} and n^{inc} are significantly smaller in the context of Adapt-CMSA-STD when compared to those for Adapt-CMSA-SETCOV. One possible explanation for this observation is that the ILP model used within Adapt-CMSA-STD makes it difficult for the algorithm to be successful. It seems as if the sub-instances are required to be as small as possible so that valid solutions can be generated by CPLEX when solving these sub-instances in a restricted time. This explanation is also supported by the obtained values for t_{ILP}. The limit for the running time of CPLEX for solving the ILP models of each iteration is about twice as high in the case of Adapt-CMSA-STD. Contrary to this, the value of the $l_{\text{size}}^{\text{init}}$ parameter determined for Adapt-CMSA-STD is much higher than that determined for Adapt-CMSA-SETCOV. Higher $l_{\text{size}}^{\text{init}}$ may be considered as a diversification mechanism, as compensation for dealing with small sub-instances.

4.7.3 Numerical Results

In this section, we provide a detailed experimental evaluation of the proposed algorithms and study their performance in various scenarios. To gain a better understanding of how they perform in different situations, we tested them on small-sized instances with 5, 10, and 15 customers, as well as larger-sized instances with 100 customers. The numerical results for the small-sized instances can be found in Tables 4.2–4.4, while the results for the larger-sized instances are presented in Tables 4.5–4.7.

To assess the effectiveness of the algorithms in handling small problem

Table 4.3 Computational results for small-sized instances with 10 customers.

Instance	CPLEX				Adapt-Cmsa-Std				Adapt-Cmsa-SetCov			
name	m	best	time	gap(%)	m	best	avg.	time	m	best	avg.	time
c101C10	3	3388.25	109.23	0	3	3388.25	3388.25	0.40	3	3388.25	3388.554	0.464
c104C10	2	2273.93	3.08	0	2	2273.93	2273.93	0.68	2	2273.93	2273.93	41.311
c202C10	1	1304.06	9.02	0	1	1304.06	1304.06	0.64	1	1304.06	1304.06	0.142
c205C10	2	2228.28	0.12	0	2	2228.28	2228.28	15.40	2	2228.28	2228.28	0.047
r102C10	3	3249.19	0.80	0	3	3249.19	3249.19	25.93	3	3249.19	3249.19	0.012
r103C10	2	2206.12	24.90	0	2	2206.12	2206.12	7.50	2	2206.12	2206.12	31.325
r201C10	1	1241.51	87.11	0	1	1241.51	1241.51	32.32	1	1241.51	1241.51	22.727
r203C10	1	1218.21	2.23	0	1	1218.21	1218.21	20.02	1	1218.21	1218.21	38.672
rc102C10	4	4423.51	0.21	0	4	4423.51	4423.51	0.34	4	4423.51	4423.51	0.020
rc108C10	3	3345.93	5.29	0	3	3345.93	3345.93	20.80	3	3345.93	3345.93	0.019
rc201C10	1	1412.86*	32393.00	16.64	1	1412.86	1412.86	2.80	1	1412.86	1412.86	1.325
rc205C10	2	2325.98	0.18	0	2	2325.98	2325.98	0.19	2	2325.98	2325.98	0.635
average	2.08	2384.82	2719.60		2.08	2384.82	2384.82	10.59	2.08	2384.82	2384.84	11.392

*The result for the CPU time limit of 9 hours.

Table 4.4 Computational results for small-sized instances with 15 customers.

Instance	CPLEX				Adapt-Cmsa-Std				Adapt-Cmsa-SetCov			
name	m	best	time	gap(%)	m	best	avg.	time	m	best	avg.	time
c103C15	3	3348.46	7183.45	7.3	3	3348.46	3348.47	68.59	3	3348.46	3348.46	5.348
c106C15	3	3275.13	1.28	0	3	3275.13	3275.13	6.49	3	3275.13	3275.13	46.034
c202C15	2	2383.62	62.26	0	2	2383.62	2383.62	18.12	2	2383.62	2393.392	7.497
c208C15	2	2300.55	4.62	0	2	2300.55	2300.55	1.55	2	2300.55	2300.55	12.231
r102C15	5	5412.78	7183.67	20.6	5	5412.78	5412.78	2.59	5	5412.78	5412.78	0.121
r105C15	4	4336.15	7.60	0	4	4336.15	4336.15	1.47	4	4336.15	4336.15	1.219
r202C15	2	2361.51	7181.88	27.3	2	2358.00	2364.90	32.32	1	**1507.32**	**1677.456**	64.285
r209C15	1	1313.24	4396.03	0	1	1313.24	1313.24	17.41	1	1313.24	1313.24	15.623
rc103C15	4	4397.67	349.61	0	4	4397.67	4397.67	0.34	4	4397.67	4397.67	0.302
rc108C15	3	3370.25	1170.76	0	3	3370.25	3370.25	58.85	3	3370.25	3370.25	0.335
rc202C15	2	2394.39	859.43	0	2	2394.39	2394.39	0.68	2	2394.39	2394.39	27.978
rc204C15	1	1403.38	7183.65	28.7	1	**1382.22**	1385.72	68.09	1	**1382.22**	**1382.546**	71.869
average	2.67	3024.76	2965.35		2.67	3022.71	3023.57	23.04	2.58	**2951.82**	**2966.83**	21.070

instances, we compared Adapt-Cmsa-Std and Adapt-Cmsa-SetCov with the application of CPLEX to the full-size instances. However, since CPLEX cannot handle larger problem instances, we used our probabilistic Clarke-Wright Savings Algorithm (pC&W) and our probabilistic sequential insertion algorithm (pSI) as benchmarks for those scenarios. We ensured that the parameters for both algorithms were set in the same way as for their application within Adapt-Cmsa-Std. Finally, we imposed a computation time limit of 150 CPU seconds for small problem instances and 900 CPU seconds for larger problem instances. Each algorithm was applied 10 times to each problem instance. Note also that, to compute objective function values, we set the cost of each vehicle used in a solution to 1000, that is, $M = 1000$.

The first column of each result table presents the instance names, while the columns with heading 'm' show the number of vehicles used in the respective solutions. For algorithms Adapt-Cmsa-Std, Adapt-Cmsa-SetCov, pC&W, and pSI, the columns labeled 'best' display the best objective function values among the solutions obtained after ten runs. Additionally, columns with the heading 'avg.'

show the average objective function values over the best solutions of each of the 10 runs. Moreover, the 'time' columns show the computation time (in seconds) of CPLEX and the average computation times of both Adapt-CMSA variants to generate the best solutions in each run. The time limit for CPLEX was set to two hours. The 'gap(%)' columns provide the percentage difference between the optimal solutions obtained and the best lower bounds achieved by CPLEX. It is worth noting that if the gap value is zero, CPLEX has found an optimal solution.

Based on the obtained results, the following observations can be made. For small-sized problem instances, CPLEX optimally solved 31 instances. However, for the remaining 5 instances (`rc201C10`, `c103C15`, `r102C15`, `r202C15`, `rc204C15`), it only provided feasible solutions. It is worth noting that these solutions were the best found within 2 hours of computation time, except for `rc201C10`, which required 9 hours of running time to derive the presented solution. On the other hand, both versions of Adapt-CMSA found optimal solutions, as proven by CPLEX. In the case of the `r202C15` instance, Adapt-Cmsa-Std and Adapt-Cmsa-SetCov were even able to improve over the solution obtained by CPLEX by 0.15% and 36.17%, respectively. Furthermore, Adapt-Cmsa-Std and Adapt-Cmsa-SetCov could improve the solution obtained by CPLEX by 1.51% in the case of the `rc204C15` instance. Moreover, both variants of Adapt-CMSA require considerably less computation time than CPLEX. More specifically, while CPLEX found its best solutions on average in 2965.35 seconds, Adapt-Cmsa-Std was able to do so in 23.04 seconds, and Adapt-Cmsa-SetCov was able to do so in just 21.07 seconds.

The numerical results for the large-sized instances demonstrate that both variants of Adapt-CMSA are superior to pC&W and pSI in terms of both best-performance and average-performance. Although the average computation time required by Adapt-Cmsa-Std and Adapt-Cmsa-SetCov was higher than that required by pC&W and pSI, this was because pC&W and pSI were not able to improve their best-found solutions any further, after some time, while the Adapt-CMSA algorithms were still able to explore the search space in order find even better solutions. It is also worth noting that solutions found by both versions of Adapt-CMSA utilize fewer vehicles than those found by pC&W and pSI.

In addition, the following observations can be made concerning the comparison of the performances of Adapt-Cmsa-Std and Adapt-Cmsa-SetCov. First, Adapt-Cmsa-SetCov significantly outperforms Adapt-Cmsa-Std in the case of random and random-clustered instances; see below for statistical tests. However, from the results presented in Table 4.5, it seems difficult

to come to a definite conclusion in the context of clustered-type instances. Adapt-Cmsa-SetCov seems to provide a slightly better performance both in terms of best and average results. However, in order to back this claim up in a scientifically well-founded way, we also present critical difference (CD) plots as a statistical tool for assisting in the evaluation of the obtained results. In particular, we used the `scmamp` tool [31] in order to generate the CD plots. In these plots, each algorithm variant is positioned along the horizontal axis according to its average ranking for the considered subset of problem instances. Algorithm variants whose performances fall below the critical difference threshold, computed with a significance level of 0.05, are considered statistically equivalent, as indicated by the horizontal bars connecting their respective markers. According to Figure 4.4, there is not a significant difference between the performance of Adapt-Cmsa-SetCov and Adapt-Cmsa-Std on clustered instances, while both outperform the probabilistic implementations of the construction heuristics.

In summary, both variants of Adapt-CMSA show a very satisfactory performance both in the context of small and large problem instances. Moreover, Adapt-Cmsa-SetCov shows superiority over Adapt-Cmsa-Std, particularly in the context of random and random-clustered instances. These claims are backed up in a statistical way by means of the graphics in Figure 4.4. However, we also observed that the performance of Adapt-Cmsa-SetCov decreases in the context of instances with a long scheduling horizon (`C2*` `R2*` and `RC2*`), see Figure 4.4f. Solutions for those instances include fewer routes and hence more customers per route when compared to the solutions for the instances with short scheduling horizons (`C1*` `R1*` and `RC1*`).

Table 4.5 Computational results for large-sized clustered instances.

Instance	pC&W				pSI				Adapt-CMSA-STD				Adapt-CMSA-SETCOV			
name	m	best	avg.	time	m	best	avg.	time	m	best	avg.	time	m	best	avg.	time
c101	21	22854.30	23028.50	406.70	13	14788.00	15574.68	513.05	12	**13043.40**	**13043.42**	385.13	12	13057.80	13063.54	292.56
c102	19	20764.20	21008.31	350.05	13	14664.70	15366.11	397.20	11	**12056.80**	**12920.23**	560.77	11	12073.10	12944.34	468.81
c103	16	17548.30	17622.53	483.74	12	13641.10	14363.90	491.02	11	12004.70	12026.90	452.13	10	**11134.90**	**11917.80**	718.09
c104	13	14388.30	14428.41	500.50	11	12404.70	13195.66	415.63	10	10872.80	11353.78	629.96	10	**10870.70**	**10876.49**	608.43
c105	19	20679.90	21608.72	588.59	13	14622.90	14928.32	346.88	11	**12023.80**	12341.60	562.10	11	12034.10	**12068.86**	582.74
c106	18	19797.20	20626.70	386.43	13	14713.90	14770.94	349.02	11	**12013.10**	12438.06	652.00	11	12025.70	**12059.29**	434.80
c107	18	19842.00	20594.06	459.03	12	13685.10	14631.25	339.14	11	**12006.40**	**12023.97**	538.41	11	12026.70	12046.38	393.01
c108	16	17589.20	18105.10	402.57	13	14617.00	14693.32	472.70	11	11994.70	12016.10	579.51	10	**11025.80**	**11822.60**	556.58
c109	14	15520.10	16415.00	454.81	12	13534.10	13628.65	353.78	10	11042.20	11885.30	714.89	10	**10941.00**	**11180.77**	746.17
c201	10	11333.60	12051.53	448.41	5	6081.90	6184.08	506.19	4	**4629.95**	**4629.95**	37.59	4	4678.37	4703.43	390.96
c202	8	9256.17	9509.34	490.70	5	6136.20	6232.74	526.69	4	**4629.95**	**4629.95**	273.58	4	4664.26	4706.94	394.84
c203	7	8188.19	8245.88	308.89	5	6247.22	6352.57	573.05	4	**4632.27**	**4690.06**	740.49	4	4641.45	4734.31	497.60
c204	5	6159.23	6200.12	416.22	4	5314.79	5354.18	556.74	4	**4633.08**	**4665.78**	801.76	4	4660.64	4737.07	716.94
c205	8	9209.37	9472.50	412.12	5	6161.46	6277.49	527.66	4	**4629.95**	**4629.95**	76.87	4	**4629.95**	**4629.95**	125.43
c206	6	7234.69	7989.99	460.46	5	6269.37	6305.64	405.68	4	**4629.95**	**4629.95**	213.28	4	**4629.95**	**4629.95**	203.04
c207	7	8062.15	8134.61	397.72	5	6278.60	6329.67	486.32	4	**4629.95**	**4629.95**	255.85	4	**4629.95**	4635.27	260.34
c208	6	7167.66	7674.06	341.68	5	6225.26	6282.06	458.79	4	**4629.95**	**4629.95**	284.78	4	**4629.95**	**4629.95**	261.72
average	12.41	13858.50	14277.37	429.92	8.88	10316.84	10615.96	454.09	7.65	8476.64	8657.94	456.42	7.53	**8373.78**	**8552.17**	450.12

Table 4.6 Computational results for large-sized random instances.

Instance	pC&W				pSI				Adapt-CMSA-STD				Adapt-CMSA-SETCOV			
name	m	best	avg.	time	m	best	avg.	time	m	best	avg.	time	m	best	avg.	time
r101	28	30091.80	30543.42	586.01	21	23068.50	23707.34	389.94	18	**19633.80**	19939.79	653.66	18	19640.60	**19661.15**	678.64
r102	23	24946.40	25955.63	227.18	19	20905.90	21536.14	310.36	17	18470.80	19292.16	707.40	16	**17474.10**	**17696.35**	798.71
r103	19	20733.40	20958.09	397.28	17	18714.90	18773.25	410.8	15	16296.50	17050.75	711.23	14	**15280.30**	**15306.17**	639.83
r104	15	16457.60	16490.35	401.43	15	16583.40	16615.25	447.08	13	14141.10	14255.53	616.85	12	**13084.30**	**13111.31**	766.15
r105	23	24903.20	25040.80	398.35	17	18882.40	19784.06	727.33	15	16389.20	17212.83	680.24	14	**15471.30**	**16346.10**	611.98
r106	20	21799.00	22295.65	407.22	16	17824.50	18599.83	545.85	15	16292.00	16836.67	701.75	14	**15314.80**	**15441.68**	746.22
r107	16	17579.50	18458.44	459.73	15	16617.60	16678.79	278.26	13	14168.90	15016.67	680.99	12	**13140.10**	**13669.50**	783.82
r108	14	15467.70	15979.98	445.48	14	15514.90	15546.85	479.26	12	13079.80	13531.30	667.03	11	**12073.70**	**12998.17**	742.49
r109	18	19684.40	19907.84	575.13	16	17690.00	17740.59	445.28	14	15237.30	15674.51	759.64	13	**14220.80**	**14468.12**	744.57
r110	15	16512.80	17014.28	514.35	15	16540.20	16610.01	532.43	13	14170.20	14905.73	528.09	12	**13114.30**	**13544.76**	770.70
r111	15	16528.30	17349.54	433.79	15	16595.30	16859.04	387.64	12	**13144.20**	14584.19	696.71	12	13148.80	**13965.98**	716.12
r112	14	15452.40	15476.60	456.40	14	15498.00	16239.69	417.67	12	13155.60	14053.56	471.58	12	**13044.10**	**13078.65**	850.31
r201	16	17540.00	17576.79	600.20	4	5786.71	5823.23	570.9	4	**5192.33**	**5216.92**	720.44	4	5276.75	5363.04	187.38
r202	10	11394.70	12692.40	369.71	4	5504.61	5585.48	447.33	3	4250.70	5020.88	688.65	3	**4193.33**	**4940.22**	876.26
r203	8	9207.85	9229.98	442.55	3	4390.30	4539.07	530.06	3	**3942.74**	4352.52	868.05	3	3985.02	**4060.18**	822.50
r204	4	5031.33	5064.99	522.23	3	4194.11	4252.43	341.83	3	3820.72	3854.31	787.19	3	**3793.76**	**3827.81**	876.38
r205	11	12426.50	13326.29	473.80	3	4584.97	4620.48	306.67	3	**4055.28**	**4124.64**	731.94	3	4065.06	4126.45	360.47
r206	9	10252.10	10691.23	418.76	3	4419.12	4513.97	591.64	3	**3978.10**	4065.05	756.40	3	3991.44	**4047.16**	784.07
r207	6	7124.47	7663.63	329.29	3	4245.71	4329.11	466.8	3	**3878.91**	**3910.07**	659.85	3	3881.97	3918.29	878.20
r208	3	4104.27	4912.11	398.70	3	4185.27	4226.41	427.82	3	3791.27	3829.39	849.73	3	**3732.80**	**3776.20**	895.79
r209	8	9304.02	9864.70	498.81	3	4413.41	4456.23	554.27	3	3975.64	4015.90	665.83	3	**3933.55**	**3977.24**	765.83
r210	7	8234.21	9104.97	500.99	3	4415.78	4450.61	463.92	3	**3920.37**	3984.78	755.82	3	3926.79	**3961.42**	740.36
r211	6	7083.79	7224.86	548.55	3	4204.05	4262.28	408.61	3	3814.42	3893.38	825.95	3	**3824.47**	**3857.62**	799.45
average	13.39	14863.47	15340.11	452.43	9.96	11512.16	11728.27	455.73	8.83	9947.82	10374.85	703.70	8.43	**9548.35**	**9788.85**	732.01

Table 4.7 Computational results for large-sized random clustered instances.

Instance	pC&W				pSI				Adapt-CMSA-STD				Adapt-CMSA-SETCOV			
name	m	best	avg.	time	m	best	avg.	time	m	best	avg.	time	m	best	avg.	time
rc101	24	26367.50	26696.36	347.88	20	22358.30	22413.20	522.92	16	**17667.70**	18513.67	718.11	16	17696.20	**17741.63**	629.24
rc102	21	23187.80	24083.73	599.41	19	21221.00	21295.40	493.34	16	17576.80	17909.78	558.28	15	**16558.20**	**16628.46**	601.70
rc103	18	19909.70	20476.46	470.41	17	19012.30	19070.74	397.58	14	15366.90	16245.06	764.45	13	**14358.20**	**14999.16**	691.74
rc104	15	16733.40	16778.44	399.26	15	16853.80	16937.69	511.23	13	14270.50	14315.17	637.05	12	**13222.50**	**13261.65**	698.43
rc105	19	21037.60	21959.83	386.88	18	20123.80	20307.09	568.26	15	16500.90	16933.42	652.30	14	**15470.70**	**15820.90**	639.84
rc106	19	20969.80	21133.97	553.93	18	20083.80	20127.15	443.43	14	15432.50	16069.05	632.16	13	**14448.30**	**15249.17**	578.17
rc107	16	17764.50	17918.18	499.62	15	16903.20	17878.81	442.24	13	14313.40	14437.31	769.62	12	**13276.50**	**13386.51**	738.62
rc108	15	16743.00	16781.76	475.26	15	16891.30	17165.01	468.15	12	13226.00	13891.71	620.56	12	**13184.70**	**13214.50**	717.49
rc201	15	16895.90	18148.54	319.12	5	7137.50	7191.55	359.46	4	**5504.77**	5819.06	703.93	4	5617.75	**5786.48**	322.85
rc202	12	13635.30	13899.58	476.49	4	5871.98	5941.11	495.09	4	**5324.64**	**5442.41**	593.36	4	5436.63	5541.80	196.92
rc203	8	9387.70	10210.36	494.81	4	5677.24	5708.69	452.79	4	5109.88	5177.69	644.21	4	**5086.44**	**5159.15**	757.20
rc204	5	6211.87	6441.15	433.68	4	5471.72	5545.00	540.05	3	4036.49	4525.03	745.44	3	**3962.88**	**4252.28**	899.13
rc205	12	13603.40	13790.39	408.58	4	5776.53	5943.19	470.09	4	**5260.14**	**5338.50**	607.45	4	5285.41	5375.54	314.93
rc206	11	12707.00	13467.00	534.29	4	5805.89	5847.72	586.12	4	5234.55	5289.90	670.19	4	**5210.57**	**5275.72**	562.86
rc207	8	9394.32	9782.01	394.18	4	5624.55	5648.26	567.96	3	**4150.60**	4930.81	694.79	3	4197.81	**4650.01**	864.38
rc208	6	7218.95	7299.79	520.74	3	4482.34	5381.34	428.75	3	3977.50	4046.09	762.57	3	**3920.17**	**4002.79**	750.41
average	14.00	15735.48	16179.22	457.16	10.56	12455.95	12650.12	484.22	8.88	10184.58	10555.29	673.40	8.50	**9808.31**	**10021.61**	622.74

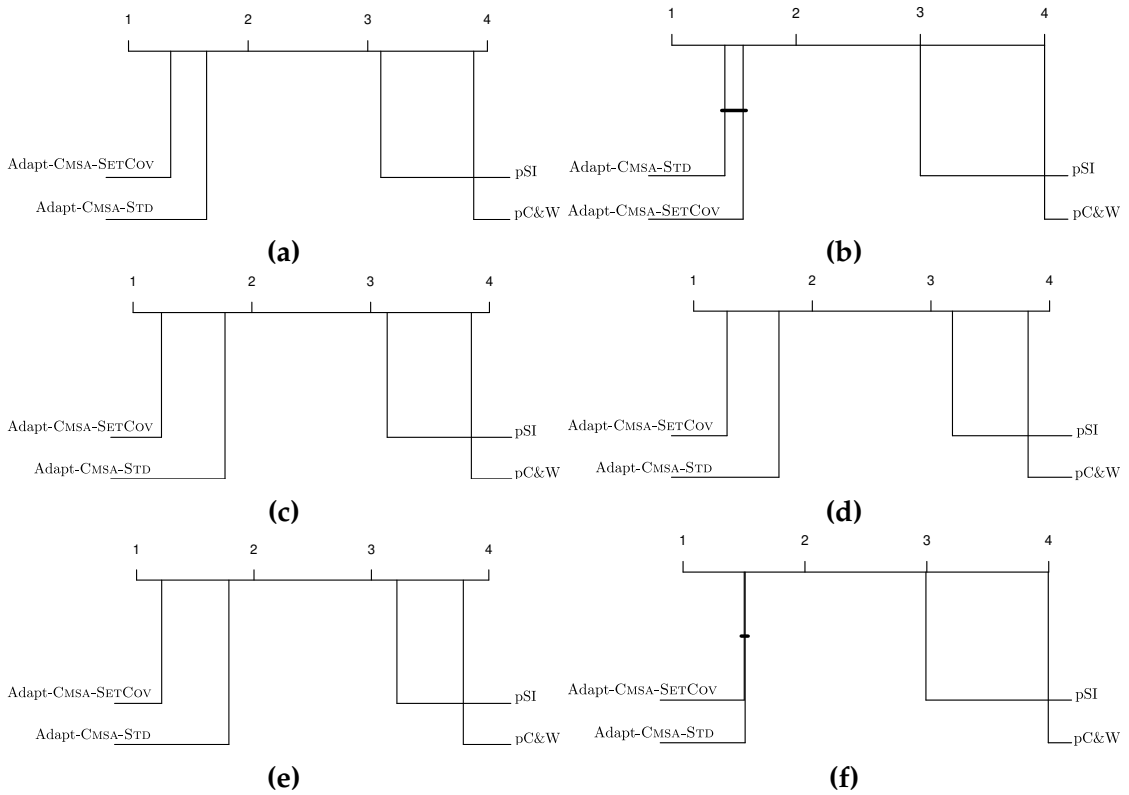
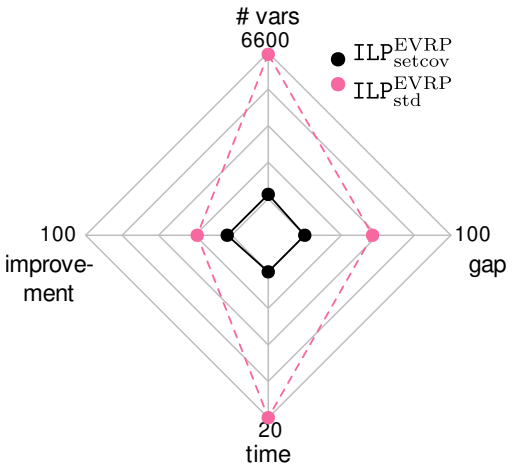


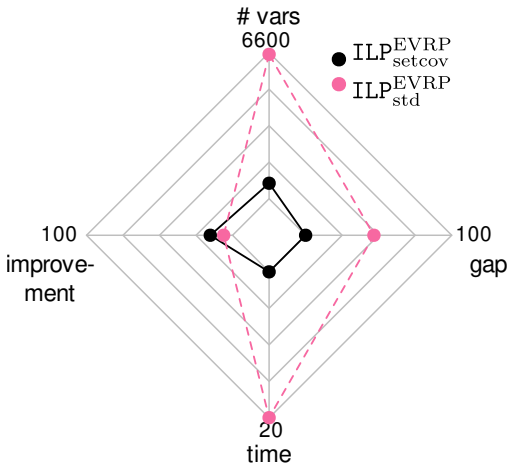
Fig. 4.4 Critical difference (CD) plots concerning the results for large instances. The results in (a) consider all instances together, while the subsequent plots display the results for subsets of the set of large instances: (b) clustered instances; (c) random instances; (d) random-clustered instances; (e) instances `r1*`, `c1*` and `rc1*`; (f) instances `r2*`, `c2*` and `rc2*`.

4.7.4 Performance Difference Between the two EVRP-TW-SPD ILP Models

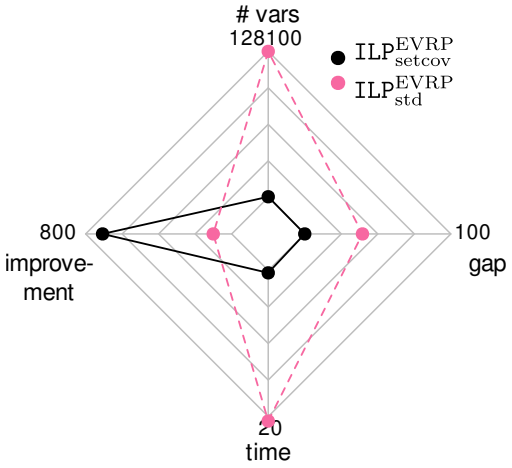
Finally, we want to show why Adapt-CMSA-SETCOV outperforms Adapt-CMSA-STD. For this purpose, we again generate sub-instances of different sizes, translate them both into models $\texttt{ILP}^{\text{EVRP}}_{\text{std}}$ and $\texttt{ILP}^{\text{EVRP}}_{\text{setcov}}$, and solve them with CPLEX. This was done for the small problem instance `r202C15` with 15 customers and for the large problem instance `c101` with 100 customers. In particular, we generated 10 sub-instances by probabilistically constructing 100, respectively 500, solutions (small problem instance) and 50, respectively 100, solutions (large problem instance) and by merging their solution components in order to obtain sub-instances. Figure 4.5 shows radar charts that present the obtained results in the four different cases. Each radar plot provides four different measures, averaged over 10 sub-instances: (1) the number of variables in the models of the sub-instances (top), (2) the relative MIP gap after termination of CPLEX (right), (3) the computation time required by CPLEX (bottom), and (4) the absolute improvement when comparing the result of solving the sub-instance with the



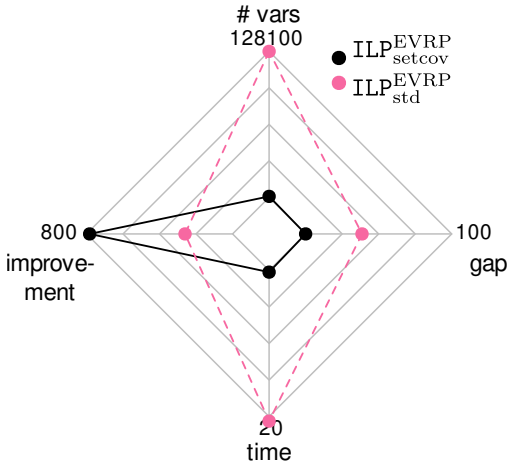
(a) Instance `r202C15` (15 customers), 100 solutions



(b) Instance `r202C15` (15 customers), 500 solutions



(c) Instance `c101` (100 customers), 50 solutions



(d) Instance `c101` (100 customers), 100 solutions

Fig. 4.5 Radar charts concerning the comparison of the two ILP models for the EVRP-TW-SPD problem applied to a small problem instance with 15 customers (see (a) and (b)), and to a large problem instance with 100 customers (see (c) and (d)).

best individual solution that was used to generate the sub-instance. The time limit for CPLEX was set to 20 CPU seconds in all cases. Note that a model is promising if the improvement is large, and the number of variables, the relative MIP gap and the required time are low. The radar charts concerning the large problem instance (see Figures 4.5c and 4.5d) indicate that this is the case for model $\texttt{ILP}^{\text{EVRP}}_{\text{setcov}}$, while the opposite is actually the case for model $\texttt{ILP}^{\text{EVRP}}_{\text{std}}$. Especially the case of the small problem instance considering the lower number of solution constructions (see Figure 4.5a) indicates that sub-instances must not be too small. Otherwise, there might not many improvements to be found in the context of model $\texttt{ILP}^{\text{EVRP}}_{\text{setcov}}$.

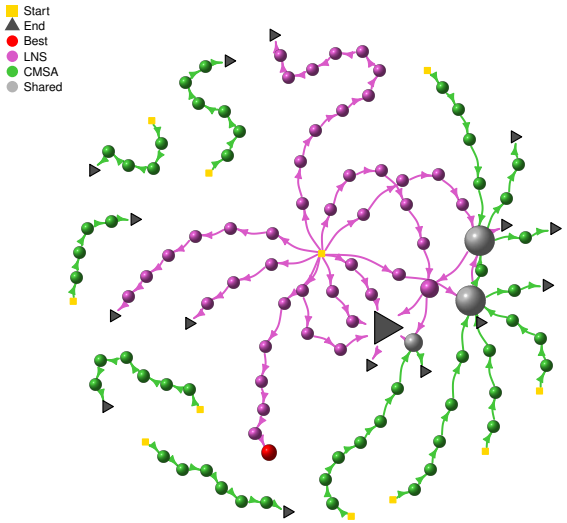


Fig. 4.6 Example of an STN graphic produced by STNWeb.

4.8 Analyzing Algorithm Behaviour Using STNWeb

This section presents an analysis of the algorithmic behaviour of Adapt-Cmsa-Std and Adapt-Cmsa-SetCov using *Search Trajectory Networks* (STNs) [140]. STNs are visualizations of graph objects resulting from repeated applications of optimization algorithms such as metaheuristics [135] to instances of optimization problems. Their purpose is to provide researchers with a tool that allows them to gain a deeper understanding of algorithm behavior. A web-based tool, called STNWeb, for the generation of STN graphics, was presented in [35].

Before delving into the analysis of our algorithm variants we first introduce the essential components of STN graphics. Figure 4.6 provides an example of an STN graphic, demonstrating the performance of two different optimization algorithms for one instance of an optimization problem.

The graphic displays the trajectories resulting from 10 runs of each algorithm. Each dot (node) of the STN represents, in general terms, a chunk of the search space containing at least one solution. The graphical elements in such an STN have the following meaning:

1. Different algorithms' trajectories are depicted in distinct colors, detailed in the legend.
2. Starting points of trajectories are marked with yellow squares.
3. Trajectory endpoints are represented as dark grey triangles, respectively by red dots. While red dots indicate endpoints corresponding to best-found solutions, dark grey triangles correspond to endpoints of worse quality.
4. Pale grey dots represent solutions (resp. chunks of the search space) shared

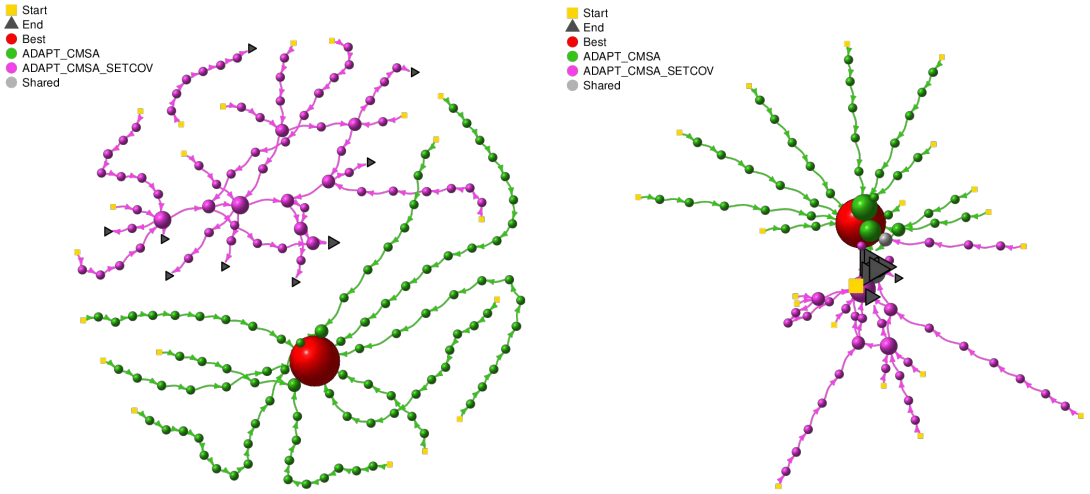
across trajectories of at least two different algorithms.

5. Vertex or dot size indicates the number of algorithm trajectories passing through the vertex: larger vertices indicate more traversing algorithm trajectories.

The graphic in Figure 4.6 shows that five trajectories of LNS (pink trajectories), as well as five trajectories of CMSA (green trajectories), are attracted by the same area of the search space (marked by pale grey vertices). In contrast, trajectory endpoints are rarely shared by different trajectories. This happens only in the case of LNS (see the large dark grey triangle).

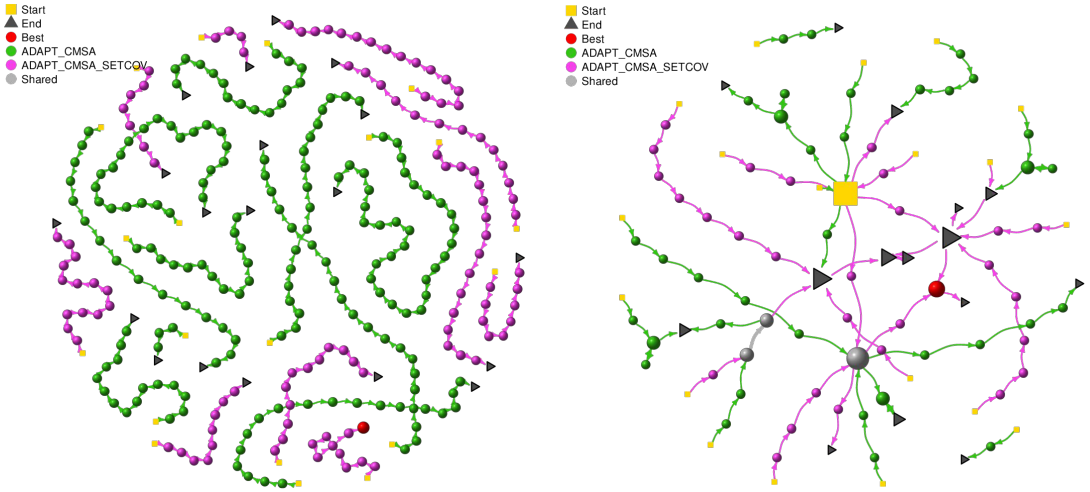
Remember that both Adapt-Cmsa-Std and Adapt-Cmsa-SetCov were applied 10 times for 900 CPU seconds to all problem instances. First, we present our findings on a specific problem instance called `c201`, which is an instance with 100 customers in which customer locations are clustered. STNWeb was used to produce the two STN graphics shown in Figure 4.7. The one in Figure 4.7a shows the original algorithm trajectories, that is, every node corresponds to a solution to the problem. In contrast, the second one (from Figure 4.7b) presents the same STN after search space partitioning. The graphic in Figure 4.7a shows that all 10 runs of Adapt-Cmsa-Std find the best-found solution (large red dot). Further, even though the trajectories of Adapt-Cmsa-SetCov show some overlap, each one of them ends up in solutions of worse quality. Moreover, it is not clear whether or not they end up close to the best-found solution (red dot). The STN after search space partitioning (Figure 4.7b), in contrast, clearly shows that all 10 trajectories of Adapt-Cmsa-SetCov are also attracted by the same area of the search space as the 10 trajectories of Adapt-Cmsa-Std. However, the algorithm does not quite succeed in finding the best solution in that area.

The second example, consisting of the complete STN in Figure 4.8a and the STN after search space partitioning in Figure 4.8b, shows the case of a random-clustered instance (`rc106`) for which Adapt-Cmsa-SetCov works better than Adapt-Cmsa-Std. While the complete STN does not show any trajectory overlaps, the STN after search space partitioning clearly shows that the Adapt-Cmsa-SetCov trajectories end up in the same area of the search space, while the Adapt-Cmsa-Std trajectories have some overlaps with the Adapt-Cmsa-SetCov trajectories, especially in early, resp. intermediate stages, of the search process. However, they simply stop earlier, before reaching the area with the best solutions.



(a) Complete STN **(b)** STN after partitioning

Fig. 4.7 STN graphics concerning the EVRP-TW-SPD. (a) and (b) show 10 runs of Adapt-Cmsa-Std and Adapt-Cmsa-SetCov for instance `c201`. While (a) shows the complete STN, (b) shows the same STN after search space partitioning.



(a) Complete STN **(b)** STN after partitioning

Fig. 4.8 STN graphics concerning the EVRP-TW-SPD. (a) and (b) show 10 runs of Adapt-Cmsa-Std and Adapt-Cmsa-SetCov for instance `rc106`. While (a) shows the complete STN, (b) shows the same STN after search space partitioning.

4.9 Application of Adapt-Cmsa-SetCov to the EVRP-SPD

In an attempt to compare our algorithm with recent work from the related literature, the performance of Adapt-Cmsa-SetCov is tested against 12 different variants of variable neighborhood search (VNS) algorithms from [194]. The EVRP problem considered in [194], denoted by EVRP-SPD, is a reduced version of the EVRP-TW-SPD. In particular, the two problems differ in the following aspects: (1) the objective function of the EVRP-SPD minimizes the total distance

traveled by the electric vehicles; (2) TW constraints are not considered in the EVRP-SPD; and (3) the batteries of the electric vehicles are fully charged when they visit a charging station. Considering these differences, the MILP model of the EVRP-SPD is formulated as follows. Note that this MILP model, which is a two-index model, is equivalent to the three-index model presented in [194].

$$\textbf{Min} \quad \sum_{i \in V_0^{'}, j \in V_{N+1}^{'}} d_{ij} x_{ij} \tag{4.26}$$

$$\sum_{j \in V_{N+1}^{'}, i \neq j} x_{ij} = 1 \qquad \forall i \in V \tag{4.27}$$

$$\sum_{j \in V_{N+1}^{'}, i \neq j} x_{ij} \leq 1 \qquad \forall i \in F^{'} \tag{4.28}$$

$$\sum_{i \in V_0^{'}, i \neq j} x_{ij} - \sum_{i \in V_{N+1}^{'}, i \neq j} x_{ji} = 0 \qquad \forall j \in V^{'} \tag{4.29}$$

$$\tau_i + (t_{ij} + s_i) x_{ij} - l_0(1 - x_{ij}) \leq \tau_j \qquad \forall i \in V_0, j \in V_{N+1}^{'}, i \neq j \tag{4.30}$$

$$\tau_i + t_{ij} x_{ij} + g(Q - y_i) - (l_0 + gQ)(1 - x_{ij}) \leq \tau_j \qquad \forall i \in F^{'}, \forall j \in V_{N+1}^{'}, i \neq j \tag{4.31}$$

$$0 \leq u_{0j} \leq C \qquad \forall j \in V_{N+1}^{'} \tag{4.32}$$

$$v_{0j} = 0 \qquad \forall j \in V_{N+1}^{'} \tag{4.33}$$

$$\sum_{i \in V_0^{'}, i \neq j} u_{ij} - \sum_{i \in V_{N+1}^{'}, i \neq j} u_{ji} = q_j \qquad \forall j \in V^{'} \tag{4.34}$$

$$\sum_{i \in V_{N+1}^{'}, i \neq j} v_{ji} - \sum_{i \in V_0^{'}, i \neq j} v_{ij} = p_j \qquad \forall j \in V^{'} \tag{4.35}$$

$$u_{ij} + v_{ij} \leq C x_{ij} \qquad \forall i \in V_0^{'}, j \in V_{N+1}^{'}, i \neq j \tag{4.36}$$

$$0 \leq y_j \leq y_i - (h d_{ij}) x_{ij} + Q(1 - x_{ij}) \qquad \forall i \in V, \forall j \in V_{N+1}^{'}, i \neq j \tag{4.37}$$

$$0 \leq y_j \leq Y_i - (h d_{ij}) x_{ij} + Q(1 - x_{ij}) \qquad \forall i \in F_0^{'}, \forall j \in V_{N+1}^{'}, i \neq j \tag{4.38}$$

$$y_i \leq Y_i \leq Q \qquad \forall i \in F_0^{'} \tag{4.39}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V_0^{'}, j \in V_{N+1}^{'}, i \neq j \tag{4.40}$$

The objective function (Eq. 4.26) minimizes the total distance traveled. Constraints (4.27) and (4.28) control the connectivity of customers and charging stations, while constraints (4.29) are flow balance constraints. Constraints (4.30) and (4.31) calculate arrival and departure times considering service and battery charging times. Note that constraint (4.31) enforces vehicles to be charged up to the full battery level when they visit a charging station, rather than allowing partial recharging. Constraints (4.32)-(4.36) guarantee that the delivery and

pickup demands of customers are satisfied simultaneously. Finally, constraints (4.37)-(4.39) calculate the battery states.

Since Adapt-Cmsa-SetCov was developed for an extended (more complex) version of the problem, the application, resp. adaptation, to the EVRP-SPD is rather straightforward. Since the reduced problem does not contain TW constraints, the feasibility check of each solution is made only on the basis of the load capacity and battery constraints. Additionally, as also described in the mathematical formulation of the problem, electric vehicles are assumed to become fully charged when they visit a charging station. In fact, note that the battery charging time is only relevant for the TW constraints. Therefore, the removal of the TW constraint turns the battery charging time into an irrelevant aspect.

We apply Adapt-Cmsa-SetCov to the same EVRP-SPD problem instances as those used in [194]. They were derived from the EVRP-TW instances introduced by [166]. In order to modify them to the EVRP-SPD, first, the TW data was removed. Then, the original demands were split into a delivery and a pickup demand using precisely the same procedure as explained in Section 4.7.1. In total, this resulted in 36 small-sized problem instances with 5, 10, and 15 customers, respectively, and in 34 large-sized problem instances with 100 customers.

Adapt-Cmsa-SetCov was applied six times (in order to be comparable to the original publication [194]) to all problem instances. Moreover, we applied CPLEX exactly once to the two-index MILP outlined above. The computation time limit was set to 2 CPU hours for CPLEX and to 2400 CPU seconds for Adapt-Cmsa-SetCov, which is the same time limit as for the VNS variants from [194]. Note also that, for the application of Adapt-Cmsa-SetCov, in order to avoid another costly parameter tuning process, we used the same parameter values as for the application to the EVRP-TW-SPD.

Tables 4.8–4.10 and Table 4.11 present comparative results on small and large-sized EVRP-SPD problem instances, respectively. Hereby, Tables 4.8–4.10 contain results obtained by solving the two different MILP models of the problem for small-sized instances. These results are provided in the columns with heading 'CPLEX1' and 'CPLEX2'. More specifically, results in column 'CPLEX1' (three-index model) were taken from [194], while results in column 'CPLEX2' were obtained by solving the two-index MILP model presented in this study. Moreover, column 'VNS' presents the best results among all VNS variants.

Table 4.8 Comparison for small-sized EVRP-SPD instances with 5 customers.

Instance	CPLEX1			CPLEX2			VNS		Adapt-CMSA-SETCOV		
name	best	time	gap(%)	best	time	gap(%)	best	time	best	avg.	time
c101C5	208.90	1.35	0	208.90	0.30	0	208.90	0.02	208.90	208.90	0.0003
c103C5	154.50	1.20	0	154.50	0.21	0	154.50	0.11	154.50	154.50	0.0002
c206C5	201.55	1.92	0	201.55	0.27	0	201.55	0.02	201.55	201.55	0.0004
c208C5	158.48	1.34	0	158.48	0.15	0	158.48	0.05	158.48	158.48	0.0004
r104C5	136.69	1.75	0	136.69	0.14	0	136.69	0.00	136.69	136.69	0.0008
r105C5	139.48	1.23	0	139.48	0.10	0	139.48	0.00	139.48	139.48	0.0003
r202C5	128.78	1.29	0	128.78	0.10	0	128.78	0.23	128.78	128.78	0.0003
r203C5	179.06	1.37	0	179.06	0.10	0	179.06	0.03	179.06	179.06	0.0003
rc105C5	208.43	1.89	0	208.43	1.70	0	208.43	0.13	208.43	208.43	0.001
rc108C5	211.53	1.36	0	211.53	0.10	0	211.53	0.16	211.53	211.53	0.0006
rc204C5	176.39	2.54	0	176.39	0.33	0	176.39	0.20	176.39	176.39	0.0377
rc208C5	167.98	2.29	0	167.98	0.11	0	167.98	0.02	167.98	167.98	0.0003
average	172.65	1.63		172.65	0.30		172.65	0.08	172.65	172.65	**0.0036**

Table 4.9 Comparison for small-sized EVRP-SPD instances with 10 customers.

Instance	CPLEX1			CPLEX2			VNS		Adapt-CMSA-SETCOV		
name	best	time	gap(%)	best	time	gap(%)	best	time	best	avg.	time
c101C10	260.01	4.85	0	260.01	12.46	0	260.01	3.13	260.01	260.01	0.0057
c104C10	239.13	3.39	0	239.13	0.56	0	239.13	1.53	239.13	239.13	0.0032
c202C10	214.96	4.12	0	214.96	2.12	0	214.96	3.05	214.96	214.96	0.0038
c205C10	224.78	4.45	0	224.78	2.10	0	224.78	0.36	224.78	224.78	0.9274
r102C10	220.97	19.01	0	220.97	5.10	0	220.97	0.72	220.97	220.97	14.4919
r103C10	160.41	10.35	0	160.41	5.24	0	160.41	2.55	160.41	160.41	0.0012
r201C10	183.11	2.36	0	183.11	5.20	0	183.11	2.30	183.11	183.11	0.0049
r203C10	214.90	5.43	0	214.90	1.62	0	214.90	1.86	214.90	214.90	0.0044
rc102C10	346.70	4.03	0	346.70	2.11	0	346.70	1.65	346.70	346.70	0.0063
rc108C10	317.96	6.00	0	317.96	7.11	0	317.96	3.30	317.96	317.96	0.0029
rc201C10	246.99	5.26	0	246.99	10.61	0	246.99	9.78	246.99	246.99	0.0029
rc205C10	306.82	4.14	0	306.82	1.47	0	306.82	0.79	306.82	306.82	0.0039
average	244.73	6.12		244.73	4.64		244.73	2.59	244.73	244.73	**1.29**

Table 4.11, on the other hand, compares the performance of Adapt-CMSA-SETCOV with Reduced VNS and General VNS, which are reported to provide the best performance for large-sized problem instances. Note also that the VNS results given in these tables are taken from the respective publication.

The tables displaying the results are organized as follows. In the first column, the names of the problem instances are listed. The 'best' columns indicate the objective function values of the best solutions obtained from six independent runs. Additionally, columns with the heading 'avg.' show the average objective function values of the best solutions found from the six runs. It is important to note that these objective function values represent the total distance traveled by the vehicles used in the respective solutions. The remaining columns with the heading 'time' display the computation time (measured in seconds) of CPLEX, as well as the average computation times of VNS and Adapt-CMSA-SETCOV to obtain the best solutions in each run.

One can make the following observations. CPLEX managed to optimally solve every small-sized problem instance (with 5, 10 and 15 customers), with

Table 4.10 Comparison for small-sized EVRP-SPD instances with 15 customers.

Instance	CPLEX1			CPLEX2			VNS		Adapt-CMSA-SetCov		
name	best	time	gap(%)	best	time	gap(%)	best	time	best	avg.	time
c103C15	255.68	30.81	0	255.68	14.88	0	255.68	20.29	255.68	255.68	67.0909
c106C15	223.84	142.65	0	223.84	15.62	0	223.84	6.36	223.84	223.84	0.0214
c202C15	314.62	373.20	0	314.62	35.95	0	314.62	40.32	314.62	314.62	0.0345
c208C15	262.50	244.40	0	262.50	22.41	0	262.50	2.84	262.50	262.50	70.4837
r102C15	258.59	681.12	0	258.59	110.70	0	258.59	23.27	258.59	258.59	0.0846
r105C15	231.96	119.88	0	231.96	15.31	0	231.96	10.78	231.96	231.96	8.3377
r202C15	275.04	64.31	0	275.04	177.24	0	275.04	13.64	275.04	275.04	0.021
r209C15	239.70	49.60	0	239.70	6.15	0	239.70	9.22	239.70	239.70	3.8396
rc103C15	291.07	52.73	0	291.07	11.31	0	291.07	9.73	291.07	291.07	0.0462
rc108C15	330.01	1197.23	0	330.01	50.98	0	330.01	5.76	330.01	330.01	0.0137
rc202C15	295.60	87.00	0	295.60	22.05	0	295.60	50.44	295.60	295.60	0.0302
rc204C15	285.13	29.20	0	285.13	57.57	0	285.13	14.64	285.13	285.13	0.3149
average	271.98	256.01		271.98	45.01		271.98	17.27	271.98	271.98	**12.53**

both the MILP model from [194] and the one presented in this chapter. However, CPLEX2 showed this performance with considerably less computation time than CPLEX1. More specifically, CPLEX1 and CPLEX2 found their best solutions on average in 87.94 and 16.65 seconds, respectively. It can be deduced that our formulation is more efficient than the one employed in [194]. However, a definite conclusion can only be reached by solving both models under the same computational conditions. Concerning the heuristic approaches, VNS and Adapt-CMSA-SetCov were able to find all the optimal solutions provided by CPLEX. However, Adapt-CMSA-SetCov presented this performance requiring less computation time than VNS. More specifically, VNS could derive its best solutions on average in 6.65 seconds, while Adapt-CMSA-SetCov could do so in just 4.61 seconds.

Regarding the large-sized EVRP-SPD instances, Adapt-CMSA-SetCov significantly outperforms the best VNS variants in terms of best and average results. Note that these variants are reported as the top two variants among all VNS variants tested in the respective publication. In addition to the results presented in Table 4.11, the superiority of Adapt-CMSA-SetCov over Reduced-VNS and General-VNS is also seen in the CD plot of Figure 4.9. Note also that Adapt-CMSA-SetCov showed this performance using considerably less computation time than both VNS variants. It is remarkable that this performance is obtained without having Adapt-CMSA-SetCov specifically tuned for the EVRP-SPD.

Table 4.11 Comparison for large-sized EVRP-SPD instances with 100 customers.

Instance	Reduced VNS - Cyclic			General VNS - Cyclic			Adapt-CMSA-SETCOV		
name	best	avg.	time	best	avg.	time	best	avg.	time
c101	734.89	775.05	2185	718.96	753.76	2402.7	**693.50**	**695.25**	1079.17
c201	567.14	606.36	2067	**563.09**	598.29	2244.5	576.52	**578.86**	1142.17
c204	585.63	613.19	2405	579.59	609.54	2356.9	**576.52**	**577.45**	1613.66
c206	**569.2**	606.71	2288	597.46	619.09	2155.3	576.52	**579.95**	1259.32
c207	**567.76**	606.47	2513	599.72	620.50	1793.2	576.79	**584.72**	1588.23
r101	823.51	884.72	2305	843.74	866.86	2395.3	**762.73**	**770.90**	831.77
r104	880.77	923.77	1222	888.24	918.66	2334.8	**758.26**	**773.21**	679.35
r106	864.58	899.18	1812	886.88	915.67	2290.6	**766.06**	**775.98**	773.49
r107	856.46	907.39	2347	864.49	927.58	2363.4	**755.29**	**771.88**	1059.22
r108	842.01	887.50	2291	857.77	885.94	2188.1	**760.38**	**771.47**	860.67
r109	865.47	895.36	2272	854.32	901.75	1848.1	**752.02**	**764.97**	1350.00
r110	880.77	922.45	1235	857.25	911.79	1834.6	**759.76**	**771.24**	864.63
r111	879.5	904.20	1596	862	905.00	2333.7	**761.32**	**775.02**	724.47
r112	876.19	903.22	1637	854.32	892.78	1924.6	**761.28**	**776.42**	729.53
r201	690.73	719.61	2133	713.9	736.72	1687.4	**650.73**	**653.18**	1107.97
r202	690.38	707.28	2247	711.74	740.64	1791.3	**643.27**	**650.63**	1055.45
r203	708.41	719.08	1964	713.9	740.60	2239.7	**646.45**	**652.14**	891.73
r204	698.59	707.81	1477	708.64	730.32	2282.2	**643.85**	**649.09**	990.88
r205	690.35	711.13	2425	704.02	730.47	2378.9	**645.79**	**651.43**	818.71
r206	694.54	717.43	2099	692.95	723.98	2205.3	**650.64**	**653.66**	813.30
r207	701.66	710.63	1020	701.42	722.25	2265.3	**643.26**	**648.80**	1159.51
r208	687.23	706.42	2283	714.12	742.15	2210.4	**644.39**	**649.03**	696.90
r209	696.17	718.88	2141	692.95	725.09	2178.9	**647.80**	**654.40**	1121.36
r210	708.41	720.19	1991	713.9	739.96	2166.8	**645.93**	**653.84**	1167.71
r211	701.66	710.63	1031	700.53	720.87	2208	**644.02**	**649.45**	1160.24
rc101	887.66	935.90	2404	885.96	920.75	1812.1	**832.30**	**843.94**	724.21
rc201	682.65	726.47	1767	698.37	722.56	1658.4	**651.88**	**655.83**	931.56
rc202	703.03	712.23	2187	710.3	726.25	2100.2	**649.68**	**653.99**	1045.01
rc203	685.09	728.94	2305	698.37	730.03	2358.5	**651.68**	**656.76**	991.55
rc204	723.14	734.23	2341	731.65	762.60	2168.8	**649.42**	**655.57**	977.83
rc205	715.51	739.29	1828	721.6	753.78	2437.4	**649.68**	**656.75**	1249.43
rc206	691.17	718.47	2402	735.66	767.70	1682.5	**646.39**	**654.16**	587.07
rc207	685.09	727.85	2309	707.51	732.01	2024.8	**652.64**	**656.07**	1384.80
rc208	721.41	752.14	2100	743.13	774.94	2438.8	**652.08**	**656.02**	957.69
average	734.02	763.53	2018.5	742.01	772.67	2140.04	**675.19**	**682.53**	**1015.05**

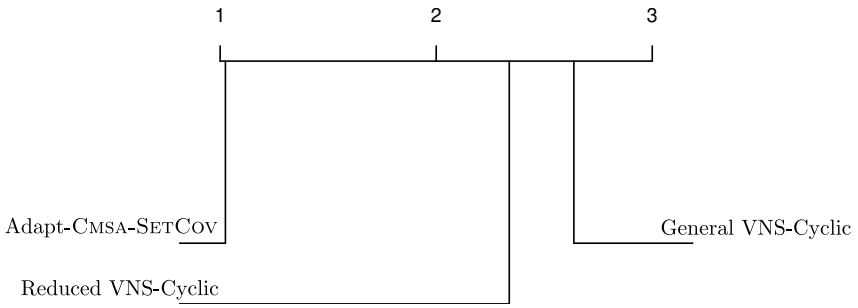


Fig. 4.9 Critical difference (CD) plot concerning the results for the large EVRP-SPD instances with 100 customers.

4.10 Conclusions and Future Research Directions

In this chapter, we presented the application of different Adapt-CMSA variants to the electric vehicle routing problem with time windows and simultaneous pickup and delivery. This problem has the property that it can be modeled by means

of an assignment-type integer linear program and also by a set-covering-based integer linear program. Both models were used for solving sub-instances at each iteration of the presented CMSA algorithms. The results have shown that the CMSA variant using the set-covering-based model generally significantly outperforms the CMSA variant using the standard assignment-type models. In fact, CMSA algorithm seems an ideal algorithmic framework in order to take profit of set-covering-based models for solving optimization problems that can be modeled in this way. This is because CMSA algorithms are less sophisticated and easier to implement in comparison to column generation approaches [68]. In addition, CMSA algorithms have the ability to explore search spaces, in contrast to simpler heuristic methods from the literature that were devised to take profit from set-covering-based models.

In order to further evaluate the performance of the proposed approach, Adapt-Cmsa-SetCov was also compared with several variable neighborhood search algorithm variants from the literature on the electric vehicle routing problem with simultaneous pickup and deliveries. Numerical results showed that Adapt-Cmsa-SetCov outperformed those variable neighborhood search variants from the literature on all problem instances. In the future, we aim to develop more efficient solution construction methods and MILP models to be employed within Adapt-CMSA. Moreover, the analysis of the algorithm will be deepened on a wider set of benchmark instances.

CHAPTER 5

APPLICATION TO THE TWO-ECHELON ELECTRIC VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERIES

5.1 Introduction

This chapter describes the application of Adapt-CMSA to the *Two-Echelon Electric Vehicle Routing Problem with Simltaneus Pickup and Delivery* (2E-EVRP-SPD). The content shown in this chapter was also presented in [6] published in the proceedings of *EVOCOP 2023:European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar)* (`https://doi.org/10.1007/978-3-031-30035-6_22`).

First, the addressed problem is formulated as a mixed integer linear program (MILP). Any general-purpose MILP solver, such as CPLEX[1] or Gurobi[2], may be used to solve this model. However, due to the multi-tier structure of the distribution network, the limited driving range of electric vehicles, and the SPD constraints, the 2E-EVRP-SPD problem is rather complex. In fact, our computational experiments show that CPLEX struggled to solve even small-sized problems to optimality. In most cases, CPLEX was only able to derive valid solutions with large optimality gaps. Therefore, we developed an Adapt-CMSA, which was already generally introduced in Chapter 2 and applied to the EVRP-TW-SPD in Chapter 4, for being able to solve large-sized problem instances.

In the context of problem instances too large for the application of CPLEX, our algorithm is compared to probabilistic versions of two well-known constructive heuristics. The numerical results show that our algorithm outperforms CPLEX in the context of rather small problem instances. Moreover, it is shown to outperform the heuristic algorithms when larger problem instances are concerned.

[1] `https://www.ibm.com/analytics/cplex-optimizer`
[2] `http://www.gurobi.com/`

Table 5.1 Sets and notations

n_d, n_s , n_r , n_c	: Number of central warehouses, satellites, charging stations and customers, respectively
N_D	: Set of central warehouses, $N_D = \{n_{d_1}, ..., n_{d_{n_d}}\}$
$N_D^{'}$	: Set of dummy central warehouses corresponding to N_D, $N_D^{'} = \{n_{d_1}^{'}, ..., n_{d_{n_d}}^{'}\}$
N_S	: Set of satellites, $N_S = \{n_{s_1}, ..., n_{s_{n_s}}\}$
$N_S^{'}$	: Set of dummy satellites corresponding to N_S, $N_S^{'} = \{n_{s_1}^{'}, ..., s_{s_{n_s}}^{'}\}$
N_R	: Set of charging stations, $N_R = \{n_{r_1}, ..., n_{r_{n_r}}\}$
N_C	: Set of customers, $N_C = \{n_{c_1}, ..., n_{c_{n_c}}\}$
N_{DS}	: Set of central warehouses and satellites, $N_{DS} = N_D \cup N_S$
N_{SD}	: Set of satellites and dummy central warehouses, $N_{SD} = N_S \cup N_D^{'}$
N_{DSD}	: Complete set of nodes in the first echelon, $N_{DSD} = N_D \cup N_S \cup N_D^{'}$
N_{RC}	: Set of charging stations and customers, $N_{RC} = N_R \cup N_C$
N_{SRC}	: Set of satellites, charging stations and customers, $N_{SRC} = N_S \cup N_R \cup N_C$
N_{RCS}	: Set charging stations, customers and dummy satellites, $N_{RCS} = N_R \cup N_C \cup N_S^{'}$
N_{SRCS}	: Complete set of nodes in the second echelon, $N_{SRCS} = N_S \cup N_R \cup N_C \cup N_S^{'}$

5.1.1 Problem Description

In the following, we provide a technical description together with a MILP model of the 2E-EVRP-SPD. For this purpose, the sets and notations from Table 5.1 are required. The 2E-EVRP-SPD can be defined on a complete, directed graph $G(N, A)$ that is formed by the following subsets of nodes: the set of central warehouses (also called depots) (N_D), the set of satellites (N_S), the set of charging stations (N_R), and the set of customers (N_C). Note that N_S and N_R also include multiple copies of each satellite and charging station to allow multiple visits to any of the satellites and charging stations. The set of arcs on the other hand (A) includes (1) arcs that connect central warehouses and satellites $A^1 = \{(i, j) \mid i \neq j \text{ and } i, j \in N_{DSD}\}$ and (2) arcs that connect satellites, customers and charging stations $A^2 = \{(l, m) \mid l \neq m \text{ and } l, m \in N_{SRCS}\}$. Each arc $(i, j) \in A^1$ is associated with a distance d_{ij}^1 and each arc $(l, m) \in A^2$ is associated with a distance d_{lm}^2.

Two different fleets of vehicles, each one homogeneous in itself, serve in the first and second echelons in order to meet customer demands. A fleet of large trucks with internal combustion engines are located in a central warehouse and transfer products between the central warehouses and the satellites, while a fleet of electric vehicles is present at the satellites and transfer products between satellites and customers (demand points). In the first echelon, a truck with a loading capacity of Q^1 starts its tour from a central warehouse, visits one or more satellites, and returns to the central warehouse from which the tour started. Not all satellites have to be visited by large trucks unless there is a demand for pickup and/or delivery. Furthermore, a satellite can be visited by multiple large vehicles if the delivery or pickup demand of the satellite exceeds the vehicle capacity. In the second echelon, on the other hand, each customer with a delivery demand

$D_i^2 > 0$ or a pickup demand $P_i^2 > 0$ (or both) must be served by an electric vehicle with a loading capacity of Q^2. An electric vehicle starts its tour with a fully charged battery (B) and the vehicle's battery is consumed in proportion to the distance travelled. The constant h represents the battery consumption rate of an electric vehicle per unit distance travelled. If a charging station is visited, the electric vehicle's battery is fully charged up to level B with a constant charging rate of $g > 0$.

Our MILP model contains the following binary decision variables. A decision variable x_{ij} takes value 1 if arc $(i, j) \in A^1$ is traversed, and 0 otherwise. Moreover, a decision variable y_{ij} takes value 1 if arc $(i, j) \in A^2$ is traversed, and 0 otherwise. Next, decision variables $BSCa_i$ and $BSCd_i$ record the battery state of charge on arrival, respectively departure, at (from) vertex $i \in N_{SRCS}$. Furthermore, for each arc $(i, j) \in A^1$, variable u_{ij}^1 denotes the remaining cargo to be delivered to satellites of the route, while v_{ij}^1 denotes the amount of cargo already collected (picked up) at already visited satellites. Similarly, for each arc $(i, j) \in A^2$, variable u_{ij}^2 denotes the remaining cargo for the route, while v_{ij}^2 denotes the amount of cargo already collected at visited customers. Since the demand of each satellite depends on the customers serviced through it, decision variables D_i^1 and P_i^1 are introduced to calculate, respectively, the delivery and pickup demands of satellites. Finally, variable z_{ij} takes value 1 if customer (i) is serviced from satellite (j), and 0 otherwise. The MILP model can then be stated as follows.

$$\textbf{Min} \quad \sum_{i \in N_{DS}} \sum_{j \in N_{DSD}} d_{ij}^1 \cdot x_{ij} + \sum_{l \in N_{SRC}} \sum_{m \in N_{SRCS}} d_{lm}^2 \cdot y_{lm} + \sum_{j \in N_{DSD}} x_{0j} \cdot c^{lv} + \sum_{i \in N_S} \sum_{j \in N_{SRCS}} y_{ij} \cdot c^{ev} \tag{5.1}$$

$$\sum_{j \in N_{SD}} x_{ij} \leq 1 \qquad \forall i \in N_S \tag{5.2}$$

$$\sum_{i \in N_{DS}, i \neq j} x_{ij} - \sum_{i \in N_{SD}, i \neq j} x_{ji} = 0 \qquad \forall j \in N_S \tag{5.3}$$

$$\sum_{i \in N_{DS}, i \neq j} u_{ij}^1 - \sum_{i \in N_{SD}, i \neq j} u_{ji}^1 = D_j^1 \qquad \forall j \in N_S \tag{5.4}$$

$$\sum_{i \in N_{DS}, i \neq j} v_{ij}^1 - \sum_{i \in N_{SD}, i \neq j} v_{ji}^1 = P_j^1 \qquad \forall j \in N_S \tag{5.5}$$

$$0 \leq u_{ij}^1 \leq Q^1 \qquad \forall i \in N_D, j \in N_{DS} \tag{5.6}$$

$$v_{ij}^1 = 0 \qquad \forall i \in N_D, j \in N_{SD} \tag{5.7}$$

$$u_{ij}^1 + v_{ij}^1 \leq Q^1 * x_{ij} \qquad \forall i \in N_D, j \in N_{DS}, i \neq j \tag{5.8}$$

$$\sum_{l \in N_C} z_{li} * D_l^2 = D_i^1 \qquad \forall i \in N_S \quad (5.9)$$

$$\sum_{l \in N_C} z_{li} * P_l^2 = P_i^1 \qquad \forall i \in N_S \quad (5.10)$$

$$\sum_{j \in N_{RCS}, i \neq j} y_{ij} = 1 \qquad \forall i \in N_C \quad (5.11)$$

$$\sum_{j \in N_{RCS}, i \neq j} y_{ij} \leq 1 \qquad \forall i \in N_R \quad (5.12)$$

$$\sum_{i \in N_{SRC}, i \neq j} y_{ij} - \sum_{i \in N_{RCS}, i \neq j} y_{ji} = 0 \qquad \forall j \in N_{RC} \quad (5.13)$$

$$\sum_{i \in N_S} z_{li} = 1 \qquad \forall l \in N_{RC} \quad (5.14)$$

$$y_{li} \leq z_{li} \qquad \forall i \in N_S, l \in N_{RC} \quad (5.15)$$

$$y_{il} \leq z_{li} \qquad \forall i \in N_S, l \in N_{RC} \quad (5.16)$$

$$y_{lm} + z_{li} + \sum_{s \in N_S, i \neq s} z_{ms} \leq 2 \qquad \forall l, m \in N_{RC}, l \neq m, \forall i \in N_S \quad (5.17)$$

$$\sum_{i \in N_{SRC}, i \neq j} u_{ij}^2 - \sum_{i \in N_{RCS}, i \neq j} u_{ji}^2 = D_j^2 \qquad \forall j \in N_{RC} \quad (5.18)$$

$$\sum_{i \in N_{SRC}, i \neq j} v_{ji}^2 - \sum_{i \in N_{RCS}, i \neq j} v_{ij}^2 = D_j^2 \qquad \forall j \in N_{RC} \quad (5.19)$$

$$0 \leq u_{ij}^2 \leq Q^2 \qquad \forall i \in N_S, j \in N_{RCS} \quad (5.20)$$

$$v_{ij}^2 = 0 \qquad \forall i \in N_S, j \in N_{RCS} \quad (5.21)$$

$$u_{ij}^2 + v_{ij}^2 \leq Q^2 * y_{ij} \qquad \forall i \in N_{RCS}, j \in N_{SRC} \quad (5.22)$$

$$0 \leq BSCa_j \leq BSCa_i - (hd_{ij})y_{ij} + B(1 - y_{ij}) \qquad \forall i \in N_C, \forall j \in N_{RCS}, i \neq j \quad (5.23)$$

$$0 \leq BSCa_j \leq BSCd_i - (hd_{ij})y_{ij} + B(1 - y_{ij}) \qquad \forall i \in N_{SR}, \forall j \in N_{RCS}, i \neq j \quad (5.24)$$

$$BSCa_i \leq BSCd_i \leq B \qquad \forall i \in N_{SR} \quad (5.25)$$

$$x_{ij} \in 0, 1 \qquad \forall i \in N_{SRC}, j \in N_{RCS}, l \neq m \quad (5.26)$$

$$y_{lm} \in 0, 1 \qquad \forall l \in N_{SRC}, m \in N_{RCS}, l \neq m \quad (5.27)$$

In this problem, solutions using fewer vehicles—that is, with fewer routes—are preferred over others, even if the total distance travelled is higher than in other routes. Therefore, the objective function does not only consider the travelled distance but adds also an extra cost c^{lv} for each large vehicle used in the first echelon and c^{ev} for each electric vehicle used in the second echelon. Note, in this context, that the number of large vehicles used in a solution is equal to the number of variables on outgoing arcs of a central warehouse with a value of 1. Moreover, the number of electric vehicles used in a solution is equal to the number of variables on outgoing arcs of satellites that have a value of 1. In this way, the objective function (5.1) minimizes

the sum of the total distance travelled and the vehicle costs. Constraints (5.2) control the connectivity of satellites and constraints (5.3) guarantee the balance of flow in the first echelon nodes. Constraints (5.4)-(5.8) guarantee that the delivery and pickup demands of satellites are satisfied simultaneously by the large vehicles serving in the first echelon. Constraints (5.9) and (5.10) determine each satellite's delivery and pickup demand to be the total delivery and pickup demands of those customers served by the relevant satellite. Constraints (5.11) and (5.12) control the connectivity of customers and charging stations. Constraints (5.13) ensure the balance of flow for the second echelon nodes. Constraints (5.14) guarantee that a customer receives service from only one satellite. Constraints (5.15)-(5.17) ensure that a tour started from a satellite ends at the same satellite. Constraints (5.18)-(5.22) guarantee that the delivery and pickup demands of customers are satisfied simultaneously by the electric vehicles serving in the second echelon. Finally, constraints (5.23)-5.25) are battery state constraints.

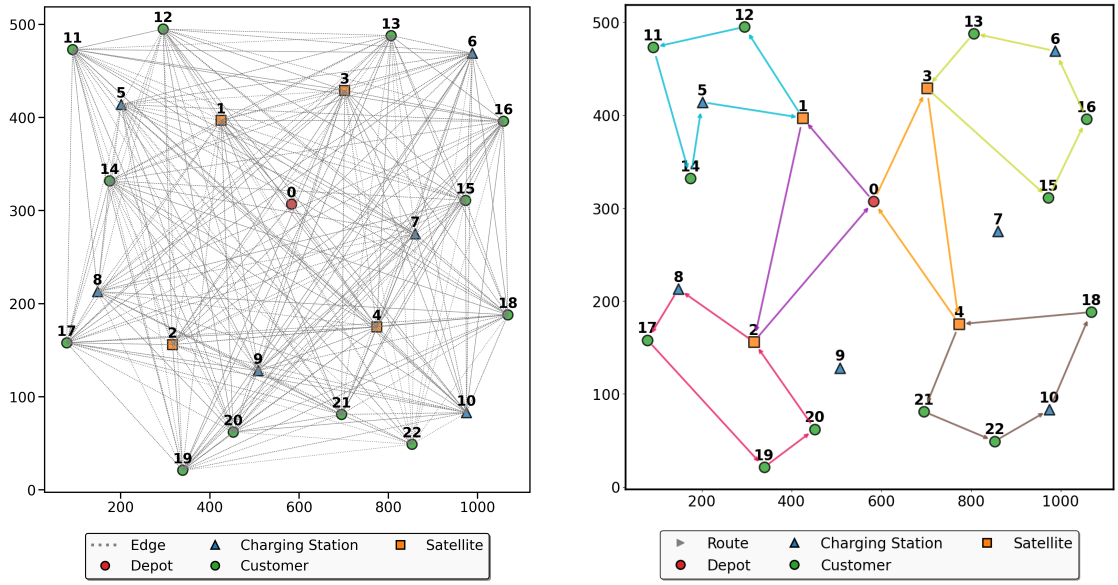
5.2 Adapt-CMSA for the 2E-EVRP-SPD

In this section, we will describe the Adapt-CMSA algorithm that we designed for the application to the 2E-EVRP-SPD. However, before describing the algorithm we first explain the solution representation.

5.2.1 Solution Representation

Any solution $S = (R^1, R^2)$ produced by the algorithm consists of two sets of routes, R^1 and R^2, where R^1 contains the routes of large vehicles in the first echelon and R^2 contains the routes of the electric vehicles in the second echelon. Each route $t^1 \in R^1$ starts from a central warehouse, visits one or more satellites and returns to the same central warehouse. Each route $t^2 \in R^2$ starts from a satellite, visits a number of locations $v \in N_{RC}$ and returns to the same satellite. As an example, let vector **I** contain the complete set of node indexes of an example problem instance with one central warehouse, two satellites, three charging stations and five customers.

$$\mathbf{I} = (\underbrace{0,}_{\text{central warehouse}} \underbrace{1,\ 2,}_{\text{satellites}} \underbrace{3,\ 4,\ 5,}_{\text{charging stations}} \underbrace{6,\ 7,\ 8,\ 9,\ 10,}_{\text{customers}})$$



(a) EVRP Instance **(b)** Solution

Fig. 5.1 Illustration of a 2E-EVRP instance and its solution. (a) presents a map showing the locations of a depot, four satellites, twelve customers, and six charging stations based on Cartesian coordinates. Gray dashed lines indicate a fully connected graph within each of the two echelons, with no direct connections between the depot and the customers or charging stations. (b) shows a valid solution to the given instance on the same map, with two distinct tours in the first echelon and 4 different tours in the second echelon represented by arrows with different colors. First echelon routes begin and end at the central warehouse passing through satellites while the second echelon routes begin and end at the satellites, passing through various customers and charging stations.

A solution S that contains one route in the first echelon (t_1^1) and two routes in the second echelon (t_1^2 and t_2^2) is represented as follows:

$$S = (R^1, R^2) \text{ where } R^1 = \left\{ t_1^1 = \{0 \to 1 \to 2 \to 0\} \right\}$$
$$R^2 = \left\{ \begin{array}{c} t_1^2 = \{1 \to 7 \to 3 \to 6 \to 1\} \\ t_2^2 = \{2 \to 9 \to 8 \to 5 \to 10 \to 2\} \end{array} \right\}$$

An extended version of the example instance and a valid solution is visually illustrated in Figure 5.1.

5.2.2 Set Covering Based Model

As described in Section 5.2.1, any solution produced by the algorithm is kept in the form of two sets of routes. Similarly, a sub-instance $C = (C^1, C^2)$ in the

context of our Adapt-CMSA algorithm consist of two sets, C^1 and C^2, containing those routes that were previously generated by the probabilistic application of solution construction heuristics. For solving such a sub-instance, in this study, we make use of the following set-covering-based MILP model. Note that the other option would have been to use the MILP model outlined in the previous section, with those arc variables fixed to zero whose corresponding arcs do not form part of any of the solutions that were merged into the sub-instance.

Given a sub-instance $C = (C^1, C^2)$, each route $r^k \in C^k$ is associated with a distance, resp. cost, value d_r^k. Moreover, ld_{sr}^2 and lp_{sr}^2 represent the total delivery and pickup loads of the route $r \in C^2$ served from satellite $s \in N_S$. As described in Section 5.1.1, D_s^1 and P_s^1 refer to delivery and pickup demands of the respective satellite. Parameter a_{sr}^1 is set to value one if satellite s is traversed by route r, and 0 otherwise. Moreover, parameter a_{ir}^2 is set to value one if customer i is visited by route r, and 0 otherwise. The binary decision variable x_r^k takes value one if the route in the k-th echelon is selected, value zero otherwise. Moreover, dp_{sr} and pp_{sr} refer to the amount of goods delivered to the satellite s by the route r. The set-covering-based model can then be stated as follows.

$$\textbf{Min} \quad \sum_{r \in C^1} d_r^1 * x_r^1 + \sum_{r \in C^2} d_r^2 * x_r^2 + \sum_{r \in C^1} c^{lv} * x_r^1 + \sum_{r \in C^2} c^{ev} * x_r^2 \tag{5.28}$$

$$\textbf{s.t.} \quad \sum_{r \in C^2} ld_{sr}^2 = D_s^1 \qquad \forall s \in N_s \tag{5.29}$$

$$\sum_{r \in C^2} lp_{sr}^2 = P_s^1 \qquad \forall s \in N_s \tag{5.30}$$

$$\sum_{r \in C^1} pd_{sr} * a_{sr}^1 = D_s^1 \qquad \forall s \in N_s \tag{5.31}$$

$$\sum_{r \in C^1} pp_{sr} * a_{sr}^1 = P_s^1 \qquad \forall s \in N_s \tag{5.32}$$

$$\sum_{s \in N_S} pd_{sr} \leq Q^1 * x_r^1 \qquad \forall r \in C^1 \tag{5.33}$$

$$\sum_{r \in C^1} pp_{sr} \leq Q^1 * x_r^1 \qquad \forall r \in C^1 \tag{5.34}$$

$$\sum_{r \in C^2} a_{ir}^2 * x_r^2 >= 1 \qquad \forall i \in V \tag{5.35}$$

$$x_{r_1}^1, x_{r_2}^2 \in 0, 1 \qquad \forall r_1 \in C^1, r_2 \in C^2, k \in \{1, 2\} \tag{5.36}$$

$$pd_{sr}, pp_{sr} \geq 0 \qquad \forall s \in N_S, r \in C^1 \tag{5.37}$$

The objective function minimizes the sum of the total distance travelled and vehicle costs. Constraints (5.29) and (5.30) determine each satellite's delivery and pickup demands. Constraints (5.31) and (5.32) ensure that partial deliveries are

allowed in the first echelon in case the delivery or pickup demand of the relevant satellite exceeds the capacity of a large vehicle. Constraints (5.33) and (5.34) guarantee that the total delivery and pickup load in large vehicles can not exceed the vehicle capacity. Constraint (5.35) ensures that each customer must be visited at least once. Finally, constraints (5.36) and (5.37) control variable domains.

5.2.3 The Adapt-CMSA Algorithm

Algorithm 5.1 shows the pseudo-code of our Adapt-CMSA algorithm for the 2E-EVRP-SPD. First, the best-so-far solution S_{bsf} is initialized as an empty solution. Then, parameters α_{bsf}, n_a and l_{size} are initialized in lines 4 and 5. The handling of these parameters was already described in previous chapters. However in order for this chapter be self-contained, they will be described in detail again below.

At each iteration, Adapt-CMSA builds a sub-instance C of the original problem instance as follows. First, C is initialized to the best-so-far solution S_{bsf}. Then, a number of n_{a} solutions are probabilistically constructed in lines 8–12. The function for the construction of a solution, ProbabilisticSolutionConstruction(S_{bsf}, α_{bsf}, l_{size}), receives—apart from the best-so-far-solution S_{bsf}—two parameters as input. Here, parameter α_{bsf} (where $0 \leq \alpha_{\text{bsf}} < 1$) is used to bias the construction of new solutions towards the best-so-far solution S_{bsf}. More specifically, the similarity between the constructed solutions and S_{bsf} will increase with a growing value of α_{bsf}. Parameter l_{size} controls the number of considered options at each solution construction step. A higher value of l_{size} results in more diverse solutions which, in turn, leads to a larger sub-instance. After the construction of a solution S (line 9), a local search is applied to each route $t^2 \in R^2$, see line 10. Well-known intra-route operators such as, *relocation*, *swap* and *two_opt* are sequentially utilized to improve each route. Each operator uses the best-improvement strategy. The *relocation* operator removes each customer from its current position and inserts it into a different position in the same route. The *swap* neighbourhood considers changing the positions of two selected nodes of the same route. Finally, the *two_opt* neighbourhood considers all possibilities of selecting two non-consecutive nodes in the same route and reversing the node sequence between the two selected nodes.

After the application of local search, the so-called *merge step* is performed in function Merge(C, S). In particular, every route from S^1 is added to C^1 and every route from S^2 is added to C^2. After probabilistically constructing n_{a} solutions and merging them to form the sub-instance C, the sub-instance is solved with

Algorithm 5.1 Adapt-CMSA for the 2E-EVRP-SPD

1: **input 1:** values for CMSA parameters t_{prop}, t_{ILP}
2: **input 2:** values for solution construction parameters α^{LB}, α^{UB}, α_{red}
3: $S_{\text{bsf}} := \emptyset$
4: $\alpha_{\text{bsf}} := \alpha^{\text{UB}}$
5: Initialize$(n_{\text{a}}, l_{\text{size}})$
6: **while** CPU time limit not reached **do**
7: $C := S_{\text{bsf}}$
8: **for** $i := 1, \ldots, n_{\text{a}}$ **do**
9: $S :=$ ProbabilisticSolutionConstruction(S_{bsf}, α_{bsf}, l_{size})
10: LocalSearch1(S)
11: $C :=$ Merge(C, S)
12: **end for**
13: $(S'_{\text{opt}}, t_{\text{solve}}) :=$ SolveSubinstance(C, t_{ILP}) {This function returns two objects: (1) the obtained solution (S'_{opt}), (2) the required computation time (t_{solve})}
14: RemoveDuplicates(S'_{opt})
15: LocalSearch2(S'_{opt})
16: **if** $t_{\text{solve}} < t_{\text{prop}} \cdot t_{\text{ILP}}$ and $\alpha_{\text{bsf}} > \alpha^{\text{LB}}$ **then** $\alpha_{\text{bsf}} := \alpha_{\text{bsf}} - \alpha_{\text{red}}$ **end if**
17: **if** $f(S'_{\text{opt}}) < f(S_{\text{bsf}})$ **then**
18: $S_{\text{bsf}} := S'_{\text{opt}}$
19: Initialize$(n_{\text{a}}, l_{\text{size}})$
20: **else**
21: **if** $f(S'_{\text{opt}}) > f(S_{\text{bsf}})$ **then**
22: **if** $n_a = n^{\text{init}}$ **then** $\alpha_{\text{bsf}} := \min\{\alpha_{\text{bsf}} + \frac{\alpha_{\text{red}}}{10}, \alpha^{\text{UB}}\}$ **else** Initialize$(n_{\text{a}}, l_{\text{size}})$ **end if**
23: **else**
24: Increment$(n_{\text{a}}, l_{\text{size}})$
25: **end if**
26: **end if**
27: **end while**
28: **output:** S_{bsf}

CPLEX, which is precisely done in function SolveSubinstance(C, t_{ILP}); see line 13. Hereby, t_{ILP} is the CPU time limit for the application of CPLEX, which is applied to the set-covering model from Section 5.2.2. Note that the output S'_{opt} of function SolveSubinstance(C, t_{ILP}) is—due to the computation time limit—not necessarily an optimal solution to the sub-instance. Since the set-covering-based model potentially allows customers to be visited more than once, S'_{opt} may contain some of the customers in multiple routes. In that case, redundant customers are removed using function RemoveDuplicates(S'_{opt}), see line 14. This function first determines all redundant customers and calculates the distance between the respective customer and the two adjacent nodes. Then, it removes all redundant customers starting from the one with the highest distance value until all customers

only appear in exactly one route. Subsequently, a local search procedure is applied to S'_{opt} using inter-route neighbourhood operators *exchange (1,1)* and *shift (1,0)*. The *exchange (1,1)* neighbourhood considers all exchanges of two customers not in the same route. The *shift (1,0)* neighbourhood looks at all possibilities of removing a customer from its current route and inserting it at any position in the other routes. Both operators are applied based on the best-improvement strategy.

The self-adaptive aspect of Adapt-CMSA is to be found in the dynamic change of parameters α_{bsf}, n_{a} and l_{size}. In the first place, we will describe the adaptation of parameter α_{bsf}. First of all, Adapt-CMSA requires a lower bound α^{LB} and an upper bound α^{UB} for the value of α_{bsf} as input. In addition, the step size α_{red} for the reduction of α_{bsf} must also be given as input. Adapt-CMSA starts by setting α_{bsf} to the highest possible value α^{UB}; see line 4.[3] In case the resulting MILP can be solved in a computation time t_{solve} which is below a proportion t_{prop} of the maximally possible computation time t_{ILP}, the value of α_{bsf} is reduced by α_{red}; see line 22. The rationale behind this step is as follows. In case the resulting MILP can easily be solved to optimality, the search space is too small, caused by a rather low number of routes in C^1 and C^2. In order to increase the size of the MILP, the solutions constructed in ProbabilisticSolutionConstruction(S_{bsf}, α_{bsf}, l_{size}, d_{rate}, h_{rate}) should be more different to S_{bsf}, which can be achieved by reducing the value of α_{bsf}.

The adaptation of parameters n_{a} and l_{size} is done in a similar way and with a similar purpose. These parameters are set to their initial values, that is, $n_{\text{a}} := n^{\text{init}}$ and $l_{\text{size}} := l_{\text{size}}^{\text{init}}$ in function Initialize($n_{\text{a}}, l_{\text{size}}$), which is called at three different occasions: (1) at the start of the algorithm (line 5), (2) whenever solution S'_{opt} is strictly better than S_{bsf} (line 19), and (3) whenever solution S'_{opt} is strictly worse than S_{bsf} and, at the same time, $n_{\text{a}} > n^{\text{init}}$ (line 22). On the other side, in those cases in which S'_{opt} and S_{bsf} are of the same quality, the algorithm can afford to generate larger sub-instances and therefore, the values of the two parameters are incremented in function Increment($n_{\text{a}}, l_{\text{size}}$). More specifically, n_{a} is incremented by one and l_{size} is incremented by $l_{\text{size}}^{\text{inc}}$.

5.2.3.1 Solution Construction

When function ProbabilisticSolutionConstruction(S_{bsf}, α_{bsf}, l_{size}) is called, one of two heuristics is randomly selected for solution construction. The first one is our version of the Clarke-Wright Savings Algorithm [41], while the second one is our insertion algorithm. In the following, both construction algorithms are

[3] Remember that solutions constructed with a high value of α_{bsf} will be rather similar to S_{bsf}.

described in detail.

1. Probabilistic Clarke-Wright Savings Algorithm. Our probabilistic version of the Clarke-Wright savings algorithm starts by assigning each customer either to the nearest satellite or to the satellite to which it is assigned in S_{bsf}. After the assignment, set $N_C^s \subseteq N_C$ contains all customers assigned to satellite s, for all $s \in N_S$. Then, the following Clarke-Wright savings procedure is applied concerning each satellite $s \in N_S$. First, a set of direct routes $R^2 = \{(s - i - s) \mid i \in N_C^s\}$ is created. Subsequently, a savings list L that contains all possible pairs (i, j) of nodes (customers and charging stations) together with their respective savings value σ_{ij} is generated. Hereby, σ_{ij} is calculated as follows:

$$\sigma_{ij} := d_{si}^2 + d_{sj}^2 - \lambda d_{ij}^2 + \mu |d_{si}^2 - d_{sj}^2| \tag{5.38}$$

The so-called *route shape parameter* λ adjusts the selection priority based on the distance between nodes i and j [193], while μ is used to scale the asymmetry between nodes i and j [143]. Note that well-working values for these parameters are obtained by parameter tuning which is presented in Section 5.3.2. Note also that the savings list L contains, at all times, only those entries (i, j) such that (1) node i and node j belong to different routes, and (2) both i and j are directly connected to the satellite of their route. For executing the Clarke-Wright Savings Algorithm, the following list of steps is iterated until the savings list L is empty.

1. First, based on the current savings values of the entries in L, a new value q_{ij} is calculated for each entry $(i, j) \in L$ as follows:

$$q_{ij} := \begin{cases} (\sigma_{ij} + 1) \cdot \alpha_{\text{bsf}} & \text{if } S_{ij}^{\text{bsf}} = 1 \\ (\sigma_{ij} + 1) \cdot (1 - \alpha_{\text{bsf}}) & \text{otherwise} \end{cases} \tag{5.39}$$

   Here, $S_{ij}^{\text{bsf}} = 1$ if node i and node j are successively visited in at least one route of S_{bsf}, and 0 otherwise. The savings list L is then sorted according to non-increasing values of q_{ij}. Finally, a reduced saving list L_r that contains the first (maximally) l_{size} elements of the whole savings list is created.

2. Next, an entry (i, j) is chosen from L_r with respect to the following probabilities:

$$\mathbf{p}(ij) := \frac{q_{ij}}{\sum_{(i',j') \in L_r} q_{i'j'}} \quad \forall\, (i, j) \in L_r \tag{5.40}$$

   Note that, the higher the value of parameter $\alpha_{\text{bsf}} \in [0, 1]$, the stronger is the bias towards choosing arcs—that is, transitions from a customer i to a

Table 5.2 Cases for tour merging w.r.t. nodes i and j

Case	Tours	Merging Procedure	Result
1	$t_1^2 : \{s \to i \to \ldots \to s\}$ $t_2^2 : \{s \to j \to \ldots \to s\}$	Reverse t_1^2, $\mathsf{rev}(t_1^2)$ Concatenate with t_2^2	$t_m^2 : \{s \to \ldots \to i \to j \to \ldots \to s\}$
2	$t_1^2 : \{s \to i \to \ldots \to s\}$ $t_2^2 : \{s \to \ldots \to j \to s\}$	Reverse both t_1^2 and t_2^2 $\mathsf{rev}(t_1^2)$, $\mathsf{rev}(t_2^2)$	$t_m^2 : \{s \to \ldots \to i \to j \to \ldots \to s\}$
3	$t_1^2 : \{s \to \ldots \to i \to s\}$ $t_2^2 : \{s \to j \to \ldots \to s\}$	Concatenate t_1^2 and t_2^2	$t_m^2 : \{s \to \ldots \to i \to j \to \ldots \to s\}$
4	$t_1^2 : \{s \to \ldots \to i \to s\}$ $t_2^2 : \{s \to \ldots \to j \to s\}$	Reverse t_2^2, $\mathsf{rev}(t_2^2)$ Concatenate with t_1^2	$t_m^2 : \{s \to \ldots \to i \to j \to \ldots \to s\}$

customer j—that appear in the best-so-far solution S_{bsf}.

3. Then, the two routes corresponding to nodes i and j are merged. The four possible cases for merging two routes are shown in 5.2.

   Based on the way in which nodes i and j are directly connected to a satellite, one or both of the routes must be reversed in order to be able to connect nodes i and j. In this context, note that the reversed version of a route t_1^2 is denoted by $\mathsf{rev}(t_1^2)$. If the merged route t_m^2 is infeasible in terms of vehicle capacity, the merged route is eliminated and the respective pair of nodes is removed from the savings list. A new candidate is selected following the procedure in the previous step. If the merged route is battery infeasible, a charging station is inserted into the infeasible route. The corresponding procedure determines the first customer in the route at which the vehicle arrives with a negative battery level and inserts the charging station between this customer and the previous customer. For this purpose, the charging station that least increases the route distance is selected and inserted between the respective nodes. If this insertion is not feasible, the previous arcs are considered instead in the same manner. In those cases in which the route is still infeasible after charging station insertion, it is eliminated, and the respective pair of nodes are removed from the savings list. A new candidate is selected following the procedure described in the previous step. This procedure is repeated while the savings list is not empty. After merging, some of the charging stations that were previously added to the routes may become redundant. Those charging stations are removed from the merged route.

4. The savings list L must be updated as described above.

After constructing the routes in the second echelon, the same procedure is

applied to construct routes for the large vehicles in the first echelon. The first difference in the procedure for first echelon routes is that all aspects related to batteries and charging stations are not considered. Second, a satellite is allowed to be visited by multiple large vehicles in case the demand exceeds the vehicle's loading capacity.

2. Probabilistic Insertion Algorithm. This heuristic constructs a solution by sequentially inserting each customer into the available routes until no unvisited customer remains. Similar to the Clarke-Wright Savings Algorithm, the algorithm first constructs the routes for the second echelon. The first route is initialized by inserting a randomly chosen customer between the satellite that is nearest to this customer. Then, a cost list L formed by all unvisited customers and all possible insertion positions together with their respective cost values is generated. The insertion cost of customer i between nodes j and k is calculated using the following equation: $c(j,i,k) = d_{ji}^2 + d_{ik}^2 - d_{jk}^2$. Then, q_{jik} is calculated for each entry $(j,i,k) \in L$ as follows:

$$q_{jik} := \begin{cases} (c(j,i,k)+1) \cdot (1-\alpha_{\text{bsf}})(1-\alpha_{\text{bsf}}) & \text{if } S_{ji}^{\text{bsf}} = 1 \text{ and } S_{ik}^{\text{bsf}} = 1 \\ (c(j,i,k)+1) \cdot (\alpha_{\text{bsf}})^2 & \text{if } S_{ji}^{\text{bsf}} = 0 \text{ and } S_{ik}^{\text{bsf}} = 0 \\ (c(j,i,k)+1) \cdot \alpha_{\text{bsf}}(1-\alpha_{\text{bsf}}) & \text{otherwise} \end{cases} \tag{5.41}$$

Next, an entry (j,i,k) is chosen from L_r with respect to the probabilities calculated using Eq. (5.41). The customer is inserted into the respective position if the vehicle capacity allows for this. Moreover, in case of battery infeasibility, a charging station is inserted into the route as explained above during the description of the Clarke-Wright Savings Algorithm. If the insertion leads to infeasibility in terms of vehicle load capacity, a new tour is initialized with the respective customer and the nearest satellite.

5.3 Experimental Evaluation

All experiments were performed on a cluster of machines with Intel® Xeon® 5670 CPUs with 12 cores of 2.933 GHz and a minimum of 32 GB RAM. CPLEX version 20.1 was used in one-threaded mode within Adapt-CMSA for solving the respective sub-instances and in standalone mode for solving the MILP models representing the complete problem instances.

Table 5.3 Parameters, their domains, and the chosen values as determined by irace.

Parameter	Domain	Value	Description
λ	$[1, 2]$	1.67	route shape parameter (Clarke-Wright)
μ	$[0, 1]$	0.32	asymmetry scaling (Clarke-Wright)
$l_{\text{size}}^{\text{init}}$	$\{3, 5, 10, 15, 20\}$	15	initial list size value
$l_{\text{size}}^{\text{inc}}$	$\{1, 3, 5, 10, 20\}$	20	list size increment
n^{init}	$\{100, 200, 300, 500, 1000\}$	1000	initial nr. of constructed solutions
n^{inc}	$\{50, 100, 200, 300, 400, 500\}$	100	increment for the nr. of constructed solutions
t_{ILP}	$\{5, 10, 15, 20, 25\}$	20	CPLEX time limit (seconds)
α^{LB}	$[0.6, 0.99]$	0.77	lower bound for α_{bsf}
α^{UB}	$[0.6, 0.99]$	0.8	upper bound for α_{bsf}
α_{red}	$[0.01, 0.1]$	0.02	step size reduction for α_{bsf}
t_{prop}	$[0.1, 0.8]$	0.79	control parameter for bias reduction

5.3.1 Problem Instances

A subset of the 2E-EVRP problem instances introduced by [29] was utilized to assess the performance of the proposed algorithm. These instances were originally derived from the 2EVRP instances known as Set2 and Set3 from [147], Set5 from [91], and Set6 from [14]. To maintain consistency with existing literature, the instance names have been retained. The characteristics of these instances are presented in the first three columns of each results table. Since the original problem instances only come with delivery demands, we had to modify them by adding pickup demands. For this purpose, the delivery demand of each customer was separated into delivery and pickup demands using the approach from [161]. The resulting instances are provided at `https://github.com/manilakbay/2E-EVRP-SPD`.

5.3.2 Parameter Tuning

In order to find well-working parameter values for Adapt-CMSA we utilized the scientific tuning software irace [124]. Instances `100-5-1`, `100-5- 2b`, `100-10-1`, `100-10-2b`, `200-10-1`, and `200-10-2b` were used for the tuning process. Note that in the case of numerical parameters, the precision of irace was fixed to two positions behind the comma. irace was applied with a budget of 2000 algorithm applications. The time limit for each problem instance was set to 900 CPU seconds. A summary of the parameters, their domains, and values selected for the final experiments are provided in Table 5.3.

5.3.3 Computational Results

In the context of small problem instances, we compare the performance of Adapt-CMSA with the standalone application of CPLEX. As CPLEX is not applicable in a standalone manner to the large-size problem instances, we compare Adapt-CMSA with our probabilistic Clarke-Wright Savings Algorithm (pC&W) and our probabilistic sequential insertion algorithm (pSI). The parameters of both algorithms were set in the same way as for their application within Adapt-CMSA. Moreover, the same computation time limit was used as for Adapt-CMSA, that is, both algorithms were repeatedly applied until a computation time limit of 150 CPU seconds (small problem instances), respectively 900 CPU seconds (large problem instances), was reached. Moreover, Adapt-CMSA, pC&W and pSI were applied 10 times to each problem instance. In order to make a fair comparison, after each application of pC&W and pSI, LocalSearch1() and LocalSearch2() are sequentially applied in order to improve the generated solutions. Finally, note that we fixed the cost of each large vehicle used in a solution to 1500 and the cost of each electric vehicle used in a solution to 1000.

The structure of the result tables is as follows. Instance names are given in the first column. The subsequent block of three columns indicates the number of satellites, the number of charging stations and the number of customers. Columns 'n_{lv}' and 'n_{ev}' provide the number of large, respectively electric, vehicles utilized by the respective solutions. In the case of Adapt-CMSA, pC&W and pSI these numbers refer to the best solution found within 10 independent runs. In the case of Adapt-CMSA, pC&W, and pSI, columns 'best' show the distance values of the best solutions found in 10 runs, while additional columns with the heading 'avg.' provide the average distance values of the best solutions of each of the 10 runs. Next, columns with the heading 'time' show the computation time (in seconds) of CPLEX and the average computation times of Adapt-CMSA to find the best solutions in each run. Note that the time limit for CPLEX was set to 12 hours. Finally, columns 'gap(%)' provide the gap (in percentage) between the best-found solutions and the best lower bounds found by CPLEX. Note that, in case the gap value is zero, CPLEX has found an optimal solution.

The computational results highlight several key observations regarding the performance of CPLEX and Adapt-CMSA across different instance sizes. First, for the small-sized instances as detailed in Tables 5.4 and 5.5, CPLEX provided feasible solutions for only nine out of twelve cases and failed to prove optimality within a 12-hour computation limit. For the remaining three instances, CPLEX could not even find a feasible solution. Adapt-CMSA could not only find the

Table 5.4 Computational results for small-sized instances - Set2.

Instances	Characteristics			CPLEX					Adapt-CMSA				
name	n_s	n_r	n_c	n_{lv}	n_{ev}	best	time	gap(%)	n_{lv}	n_{ev}	best	avg.	time
n22-k4-s6-17	2	4	21	2	**3**	**5174**	43076.7	12.8	2	**3**	**5174**	5174.0	2.1
n22-k4-s8-14	2	4	21	2	**3**	**4870**	7465.1	15.4	2	**3**	**4870**	4870.0	1.9
n22-k4-s9-19	2	4	21	2	**3**	**4750**	43070.1	8.2	2	**3**	**4750**	4750.0	3.1
n22-k4-s10-14	2	4	21	2	**3**	**5442**	43075.3	19.9	2	**3**	**5442**	5442.0	5.9
n22-k4-s11-12	2	4	21	2	**3**	5357	43019.9	34.8	2	**3**	**5290**	5318.8	9.9
n22-k4-s12-16	2	4	21	2	**3**	**3691**	43074.0	8.4	2	**3**	**3691**	3695.7	3.4
average	-	-	-	2	**3**	4880.7	37130.2	16.6	2	**3**	**4869.5**	**4875.1**	4.4

Table 5.5 Computational results for small-sized instances - Set3.

Instances	Characteristics			CPLEX					Adapt-CMSA				
name	n_s	n_r	n_c	n_{lv}	n_{ev}	best	time	gap(%)	n_{lv}	n_{ev}	best	avg.	time
n22-k4-s13-14	2	4	21	2	4	5658	42996.9	17.0139	2	**3**	**5918**	5918.0	10.29
n22-k4-s13-16	2	4	21	2	**3**	-	-	-	2	**3**	**6584**	6584.0	5.39
n22-k4-s13-17	2	4	21	2	**3**	**5325**	43073	12.4332	2	**3**	**5325**	5360.0	34.73
n22-k4-s14-19	2	4	21	2	**3**	-	-	-	2	**3**	**6167**	6168.3	12.47
n22-k4-s17-19	2	4	21	2	**3**	-	-	-	2	**3**	**6517**	6517.0	35.19
n22-k4-s19-21	2	4	21	2	**3**	**5505**	43071.1	18.3625	2	**3**	**5505**	5505.0	1.84
average	-	-	-	-	-	-	-	-	-	3	**6002.7**	6008.7	16.65

Table 5.6 Computational results for medium-sized instances - Set2.

Instances	Characteristics			CPLEX					Adapt-CMSA				
name	n_s	n_r	n_c	n_{lv}	n_{ev}	best	time	gap(%)	n_{lv}	n_{ev}	best	avg.	time
n33-k4-s1-9	2	4	21	2	**3**	7506	19749.5	10.2	2	**3**	**7479**	7499.7	76.5
n33-k4-s2-13	2	4	21	2	**3**	**7358**	43058.1	12.2	2	**3**	**7358**	7365.9	68.7
n33-k4-s3-17	2	4	21	-	-	-	-	-	2	**3**	**7538**	7567.8	42.4
n33-k4-s4-5	2	4	21	-	-	-	-	-	2	**3**	**7947**	8122.5	14.9
n33-k4-s7-25	2	4	21	2	**3**	**7880**	43054.2	9.3	2	**3**	**7880**	7887.8	15.2
n33-k4-s14-22	2	4	21	-	-	-	-	-	2	**3**	**8173**	8173.0	17.7
average	-	-	-	-	-	-	-	-	2	**3**	**7729.2**	7769.5	39.2

Table 5.7 Computational results for medium-sized instances - Set3.

Instances	Characteristics			CPLEX					Adapt-CMSA				
name	n_s	n_r	n_c	n_{lv}	n_{ev}	best	time	gap(%)	n_{lv}	n_{ev}	best	avg.	time
n33-k4-s16-22	2	4	21	2	3	7057	43076.6	15.0971	2	3	6828	6843.9	23.90
n33-k4-s16-24	2	4	21	-	-	-	-	-	2	3	6996	7031.6	29.02
n33-k4-s19-26	2	4	21	2	3	7015	43084.7	30.9287	2	3	6926	7002.5	12.96
n33-k4-s22-26	2	4	21	2	3	6953	43077.6	13.3979	2	3	6840	6958.1	40.44
n33-k4-s24-28	2	4	21	2	3	8676	14158.1	23.6482	2	3	6744	6864.4	22.44
n33-k4-s25-28	2	4	21	2	3	7436	10677.8	17.2142	2	3	6928	6928.0	5.48
average	-	-	-	-	-	7427.4	30814.96	20.06	-	-	6877.0	6938.1	22.37

same results provided by CPLEX in seven of these instances but also improved the solutions obtained by CPLEX, respectively was able to provide a solution in those cases in which CPLEX failed to provide one. In the medium-sized instances, detailed in Tables 5.6 and 5.7, CPLEX managed to find feasible solutions for eight out of twelve instances without proving optimality within the same

time constraints. Here, Adapt-CMSA was able to match CPLEX's results in two cases and surpassed CPLEX's performance in the remaining instances by either improving solutions or finding solutions in cases where CPLEX failed. Moreover, the average computation time required for Adapt-CMSA to find the best solution in each run is considerably lower than the time required for CPLEX. More specifically, while CPLEX found its best solutions on average in almost 12 hours, Adapt-CMSA was able to do this on average in approx. 10 seconds for the small-sized instances. Concerning the larger scenarios in Set6a of the medium and large-sized instances, as shown in Tables 5.8 and 5.9, Adapt-CMSA significantly outperforms both pC&W and pSI, both in terms of best performance and average performance. Note that, even though pC&W provides better distance values than Adapt-CMSA, this was achieved using more electric vehicles than the solution found by Adapt-CMSA.

Table 5.8 Computational results for medium-sized instances - Set6a.

Instances	Characteristics			pC&W					pIns					Adapt-CMSA				
name	n_s	n_r	n_c	n_{lv}	n_{ev}	best	avg.	time	n_{lv}	n_{ev}	best	avg.	time	n_{lv}	n_{ev}	best	avg.	time
A-n51-4	4	5	50	1	6	7700	7757.6	46.02	1	3	7418	7599.9	64.02	1	3	**7043**	**7366.9**	41.44
A-n51-5	5	6	50	1	5	8314	8411.7	13.45	1	3	7716	7913.2	100.91	1	3	**7619**	**7684.6**	48.80
A-n51-6	6	7	50	1	6	7955	7983	52.77	1	3	7132	7375.5	90.66	1	3	**7033**	**7071.3**	45.83
A-n76-4	4	7	75	2	9	10430	10553.1	139.71	2	8	9911	10361.7	54.90	2	6	**9447**	**9794.8**	126.71
A-n76-5	5	7	75	2	10	11330	11465.9	139.92	2	8	10506	11012.4	50.44	2	6	**10265**	**10625.7**	140.46
A-n76-6	6	7	75	2	9	10898	10992.5	100.85	2	8	9699	9978.7	85.08	2	6	**9263**	**9538.5**	118.29
B-n51-4	4	5	50	1	6	7344	7392.6	44.48	1	3	6662	7028.5	49.79	1	3	**6311**	**6480.1**	62.59
B-n51-5	5	6	50	1	5	8032	8054	11.12	1	3	7107	7367.9	84.30	1	3	**7020**	**7085.2**	64.94
B-n51-6	6	7	50	1	6	7490	7511.3	3.89	1	3	6610	6805.4	61.42	1	3	**6302**	**6375.8**	63.06
B-n76-4	4	7	75	2	9	10473	10563.7	104.10	2	8	9968	10216.3	61.75	2	7	**9650**	**9814.2**	129.49
B-n76-5	5	7	75	2	10	9803	10044.2	106.37	2	9	9285	9727.7	72.60	2	7	**8910**	**9114**	121.78
B-n76-6	6	7	75	2	9	9295	9388.9	78.18	2	8	8394	8851.2	82.52	2	6	**8008**	**8292.5**	140.11
C-n51-4	4	5	50	1	5	9327	9518.7	9.24	1	3	7980	8426.9	82.68	1	3	**7317**	**7624.9**	65.36
C-n51-5	5	6	50	1	5	9663	9719.6	0.28	1	3	8829	9018.6	50.39	1	3	**8319**	**8519.9**	56.46
C-n51-6	6	7	50	1	6	8724	8851.2	28.27	1	3	7747	8165.5	69.09	1	3	**7233**	**7448.1**	77.55
C-n76-4	4	7	75	2	7	12842	13074.5	119.72	2	8	11921	12173.4	83.04	2	6	**10862**	**11378.5**	159.41
C-n76-5	5	7	75	2	9	14632	14931.3	128.82	2	8	12502	12809.5	77.33	2	6	**11333**	**11640.8**	144.47
C-n76-6	6	7	75	2	9	12808	13042.6	78.68	2	8	11596	12228	95.27	2	6	**10813**	**11448.5**	128.38
average	-	-	-	-	-	9836.7	9958.7	66.99	-	-	8943.5	9281.1	73.12	-	-	**8486.0**	**8739.1**	96.39

Table 5.9 Computational results for large-sized instances - Set5.

Instances	Characteristics			pC&W					pIns					Adapt-CMSA				
name	n_s	n_r	n_c	n_{lv}	n_{ev}	best	avg.	time	n_{lv}	n_{ev}	best	avg.	time	n_{lv}	n_{ev}	best	avg.	time
100-5-1	5	10	100	2	15	13402	13742.7	375.2	2	20	13895	14482.2	552.6	2	**13**	**12428**	**13050.5**	446.5
100-5-1b	5	10	100	2	8	10083	10592.4	534.6	2	9	10278	11024.3	472.2	2	**6**	**9419**	**9727.6**	336.8
100-5-2	5	10	100	2	17	8277	8663.4	582.2	2	20	8729	9153.4	426.5	2	**16**	**7982**	**8161.3**	639.1
100-5-2b	5	10	100	2	8	7173	7352.9	612.1	2	9	7076	7425.2	360.4	2	**7**	**6689**	**7198.1**	494.7
100-5-3	5	10	100	3	15	9480	9610.0	601.7	2	19	9422	10227.3	390.1	2	**13**	**8487**	**9018.3**	674.4
100-5-3b	5	10	100	2	15	8901	9607.7	419.3	2	20	9385	9942.1	367.5	2	**13**	**8581**	**8984.0**	512.1
100-10-1	10	11	100	2	20	9936	10038.1	632.6	2	22	10188	10631.0	421.8	2	**16**	**9767**	**10176.8**	364.4
100-10-1b	10	11	100	2	9	9268	9344.9	456.0	2	10	9252	9593.9	547.9	2	**7**	**8745**	**9013.6**	502.9
100-10-2	10	11	100	2	18	8491	8610.5	560.5	2	20	9218	9380.2	461.3	2	**15**	**8036**	**8450.4**	476.0
100-10-2b	10	11	100	2	9	7769	8000.9	587.6	2	9	7907	8365.5	419.4	2	**7**	**7106**	**7335.7**	377.7
100-10-3	10	11	100	2	18	**8620**	**8734.0**	641.3	2	19	9163	9419.2	269.7	2	**13**	8803	9038.8	354.1
100-10-3b	10	11	100	2	11	7821	7828.6	276.3	2	8	8189	8433.4	661.6	2	**7**	**7501**	**7713.9**	559.0
200-10-1	10	20	200	2	36	13674	14020.8	612.2	2	46	14458	14902.1	396.3	2	**31**	**13622**	**14013.9**	771.8
200-10-1b	10	20	200	2	18	11356	11699.2	354.8	2	21	11661	12086.5	485.1	2	**14**	**10977**	**11529.5**	777.2
200-10-2	10	20	200	2	34	10718	11037.1	625.9	2	48	12151	12296.5	316.8	2	**31**	**10816**	**11452.6**	698.0
200-10-2b	10	20	200	2	17	8773	8922.9	683.6	2	22	9574	9992.5	478.0	2	**15**	**9110**	**9501.6**	745.9
200-10-3	10	20	200	2	33	14494	14689.4	441.2	2	46	15357	15836.1	299.7	2	**29**	**13869**	**14247.3**	840.9
200-10-3b	10	20	200	2	18	10922	11170.3	472.1	2	22	11357	11785.8	479.5	2	**13**	**9943**	**10301.8**	770.0
average	-	-	-	2.1	17.7	9953.2	10203.7	526.1	**2.0**	21.7	10403.3	10832.1	433.7	**2.0**	**14.8**	**9548.9**	**9939.8**	574.5

5.4 Discussion and Conclusions

This chapter described the application of Adapt-CMSA to the two-echelon electric vehicle routing problem with simultaneous pickup and deliveries. At each iteration, the algorithm first creates a sub-instance of the considered problem instance by merging the best-so-far solution with a number of solutions probabilistically generated using two different solution construction mechanisms, a Clarke-Wright Savings Heuristic, and an insertion heuristic. The resulting sub-instance is then solved by the application of the MILP solver CPLEX. Preliminary computational experiments showed that making use of the classical MILP model for the purpose of solving sub-instances was not feasible. Therefore, a set-covering-based model was used and solved with CPLEX. Computational experiments were performed on 12 small- and medium-sized problem instances and on 18 large-sized instances. The proposed approach was evaluated and compared to CPLEX in the context of the small- and medium-sized problem instances, and to probabilistic versions of the Clarke-Wright Savings Heuristic and the insertion heuristic for large-sized instances. Numerical results indicated that Adapt-CMSA exhibits superior performance for problem instances of all size ranges. In the future, we aim to deepen the analysis of the algorithm on a wider set of instances.

CHAPTER 6

APPLICATION OF VARIABLE NEIGHBORHOOD SEARCH TO THE TWO-ECHELON ELECTRIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

6.1 Introduction

This chapter demonstrates the application of a Variable Neighborhood Search Algorithm to the *Two-Echelon Electric Vehicle Routing Problem with Time Windows* (2E-EVRP-TW). The content shown in this chapter was also presented in our paper [3] that was published in the Applied Science Journal (`https://doi.org/10.3390/app12031014`).

First, we define the 2E-EVRP-TW problem by means of a three-index node-based mixed-integer linear programming (MILP) model. Any general-purpose MILP solver, such as CPLEX or Gurobi, may be used to solve this model. However, due to the multi-tier structure of the distribution network, the limited driving range of electric vehicles, and the time window constraints, the 2E-EVRP-TW problem is rather complex. In fact, our computational experiments show that CPLEX is only able to solve small-sized problem instances to optimality. Therefore, we also developed a variable neighborhood search (VNS) approach to solve the problem[1] In addition, we developed an initial solution generation method based on Clarke-Wright Savings Algorithm [41], considering 2E-EVRP-TW assumptions and characteristics. The VNS approach makes use of this heuristic to obtain an initial solution.

As described in Section 1.3.3.3, VNS provides a powerful search framework by systematically changing neighborhood structures (shaking) to avoid getting stuck in local optima and by intensifying the search in the vicinity of the incumbent solution by applying local search. In addition to the classical shaking and local search operators, we also utilize large neighborhood search (LNS) operators known as "destroy and repair", resp. "removal and insertion", to enhance the

[1] Note that this work was chronologically done before all the works on CMSA and Adapt-CMSA.

performance of VNS. In this context, note that, since (1) the problem dimension of the first echelon is much smaller than that of the second echelon and (2) the first echelon does not involve any constraint, it can easily be solved using a savings heuristic. Therefore, whenever the solution for the second echelon changes, the first echelon tours are generated again by utilizing our Clarke-Wright Savings Heuristic.

Finally, a last contribution concerns the generation of new problem-specific benchmark sets. This was necessary due to the lack of an available benchmark set for the 2E-EVRP-TW problem.

6.1.1 Problem Description and Mathematical Model

Given is a directed graph $G = (N, A)$ in which the set of nodes (N) is composed of the following four subsets: the set of central warehouses (N_D), the set of satellites (N_S), the set of charging stations (N_R), and the set of customers (N_C). The set of arcs (A) includes (1) arcs that connect central warehouses and satellites $A1 = \{(i, j)|i \neq j \text{ and } i, j \in N_D \cup N_S\}$ and (2) arcs that connect satellites, customers and charging stations $A2 = \{(l, m)|l \neq m \text{ and } l, m \in N_S \cup N_R \cup N_C\}$. In other words, $A1$ contains the arcs of the first echelon, and $A2$ contains the arcs of the second echelon. Traveling along an arc $(i, j \in A1)$, $(l, m \in A2)$, has a cost/distance of $d1_{ij}$, $d2_{lm}$. Each customer $i \in N_C$ has a demand $D2_i$ and a time window that indicates the earliest possible visiting time twe_i and the latest possible visiting time twl_i. Two different fleets of vehicles, each one homogeneous in itself, serve in the first and second echelons in order to meet customer demands. A fleet of large trucks $V1$ with internal combustion engines are located in a central warehouse and carry products from the central warehouses to the satellites, while a fleet of electric vehicles $V2$ are present at the satellites and distribute products to customers (demand points). In the first echelon, a truck $k \in V1$ starts its tour from a central warehouse, visits one or more satellites, and returns to the central warehouse from which the tour started. The total amount of deliveries may not exceed the load capacity $Q1_k$ of vehicle k. In the second echelon, an electric vehicle $e \in V2$ starts its tour from a satellite, visits one or more customers and charging stations if necessary, and returns to the satellite from which the tour started. The total amount of deliveries cannot exceed the load capacity $Q2_e$ of electric vehicle e. A customer can only be served by one electric vehicle. An electric vehicle starts its tour with a fully charged battery (battery level BC_e) and the vehicle's battery is consumed in proportion to the distance traveled. If a charging station is visited, the electric vehicle's battery is fully charged up to level BC_e with a constant charging speed.

Note that an electric vehicle may need to visit a charging station multiple times. Therefore, set N_R includes charging stations as well as copies of each charging station in order to allow multiple visits to any charging station. The idea of using such dummy vertices was introduced for the first time in [16] in order to permit multiple visits to intermediate satellites. Moreover, this approach was adopted in [62] for a green vehicle routing problem. Determining the number of copies of each charging station (ψ) is, however, not a trivial task. An insufficient number of copies may prevent finding an optimal solution due to not allowing a sufficient number of multiple visits of the same charging station. On the other hand, an unnecessarily large number of copies of each charging station would increase the model size, resulting in longer running times of the MILP solvers. As a result of preliminary experiments on various instance sets, we set ψ to 3.

We developed a three-index node-based integer programming model for 2E-EVRP-TW. Including the vehicles as a third index ensures that the model may be used for instances that contain heterogeneous fleets with different vehicle characteristics. In the following, we first introduce the notations, sets and problem data used by the MILP model. Subsequently, the model is outlined in terms of the decision variables, the objective function and the constraints.

Table 6.1 Notations and Sets

Symbol	Description
n_d	Number of central warehouses
n_s	Number of satellites
n_{cs}	Number of charging stations
ψ	Number of copies of each charging station
n_c	Number of customers
nv_1	Number of available vehicles in the first echelon
nv_2	Number of available electric vehicles
N_D	Set of central warehouses (node indices: $1, \ldots, n_d$)
N_S	Set of satellites (node indices: $n_d + 1, \ldots, n_d + n_s$)
N_R	Set of charging stations and their copies (node indices: $n_d+n_s+1, \ldots, n_d+n_s + n_{cs} \cdot \psi$)
N_C	Set of customers (node indices: $n_d + n_s + n_{cs} \cdot \psi + 1, \ldots, n_d + n_s + n_{cs} + n_c$)
N_{DS}	Set of central warehouses and satellites (node indices: $1, \ldots, n_d + n_s$)
N_{RC}	Set of charging stations and customers (node indices: $n_d+n_s+1, \ldots, n_d+n_s + n_{cs} \cdot \psi + n_c$)
N_{SRC}	Set of satellites, charging stations, and customers (node indices: $n_d + 1, \ldots, n_d + n_s + n_{cs} \cdot \psi + n_c$)
N	Set of all nodes including warehouses, satellites, charging stations, and customers (node indices: $1, \ldots, n_d + n_s + n_{cs} \cdot \psi + n_c$)
$V1$	Set of large vehicles serving in the first echelon ($\|V1\| = nv_1$)
$V2$	Set of electric vehicles ($\|V2\| = nv_2$)

Table 6.2 Problem Data

Symbol	Description
$d1_{ij}$	Distance between node i and node j, $(i, j \in N_{DS})$
$d2_{lm}$	Distance between node l and node m, $(l, m \in N_{SRC})$
$Q1_k$	Loading capacity of large vehicle $k \in V1$
$Q2_e$	Loading capacity of electric vehicle $e \in V2$
M	A large constant number
BC_e	Battery capacity of electric vehicle $e \in V2$
g_e	Charging rate of electric vehicle $e \in V2$
$D2_i$	Demand of customer i, $(\forall i \in N_C)$
s_i	Service time of customer i, $(\forall i \in N_C)$
twe_i	Earliest visiting time of customer i, $(\forall i \in N_C)$
twl_i	Latest visiting time of customer i, $(\forall i \in N_C)$

Table 6.3 Decision Variables

Symbol	Description
x_{kij}	$\begin{cases} 1 & \text{if vehicle } k \text{ visits node } j \text{ after node } i \text{ in the first echelon,} \\ 0 & \text{otherwise, } \forall k \in V1, \forall i, j \in N_{DS} \end{cases}$
y_{elm}	$\begin{cases} 1 & \text{if vehicle } e \text{ visits node } m \text{ after node } l \text{ in the second echelon,} \\ 0 & \text{otherwise, } \forall e \in V2, \forall l, m \in N_{SRC} \end{cases}$
z_{li}	$\begin{cases} 1 & \text{if customer } l \text{ gets service from satellite } i, \\ 0 & \text{otherwise} \end{cases}$
$U1_{kij}$	Amount of product in vehicle k traveling from node i to node j (first echelon), $\in \{0, \ldots, Q1_k\}$
$U2_{elm}$	Amount of product in vehicle e traveling from node l to node m (second echelon), $\in \{0, \ldots, Q2_e\}$
$BSCa_{le}$	Battery level of electric vehicle e at arrival to node l, $\in [0, BC_e]$
$BSCd_{le}$	Battery level of electric vehicle e when departing from node l, $\in [0, BC_e]$
$D1_j$	Demand of satellite j, $\in \{0, \ldots, \sum_{i \in N_C} D2_i\}$
$w1_{ki}$	Visiting time of node i by vehicle k (first echelon), $\in [0, twl_i]$
$w2_{el}$	Visiting time of node l by vehicle e (second echelon), $\in [twe_l, twl_l]$

MILP model

$$\textbf{Min} \quad \sum_{k\in V1}\sum_{i\in N_{DS}}\sum_{j\in N_{DS}} d1_{ij}\cdot x_{kij} + \sum_{e\in V2}\sum_{l\in N_{SRC}}\sum_{m\in N_{SRC}} d2_{lm}\cdot y_{elm} \tag{6.1}$$

s.t.

$$\sum_{k\in V1}\sum_{i\in N_{DS}} x_{kij} = 1 \qquad \forall j \in N_S \tag{6.2}$$

$$\sum_{i\in N_{DS}} x_{kij} - \sum_{i\in N_{DS}} x_{kji} = 0 \qquad \forall k \in V1, \forall j \in N_{DS} \tag{6.3}$$

$$\sum_{j\in N_S} x_{kij} \leq 1 \qquad \forall i \in N_D, \forall k \in V1 \tag{6.4}$$

$$\sum_{i\in N_S}\sum_{j\in N_D} x_{kij} \leq 1 \qquad \forall k \in V1 \tag{6.5}$$

$$x_{kij} = 0 \qquad \forall k \in V1, \forall i,j \in N_D \tag{6.6}$$

$$\sum_{k\in V1}\sum_{i\in N_{DS}} U1_{kij} - \sum_{k\in V1}\sum_{i\in N_{DS}} U1_{kji} \leq D1_j \qquad \forall j \in N_S \tag{6.7}$$

$$U1_{kji} = 0 \qquad \forall i \in N_D, \forall j \in N_S, \forall k \in V1 \tag{6.8}$$

$$U1_{kij} \leq Q1_k \cdot \sum_{k\in V1} x_{kij} \qquad \forall i,j \in N_{DS}, \forall k \in V1 \tag{6.9}$$

$$D1_i = \sum_{l\in N_C} z_{li}\cdot D2_l \qquad \forall i \in N_S \tag{6.10}$$

$$\sum_{e\in V2}\sum_{l\in N_{SRC}} y_{elm} = 1 \qquad \forall m \in N_C \tag{6.11}$$

$$\sum_{l\in N_{SRC}} y_{elm} - \sum_{l\in N_{SRC}} y_{eml} = 0 \qquad \forall e \in V2, \forall m \in N_{SRC} \tag{6.12}$$

$$\sum_{m\in N_{RC}} y_{elm} \leq 1 \qquad \forall e \in V2, \forall l \in N_S \tag{6.13}$$

$$\sum_{m\in N_{RC}} y_{eml} \leq 1 \qquad \forall e \in V2, \forall l \in N_S \tag{6.14}$$

$$\sum_{i\in N_S} z_{li} = 1 \qquad \forall l \in N_C \tag{6.15}$$

$$\sum_{e\in V2} y_{eli} \leq z_{li} \qquad \forall i \in N_S, \forall l \in N_C \tag{6.16}$$

$$\sum_{e\in V2} y_{eil} \leq z_{li} \qquad \forall i \in N_S, \forall l \in N_C \tag{6.17}$$

$$\sum_{i\in N_S}\sum_{j\in N_{RC}} y_{eij} \leq 1 \qquad \forall e \in V2 \tag{6.18}$$

$$y_{elm} + z_{li} + \sum_{s\in N_S, s\neq i} z_{ms} \leq 2 \qquad \forall e \in V2, \forall l \in N_C, \forall m \in N_C, l \neq m, \forall i \in N_S \tag{6.19}$$

$$\sum_{e\in V2}\sum_{l\in N_{SRC}} U2_{elm} - \sum_{e\in V2}\sum_{l\in N_{SRC}} U2_{eml} \leq D2_m \qquad \forall m \in N_{RC} \tag{6.20}$$

$$U2_{eij} = 0 \qquad \forall i \in N_{RC}, \forall j \in N_S, \forall e \in V2 \quad (6.21)$$

$$U2_{elm} \leq Q2_v \cdot \sum_{e \in V2} y_{elm} \qquad \forall l, m \in N_{SRC}, \forall e \in V2 \quad (6.22)$$

$$x_{kii} = 0 \qquad \forall i \in N_{DS}, \forall k \in V1 \quad (6.23)$$

$$y_{ell} = 0 \qquad \forall l \in N_{SRC}, \forall e \in V2 \quad (6.24)$$

$$BSCa_{le} \geq 0 \qquad \forall l \in N_{SRC}, \forall e \in V2 \quad (6.25)$$

$$BSCa_{le} = BC_e \qquad \forall l \in N_S, \forall e \in V2 \quad (6.26)$$

$$BSCa_{me} \leq BSCa_{le} - (h \cdot d2_{lm}) \cdot y_{elm} + BC_e \cdot (1 - y_{elm}) \qquad \forall l \in N_C, \forall m \in N_{SRC}, l \neq m, \forall e \in V2 \quad (6.27)$$

$$BSCa_{me} \leq BSCd_{le} - (h \cdot d2_{lm}) \cdot y_{elm} + BC_e \cdot (1 - y_{elm}) \qquad \forall l \in N_{SRC}, \forall m \in N_{SRC}, l \neq m, \forall e \in V2 \quad (6.28)$$

$$BSCa_{le} \leq BSCd_{le} \qquad \forall l \in N_{SRC}, \forall e \in V2 \quad (6.29)$$

$$BSCd_{le} \leq BC_e \qquad \forall l \in N_{SRC}, \forall e \in V2 \quad (6.30)$$

$$BSCa_{le} = BSCd_{le} \qquad \forall i \in N_C, \forall e \in V2 \quad (6.31)$$

$$BSCd_{le} = BC_e \qquad \forall l \in N_R, \forall e \in V2 \quad (6.32)$$

$$\sum_{l \in N_{SRC}} U2_{elm} = \sum_{l \in N_{SRC}} U2_{elm} \qquad \forall m \in N_R^{'}, \forall e \in V2 \quad (6.33)$$

$$w1_{ki} = 0 \qquad \forall i \in N_D, \forall k \in V1 \quad (6.34)$$

$$w1_{kj} \geq w1_{ki} + d1_{ij} + s_i - M * (1 - x_{kij}) \qquad \forall i \in N_{DS}, \forall j \in N_S^{'}, \forall k \in V1 \quad (6.35)$$

$$w2_{ej} \geq w1_{ki} + d1_{ij} + s_i - M * (1 - x_{kij}) \qquad \forall i \in N_{DS}, \forall j \in N_S^{'}, \forall e \in V2, \forall k \in V1 \quad (6.36)$$

$$w2_{ej} \geq w2_{ki} + d2_{ij} + s_i - M \cdot (2 - y_{eij} - \sum_{h \in N_{DS}} x_{khi}) \qquad \forall i \in N_S^{'}, \forall j \in N_{SRC}, \forall e \in V2, \forall k \in V1 \quad (6.37)$$

$$w2_{em} \geq w2_{el} + d2_{lm} + s_l - M \cdot (1 - y_{elm}) \qquad \forall l, m \in N_{RC}, \forall e \in V2 \quad (6.38)$$

$$w2_{em} + d2_{lm} \cdot y_{elm} + g_v \cdot (BC_e - BSCa_{le}) - (M + g_v \cdot BC_e) \cdot (1 - y_{elm}) \leq w2_{em} \qquad \forall l \in N_R^{'}, \forall m \in N_{SRC}, l \neq m, \forall e \in V2 \quad (6.39)$$

$$w1_{ki} \geq twe_i \qquad \forall i \in N_{DS}, \forall k \in V1 \quad (6.40)$$

$$w2_{el} \geq twe_l \qquad \forall l \in N_{SRC}, \forall e \in V2 \quad (6.41)$$

$$w1_{ki} \leq twl_i \qquad \forall i \in N_{DS}, \forall k \in V1 \quad (6.42)$$

$$w2_{el} \leq twl_l \qquad \forall l \in N_{SRC}, \forall e \in V2 \quad (6.43)$$

The objective function 6.1 minimizes the total distance traveled by all utilized vehicles in both echelons. Constraint 6.2 guarantees that each satellite will be visited by a truck. Constraints 6.3 and 6.12 ensure the balance of flow for

the satellites and customers, respectively. Constraints 6.4 and 6.5 ensure that vehicles in the first echelon are used only if needed. Constraint 6.6 does not allow direct transportation between central warehouses if there is more than one warehouse. Constraints 6.7 and 6.20 guarantee that the demand of each satellite and customer is met by the vehicles serving in the relevant echelon, respectively. Constraints 6.8 and 6.21 ensure that no product remains in the vehicle when returning to the central warehouse in the first echelon and to the satellite in the second echelon, respectively. Constraints 6.9 and 6.22 indicate that the vehicle capacity cannot be violated. Constraint 6.10 determines each satellite's demand to be the total demand of those customers served by the relevant satellite. Constraint 6.11 guarantees that each customer is visited only once. Constraints 6.13 and 6.14 ensure that vehicles in the second echelon are only used when they are needed. Constraint 6.15 ensures that each customer is served by only one satellite. Constraints 6.16, 6.17, and 6.19 ensure that each electric vehicle completes its tour at the same satellite from which it started the tour. Constraint 6.18 guarantees that each electric vehicle can provide service through only one satellite. Constraints 6.23 and 6.24 prevent returning to the node from which a vehicle just departed. Constraints 6.25–6.32 are battery state constraints. Constraint 6.33 states that the load of a vehicle is the same when arriving and departing from a charging station. Constraints 6.34–6.39 calculate arrival and departure times considering service and battery charging times. Moreover, constraints 6.40–6.43 restrict the visiting time of each customer with respect to the time windows.

The classical VRP is NP-Hard [117]. The multi-tier distribution structure (two echelons) and additional limitations such as the driving range of electric vehicles and customers' time windows further increase the complexity. As the computation time required to solve such complex problems to optimality increases dramatically with a growing instance size, most approaches from the related literature for similar problems are approximate techniques, especially in the context of large-sized problem instances. In this chapter, we propose an approach based on VNS to solve the 2E-EVRP-TW. Algorithm 6.2 presents the general structure of the proposed algorithm. It starts with the application of a modified version of the Clarke-Wright Savings Algorithm to obtain an initial solution quickly and efficiently. Subsequently, shaking and local search procedures are applied to improve the initial solution. However, before describing the proposed algorithm, we first explain how a solution S is represented, and subsequently we outline an extended objective function used to handle infeasible solutions.

6.2 Solution Approach

In the following, we first provide the representation of solutions and the description of an extended objective function for dealing with infeasible solutions. Then, an extended Clarke-Wright Algorithm is provided, followed by the description of the VNS procedure.

6.2.1 Solution Representation and Extended Objective Function

In our implementation, a solution S is represented by two sets of routes: $R1$ and $R2$. Each route $\tau 1 \in R1$ starts from a central warehouse, visits one or more satellites from N_S, and returns to the same central warehouse. Each route $\tau 2 \in R2$ starts from a satellite $s \in N_S$, visits a sequence of locations/nodes $v \in N_{RC}$, and returns to the same satellite. An exemplary solution for a 2E-EVRP-TW instance with a single central warehouse, two satellites, three charging stations and five customers is shown below. The vector $\mathbf{I}$ includes the complete set of node indexes of the example problem instance. Moreover, the solution S contains one route in the first echelon ($\tau 1_1$) and two routes in the second echelon ($\tau 2_1$ and $\tau 2_2$).

$$\mathbf{I} = (\underbrace{0,}_{\text{central warehouse}} \underbrace{1,\ 2,}_{\text{satellites}} \underbrace{3,\ 4,\ 5,}_{\text{charging stations}} \underbrace{6,\ 7,\ 8,\ 9,\ 10,}_{\text{customers}})$$

$$S = (R1, R2) \text{ where } R1 = \left\{ \tau 1_1 = \{0 \to 1 \to 2 \to 0\} \right\}$$

$$R2 = \left\{ \begin{array}{c} \tau 2_1 = \{1 \to 7 \to 3 \to 9 \to 1\} \\ \tau 2_2 = \{2 \to 10 \to 4 \to 8 \to 6 \to 2\} \end{array} \right\}$$

The usefulness of allowing the algorithm to visit unfeasible solutions during the search process has already been recognized in the metaheuristics community, especially in the field of evolutionary computation [134]. In this work, we do this in a similar way as in [166] in the context of the EVRP. In particular, the extended objective function that evaluates both feasible and unfeasible solutions by means of penalty values for capacity, battery, and time windows violations is defined as follows:

$$f_{ext}(S) = f(S) + \omega_c P_{cap}(S) + \omega_b P_{bat}(S) + \omega_{tw} P_{tw}(S) \tag{6.44}$$

Here, $f(S)$ refers to the objective function of the 2E-EVRP-TW problem, that

is, the sum of the distances traveled by all utilized vehicles from the first and the second echelon. Furthermore, $P_{cap}(S)$, $P_{bat}(S)$ and $P_{tw}(S)$ denote the capacity, battery and time windows violations in solution S. In this context, the function for calculating the capacity violations of a solution S with m routes in the first echelon and n routes in the second echelon is defined as follows:

$$P_{cap}(S) = \sum_{i=1}^{m} \max\Big\{\big(\sum_{j\in\tau 1_i} D1_j\big) - Q1, 0\Big\} + \sum_{k=1}^{n} \max\Big\{\big(\sum_{l\in\tau 2_k} D2_l\big) - Q2, 0\Big\} \quad (6.45)$$

In words, if the total demand of the satellites (resp. the customers) on a route exceeds the vehicle capacity, the capacity violation of the route is determined as the difference between the vehicle capacity and the total demand of the route. Otherwise, it is set to zero. Note that, in an abuse of notation, $j \in \tau 1_i$ refers to a satellite j visited by route $\tau 1_i$ and $l \in \tau 2_k$ refers to a customer l visited by route $\tau 2_k$.

Next, the total battery violation of a solution S, $P_{bat}(S)$, is calculated using Equations (6.46)-(6.47):

$$P_{bat}(S) = \sum_{k=1}^{n} P_{bat}(\tau 2_k) \quad \text{, where} \quad (6.46)$$

$$P_{bat}(\tau 2_k) = \sum_{l\in\tau 2_k} \big|\min\{BSCa_{le}, 0\}\big| \quad (6.47)$$

That is, this function sums (for all second echelon routes $\tau 2_k$) the battery level violations of the electric vehicles at the arrival to all nodes $l \in \tau 2_k$. Hereby, the term *node* refers to customers and charging stations. Finally, similar to the approach used to calculate P_{bat}, P_{tw} is calculated using Equations (6.48)-(6.49):

$$P_{tw}(S) = \sum_{k=1}^{n} P_{tw}(\tau 2_k) \quad \text{, where} \quad (6.48)$$

$$P_{tw}(\tau 2_k) = \sum_{l\in\tau 2_k} \max\big\{w2_{ei} - twl_i, 0\big\} \quad (6.49)$$

These three penalty terms are added as a weighted sum to the original objective function value (see Equation 6.44). The corresponding three weights are denoted by ω_c, ω_b and ω_{tw}. At the start of the VNS algorithm, all three weights are set to an initial value p_{init}. Then, they are dynamically updated between $p_{\min}$ and $p_{\max}$. More specifically, if any of three terms (capacity, battery, and time windows violations) are greater than zero for p_{iter} successive iterations, the respective penalty weight is increased by $p^+ > 0$. On the other hand, if the respective solution is feasible in terms of any of three constraint violation types,

the respective weight is decreased by means of a division by $p^- > 1$.

6.2.2 Initial Solution Construction

The VRP literature offers numerous heuristic approaches in order to construct initial solutions to different VRP variants. The Clarke-Wright Savings Algorithm is one of the most commonly used methods because of its simplicity, performance, and ease of adaptation to different problem variants. This chapter proposes a savings-based initial solution construction algorithm that considers the multi-tier transportation structure and additional constraints (capacity, battery, and time windows) of the 2E-EVRP-TW. Algorithm 6.1 provides a high-level pseudo-code of this procedure.

First, each customer is assigned to the nearest satellite. After this assignment, set $N_C^s \subseteq N_C$ contains all customers assigned to satellite s, for all $s \in N_S$. Note that, with this assignment, an indirect time window arises for each satellite based on the customer's time windows. Assume, for example, that goods are transported from central warehouse 0 to satellite s, and then from satellite s to customers i and j, that is, $i, j \in N_C^s$. As illustrated in Figure 6.1, the electric vehicle must depart from satellite s before a certain time in order to be able to visit customers within their time windows. Therefore, the large truck in the first echelon must deliver goods to satellite s no later than $twl_s := \min\{twl_v - d2_{sv} \mid v \in N_C^s\}$ such that the electric vehicle is able to visit customer i and j before twl_i and twl_j, respectively.

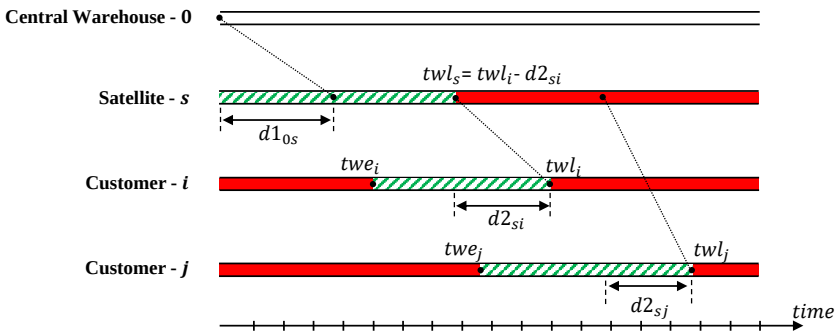


Fig. 6.1 An illustration of the indirect time windows arising for a satellite depending on the customers it must serve. Note that time windows are indicated in green color.

After calculating time windows for each satellite, first, the routes for the large vehicles in the first echelon and, second, the routes for the electric vehicles in the second echelon are constructed using the savings heuristic. In the following, we explain the steps for constructing the routes for the electric vehicles in the second echelon. In particular, for each satellite $s \in N_S$ the following steps are applied:

1. A set of direct routes $R2 = \{(s \rightarrow i \rightarrow s) \mid i \in N_C^s\}$ is created. However, note that not all of these single-customer tours are necessarily battery feasible. If this occurs, a charging station with the minimum insertion cost is inserted into the route. To achieve this, first, for each charging station $r \in N_R$ the cost $C_{insert}(r)$ of inserting r between satellite s and customer i is calculated as $C_{insert}(r) = d2_{sr} + d2_{ri} - d2_{si}$. Then, a charging station $r' \in N_R$ such that $C_{insert}(r') \leq C_{insert}(r)$ for all $r \in N_R$ is inserted into the infeasible route. Only one charging station is allowed to be inserted to fix infeasibility. In the unique case that the battery infeasibility cannot be eliminated despite charging station insertion, the relevant tour is removed, and the customer in the tour is added to the initially empty list of unvisited customers L_u.

2. Subsequently, a savings list formed by all possible pairs of nodes (customers and charging stations) together with their respective savings values is generated. A pair of nodes $(i, j \in N_{RC} \mid i \neq j)$ must fulfill the following conditions to be included in the savings list: (1) node i and node j must belong to different routes, and (2) both i and j must be directly connected to the satellite in the route to which they belong. With regard to the calculation of the savings value $s2_{ij}$ for two nodes i and j, the literature offers various enhancements and extensions. In this study, we have utilized the formulation introduced by [7]:

$$s2_{ij} = d2_{si} + d2_{sj} - \lambda d2_{ij} + \mu |d2_{si} - d2_{sj}| + \gamma \frac{D2_i + D2_j}{\overline{D}} \tag{6.50}$$

Note that, according to this formula, both the distances between nodes, as well as the customer demands have an influence on the route construction process. More precisely, the first four terms of Equation (6.50) are based on the distances between nodes, while the last term ($\frac{(D2_i + D2_j)}{\overline{D}}$) takes into account the customer demands. Hereby, $D2_i$ and $D2_j$ refer to the demands of customers i and j, while $\overline{D}$ indicates the average demand of customers in $N_C^s \setminus L_u$. As a result, tours that include customers with higher demands are prioritized during tour merging operations and vehicle capacities are used more effectively. Finally, note that the so-called *route shape parameter* λ adjusts the selection priority based on the distance between customers i and j [193], while μ is used to scale the asymmetry between customers i and j [143]. Parameter γ weights the demand information. Note that well-working values for these parameters are obtained by parameter tuning which is presented in Section 6.3.2. Finally, the savings list is sorted

Table 6.4 Cases for tour merging w.r.t. nodes i and j

Case	Tours	Merging Procedure	Result
1	$\tau 2_1 : \{s \to i \to \ldots \to s\}$ $\tau 2_2 : \{s \to j \to \ldots \to s\}$	Reverse $\tau 2_1$, $\mathsf{rev}(\tau 2_1)$ Concatenate with $\tau 2_2$	$\tau 2_m : \{s \to \ldots \to i \to j \to \ldots \to s\}$
2	$\tau 2_1 : \{s \to i \to \ldots \to s\}$ $\tau 2_2 : \{s \to \ldots \to j \to s\}$	Reverse both $\tau 2_1$ and $\tau 2_2$ $\mathsf{rev}(\tau 2_1)$, $\mathsf{rev}(\tau 2_2)$	$\tau 2_m : \{s \to \ldots \to i \to j \to \ldots \to s\}$
3	$\tau 2_1 : \{s \to \ldots \to i \to s\}$ $\tau 2_2 : \{s \to j \to \ldots \to s\}$	Concatenate $\tau 2_1$ and $\tau 2_2$	$\tau 2_m : \{s \to \ldots \to i \to j \to \ldots \to s\}$
4	$\tau 2_1 : \{s \to \ldots \to i \to s\}$ $\tau 2_2 : \{s \to \ldots \to j \to s\}$	Reverse $\tau 2_2$, $\mathsf{rev}(\tau 2_2)$ Concatenate with $\tau 2_1$	$\tau 2_m : \{s \to \ldots \to i \to j \to \ldots \to s\}$

according to non-increasing savings values.

3. At each iteration, the two routes that contain a pair of nodes (i, j) with the highest savings value $s2_{ij}$ are selected from R (e.g., $\tau 2_1, \tau 2_2$). Then, the chosen routes are merged by connecting nodes i and j. All of the merging scenarios are shown in Table 6.4.

   Based on the way in which nodes i and j are connected to the respective satellite, one or both of the routes must be reversed in order to be able to connect nodes i and j. In this context, note that the reversed version of a tour $\tau 2_1$ is denoted by $\mathsf{rev}(\tau 2_1)$. If the merged route is infeasible in terms of vehicle capacity or time windows, the route is eliminated, and merging continues considering the pair of nodes with the next-highest savings value. If the merged route is battery infeasible, a charging station r with a lowest insertion cost is inserted between node i and j (e.g., $\{s \to \ldots \to i \to r \to j \to \ldots \to s\}$). In these cases in which the route is still infeasible after charging station insertion, it is eliminated, and merging continues with the next pair of customers from the savings list. This procedure is repeated while the savings list is not empty. After merging, some of the charging stations that were previously added to the routes may become redundant. These charging stations are removed from the merged route.

4. Update the savings list as described in step 2 and repeat step 3 until no further pairs of tours can be merged.

5. Finally, the customers in L_u (the list of unvisited customers) are inserted into the constructed tours using the greedy insertion operator, which is described in detail in Section 6.2.3.3 .

Algorithm 6.1 Modified Clarke-Wright Savings Heuristic for the 2E-EVRP-TW

Assign customers to the nearest satellite
Determine the latest possible visiting time (twl_s) for each satellite $s \in N_S$
Construct first echelon routes using the savings heuristic
for each satellite s **do**
  Create back-and-forth tours for each customer $i \in N_C^s$ $(s - i - s)$
  if the created tour is infeasible in terms of the battery constraints **then**
    Insert a charging station using the *greedy CS insertion* operator (see Section 6.2.3.3)
    if the tour is still infeasible **then**
      Discard the tour and add the customer to the unvisited customer list L_u
    end if
  end if
  Generate the savings list and sort it in descending order based on the savings values
  while savings list is not empty **do**
    Merge the two tours with the greatest savings value
    if vehicle capacity or time window constraints are violated **then**
      Discard the tour and remove the corresponding pair of customers from the savings list
    else
      if the merged tour is infeasible in terms of the battery constraint **then**
        Insert a charging station with a minimum insertion cost
        if the tour is still infeasible **then**
          Discard the tour and remove the pair of customers from the savings list
        else
          Accept the merged tour and update the saving list
        end if
      else
        Accept the merged tour and update the saving list
      end if
    end if
  end while
  Insert all customers from L_u into the constructed tours using the *greedy customer insertion* operator (see Section 6.2.3.3)
end for

Finally, note that the same procedure is applied to construct routes for the large vehicles in the first echelon. In this case, all aspects related to batteries and charging stations are removed from the heuristic procedure.

6.2.3 Variable Neighborhood Search for the 2E-EVRP-TW

The initial solution constructed by our version of the Clarke-Wright savings heuristic from above is used as input for a variable neighborhood search (VNS) approach outlined in Section 1.3.3.3.

Algorithm 6.2 presents a pseudo-code of our implementation of VNS for the 2E-EVRP-TW. The proposed algorithm starts by taking an initial solution S_{init} as input. Then, the neighborhood structures used for shaking $N_k^{shake}, (k = 1, \ldots, k_{max})$ and the ones used for local search $N_h^{local}, (h = 1, \ldots, h_{max})$ are determined. These neighborhood structures will be explained in detail in following sections. However, note that, in addition to rather standard inter-route and intra-route operators, our VNS also makes use of so-called destroy-and-repair operators for the shaking step. Operators such as these ones were mostly introduced in the context of approaches based on large neighborhood search (and very large neighborhood search) algorithms and have shown to be highly useful for exploring large search spaces [148]. In particular, the reconstruction of a partially destroyed solution using various reinsertion operators has the potential to produce solutions that may have been difficult to reach otherwise. Concerning the specific case of VRP problems, removing rather large components of a solution and reinserting them into other positions of the solution may help reduce the number of routes and the number of vehicles, respectively.

After obtaining the initial solution S_{init}, both the current solution S_{cur} and the best-so-far solution S_{bsf} are initialized to S_{init}. At each main iteration of VNS, the shaking neighborhoods are randomly ordered. This order is then used until the current neighborhood utilized for shaking (indicated by k) is the k_{max}-th neighborhood. Next, a random solution S_{shake} is chosen from the current shaking neighborhood k. In case this neighborhood is a removal/destroy operator, the partially destroyed solution must subsequently be repaired with an insertion operator; see lines 15–17 of Algorithm 6.2. After re-constructing the first echelon tours of S_{shake} using our Clarke-Wright Savings Algorithm (line 18), local search is applied to S_{shake}. For this purpose we applied the variable neighborhood descent (VND) method shown in Algorithm 6.3. In the VND phase, a set of local search operators are applied to S_{shake} in a predefined and fixed order. In this context, note that after each application of a shaking and/or a local search operator, the first echelon routes must be reconstructed since satellite demands

may have changed.

In a basic VNS, only improved solutions are accepted. However, in the context of problems with many unfeasible solutions, this may cause the algorithm to get stuck during the search process. Therefore, we have adopted the method introduced by [90] for the solution acceptance decisions (lines 20–29). Based on this method, while improved solutions are always accepted, non-improving solutions are accepted with a certain probability p_{accept}. At each iteration, function ClcAcceptanceProbability($f_{ext}(S_{cur}), f_{ext}(S_{local}), T$) calculates p_{accept} as follows:

$$p_{accept} = \frac{e^{(-(f_{ext}(S_{local}) - f_{ext}(S_{cur})))}}{T} \tag{6.51}$$

Here, $f_{ext}(S_{cur})$ and $f_{ext}(S_{local})$ are values of the extended fitness function of the current solution and of the solution after local search, respectively. Lastly, T refers to the actual temperature value. At the beginning, T is initialized to an initial temperature T_{init} which is decreased by t^- at each main iteration of VNS (see line 30). In this way, while a non-improving solution is more likely to be accepted early during the search, the probability of accepting non-improving solutions will decrease with a growing iteration number. However, in case no improved solution was found during $iter\_ni_{max}$ iterations, T is reset to T_{init} in order to enhance diversification.

Algorithm 6.2 VNS for the 2E-EVRP-TW

input: an initial solution S_{init}
S_{shake} : Solution obtained after shaking
S_{local} : Locally optimal solution after VND
S_{cur} : Current solution
S_{bsf} : Best-so-far solution
$iter\_ni$: The number of non-improving solutions
$iter\_ni_{max}$: The maximum iteration limit for non-improving solutions
Determine set of neighborhood structures for shaking $\{N_k^{shake} \mid k = 1, \ldots, k_{max}\}$ and local search $\{N_h^{local} \mid h = 1, \ldots, h_{max}\}$
$S_{cur}, S_{bsf} \leftarrow S_{init}$
while the computational time limit is not reached **do**
  Create π of the shaking neighborhoods N_k^{shake}
  Set $k \leftarrow 1$
  while $k \leq k_{max}$ **do**
    Apply shaking: Choose S_{shake} from $N_{\pi(k)}^{shake}(S_{cur})$, the $\pi(k)^{th}$ shaking neighborhood of S_{cur}
    if $N_{\pi(k)}^{shake}$ is a removal/destroy operator **then** Repair S_{shake} **end if**
    Re-construct first echelon tours using Clarke-Wright Savings Algorithm
    Apply local search: $S_{local} \leftarrow$ VND(S_{shake})
    $p_{accept} \leftarrow$ ClcAcceptanceProbability($f_{ext}(S_{cur}), f_{ext}(S_{local}), T$)
    $\rho \leftarrow$ rand()
    if $f_{ext}(S_{local}) < f_{ext}(S_{bsf})$ **then** $S_{bsf} \leftarrow S_{local}$ **end if**
    if $f_{ext}(S_{local}) < f_{ext}(S_{cur})$ or $\rho < p_{accept}$ **then**
      $S_{cur} \leftarrow S_{local}$, $k \leftarrow 1$
    else
      $k \leftarrow k + 1$, $iter\_ni \leftarrow iter\_ni + 1$
    end if
    Decrease T by t^-
    if $iter\_ni = iter\_ni_{max}$ **then**
      $T \leftarrow T_{init}$
      $iter\_ni \leftarrow 0$
    end if
  end while
end while

Both shaking and local search operators are only applied to the second echelon routes. After obtaining the second echelon tours, the first echelon tours can easily

be constructed by saving heuristics; see lines 15 and 19.

Algorithm 6.3 Variable Neighborhood Decent (VND)

1: **input:** S_{shake}
2: S_{VND} : Solution obtained after applying a local search operator
3: Set $h \leftarrow 1$
4: **while** $h \leq h_{max}$ **do**
5: Generate S_{VND} from the h^{th} local search neighborhood of S_{shake} (S_{VND} : $N_h^{local}(S_{shake})$)
6: **if** $f_{ext}(S_{VND}) < f_{ext}(S_{shake})$ **then**
7: $S_{shake} \leftarrow S_{VND}$
8: $h \leftarrow 1$
9: **else**
10: $h \leftarrow h + 1$
11: **end if**
12: **end while**
13: **output:** S_{shake}

The following four sections will provide a detailed description of standard shaking operators, removal/destroy operators, repair operators, and local search neighborhoods, respectively.

6.2.3.1 Standard Shaking Operators

Random cyclic exchange: This operator was originally introduced by [178]. It transfers a node sequence (consisting of customers and/or charging stations) from one route to another in a cyclic way. This operator is quite advantageous for many sequence-based combinatorial optimization problems as it enables the generation of a large variety of moves with a single operator.

The cyclic exchange operator we have applied takes two parameters as input: (1) the number of routes (ζ) to be involved in the cyclic move, and (2) the maximum number of nodes (θ_{max}) to be transferred from one route to another. First, the operator randomly selects ζ routes. Then, a random integer number θ from the interval [1,θ_{max}] is independently determined for each route involved in the cyclic move. This value refers to the route-specific length of the node sequence to be transferred. Finally, θ consecutive nodes are randomly selected from each route and transferred to the next route in the cyclic move. If ζ is greater than the total number of routes existing in a solution, then ζ is set to the total number of routes. Similarly, if θ is greater than the total number of nodes in a route, then θ is

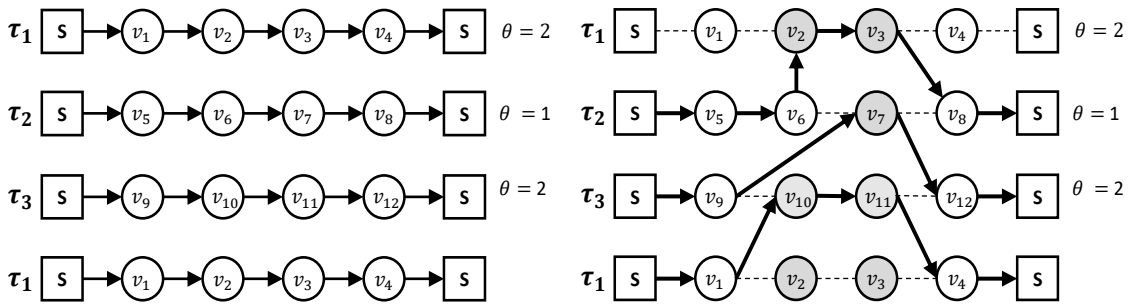


Fig. 6.2 An illustration of the cyclic exchange operator with $\zeta = 3$. Note that the route at the top and the route at the bottom are the same in order to show the cyclic nature of the move.

readjusted. The optimal values for both parameters are determined by parameter tuning (see Section 6.3.2). Figure 6.2 illustrates a cyclic exchange move with three routes.

Random sequence relocation: This operator selects a node sequence from one route and transfers it to another route. The origin and destination routes, the node sequence to be relocated, and the insertion position in the destination route are determined randomly. Parameter max_n limits the number of nodes to be transferred. The optimal value for max_n is determined by parameter tuning (Section 6.3.2).

6.2.3.2 Removal/Destroy Operators

One of the most critical aspects of a destroy operators is deciding on the number of nodes in the solution to be removed. Limiting the amount of destruction too much may cause a poor exploration performance of the algorithm. On the contrary, repairing a largely destroyed solution can be time-consuming and may result in a poor quality solution depending on the utilized repair procedure [148]. Therefore, we used a random removal rate between a lower and an upper bound to determine how many nodes (or routes) will be removed from the current solution. Well-working upper and lower bounds are decided via parameter tuning (Section 6.3.2) and fixed for each group of instances.

Random customer removal: First, a random number ρ is drawn from the interval $[rr1_{Lb}, rr1_{Ub}]$. This number is the fraction of customers to be removed from the solution, henceforth called the removal rate. Note that $rr1_{Lb}$ and $rr1_{Ub}$ are the lower and upper bounds for the removal rate. Finally, a randomly chosen number of max{1, $\lfloor \rho * n_c \rfloor$ } randomly chosen customers are removed from the current solution and added to a removal list L_r.

Random route removal: Similar to the random customer removal operator above, a random number ρ is drawn from the interval $[rr2_{Lb}, rr2_{Ub}]$. This number is the fraction of routes to be removed from the solution. Assume that solution S has n routes in the second echelon. After drawing number ρ, a number of max{1, $\lfloor \rho * n \rfloor$ } randomly chosen routes are removed from the current solution and all customers from these routes are added to a removal list L_r.

Close satellite: This operator closes a randomly chosen satellite and adds all the customers served through this satellite to a removal list L_r.

6.2.3.3 Repair Operators

A partially destroyed solution may either be repaired using an exact or a heuristic approach. Although exact approaches guarantee the optimal insertion of removed customers or routes, they are much more time consuming than heuristic approaches, especially when a rather large part of the solution is destroyed. Moreover, too much optimality in the repair operator may limit the diversification capabilities of the search. Therefore, we have applied greedy and best-insertion strategies for repairing partially destroyed solutions; see also [52, 158]. In the following, partially destroyed solutions are labelled S_d.

Greedy customer insertion: This operator reinserts each customer from L_r into the partially destroyed solution S_d according to the last-in-first-out (LIFO) principle. Henceforth, N_{S_d} denotes the set of nodes (customers and charging stations) that still form part of the partially destroyed solution S_d. Let $v \in L_r$ be the customer that is to be re-inserted into S_d. First, for each pair $i, j \in N_{S_d}$ such that i and j are consecutive nodes in one of the routes of S_d, the insertion cost δ_{vij} is calculated as follows:

$$\delta_{vij} = d2_{iv} + d2_{vj} - d2_{ij} \tag{6.52}$$

Then, customer v is inserted at the position with the lowest insertion cost. Suppose that the obtained route after insertion is infeasible in terms of vehicle capacity or time windows constraints. In this case, customer v is inserted at the next-cheapest position in terms of the insertion cost, and so on. If no feasible insertion position can be found, the customer is finally inserted at the initially best position, ignoring the infeasibility. In case of battery infeasibility, a charging station is added to the route. These procedures are applied until no customer remains in L_r.

Greedy customer insertion with noise: This operator is a special version of the greedy customer insertion operator described above. It utilizes the following modified cost function with a noise parameter for calculating the insertion cost of a customer v:

$$\delta_{vij}^{noise} = \delta_{vij} + d_{max} + \alpha + \beta \tag{6.53}$$

Here, d_{max} refers to the maximum distance between all nodes in N_{SRC}, and α refers to the noise parameter set to 0.1 ([52, 108–110]). Finally, β is a uniform random number generated independently for the calculation of each cost value from the interval $[-1, 1]$.

Best customer insertion: Instead of re-inserting customers from L_r in the LIFO order, this operator aims to find the best insertion position for all customers. Each time, the insertion costs of all remaining customers from L_r are calculated using Equation (6.52). Then, the customer with the best insertion cost is inserted into the best possible position. This operator is much more time consuming than the greedy customer insertion operator. It may, however, lead to better results. Infeasible insertions are handled in the same way as the greedy customer insertion operator.

Greedy CS insertion: In the case of battery infeasibility, this operator inserts a charging station with the lowest insertion cost at the point at which infeasibility occurs. Figure 6.3 illustrates the insertion of a charging station to a battery-infeasible route. Assuming that the electric vehicle runs out of battery before reaching node v_4, the operator first tries to insert a charging station $r^* := \text{argmax}\{d2_{3r} + d2_{r4} - d2_{34} \mid r \in N_R\}$ between nodes v_3 and v_4. In case the battery level is not high enough to reach the charging station that is to be inserted, a possible insertion is tried before node v_3, etc.

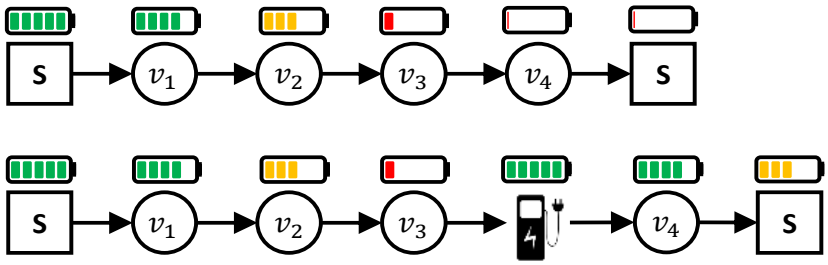


Fig. 6.3 An illustration of the charging station insertion operator. In the battery-infeasible route, the electric vehicle runs out of battery before reaching node v_4.

6.2.3.4 Local Search Neighborhoods

For the local search phase (that is, for the application within VND), the algorithm makes use of three inter-route operators (*exchange(1,1)*, *shift(1,0)*, and *swap*) and three intra-route operators (*relocation*, *two_opt*, and *CS_reinsertion*). In all these neighborhoods—except for *CS_reinsertion*—we use the first-improvement strategy, that is, a neighborhood exploration stops once the first improving solution is found. Infeasible moves are also allowed but they are penalized. Figure 6.4 graphically illustrates these local search neighborhoods.

The *exchange (1,1)* neighborhood considers all exchanges of each customer with every other customer not in the same route. The *shift (1,0)* neighborhood looks at all possibilities of removing a customer from its current route and inserting it at any position in the rest of the routes. Next, the *relocation* operator removes each customer from its current position and inserts it into another position in the same route. The *swap* neighborhood considers changing the positions of two selected nodes of the same route. The *two_opt* neighborhood considers all possibilities of selecting two non-consecutive nodes in the same route and reversing the node sequence between the two selected nodes. Note that there must be at least three nodes between the two selected nodes in order not to repeat moves already considered in the *swap* operator. The *CS_relocation* operator removes the current charging stations of a route and reinserts them in different positions in the same route in order to find the best positions for the charging stations. Unlike *CS_relocation*, the *CS_reinsertion* operator removes the current charging stations from a route. Instead of reinserting the removed ones, the greedy charging station insertion operator from the previous section is applied to repair the route. Thus, charging stations different from the removed ones may be inserted into the route.

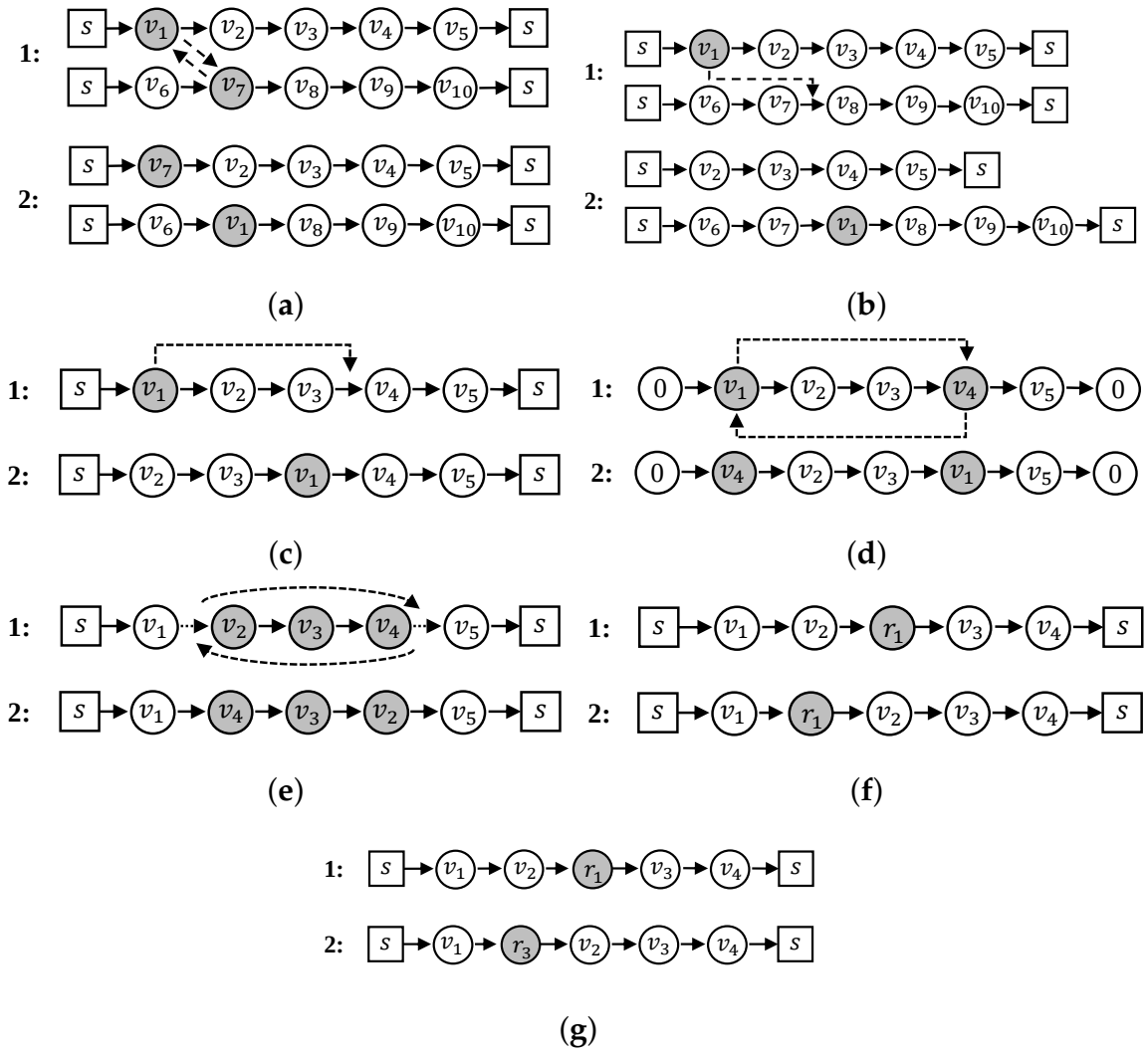


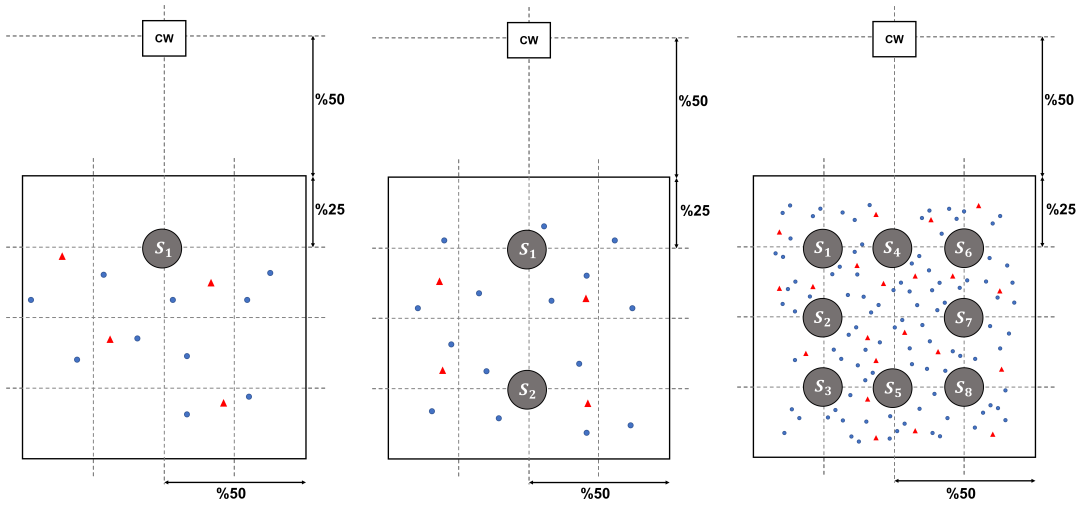
Fig. 6.4 An illustration of local search operators. (**a**) The exchange(1,1) operator, (**b**) The shift(1,0) operator, (**c**) The relocation operator, (**d**) The swap operator, (**e**) The two-opt operator, (**f**) CS_relocation operator, (**g**) CS_reinsertion operator.

6.3 Experimental Evaluation

In addition to our Clarke-Wright Savings Heuristic and VNS we also tried to solve all problem instances with the MILP solver CPLEX. All experiments were performed on a cluster of machines with Intel® Xeon® 5670 CPUs with 12 cores of 2.933 GHz and a minimum of 32 GB RAM. Note that CPLEX version 12.10 was used in one-threaded mode.

6.3.1 Generation of 2E-EVRP-TW Instances

Due to a lack of available benchmark sets for the 2E-EVRP-TW, we generated new problem-specific instance sets by extending the benchmark sets provided in [166]. These instances were proposed for the electric vehicle routing problem with time



(**a**) (**b**) (**c**)

Fig. 6.5 Illustration of the locations of the central warehouse and the satellite(s) in different cases. Blue dots refer to customers and red triangles are charging stations. (**a**) Small instance with 10 customers, (**b**) Small instance with 15 customers, (**c**) Large instance (100 customers).

windows and consist of 36 small and 56 large instances. Small instances are composed of 5, 10, or 15 customers with a varying number of charging stations (between 2 and 5), while the large ones include 100 customers and 21 charging stations. We have extended these instance sets following the methodology proposed in [84]. In particular, first, the number of satellites to be added to each instance was determined. Then, the locations of those satellites and the one of a single central warehouse was specified.

Concerning the number of satellites, we decided to use one single satellite in the case of small instances with at most 10 customers, two satellites in the case of 15 customers, and eight satellites for the large instances. Note that the customers of each instance are scattered over the intersections of a 100×100 grid. The location of the single central warehouse was determined for each instance to be outside this area, at $(50, 150)$. The single satellite in the case of instances with at most 10 customers was placed at $(50, 75)$, while the two satellites in the case of instances with 15 customers were places at $(50, 25)$ and $(50, 75)$. Finally, the eight satellites for all remaining instances were placed at $(25, 25)$, $(25, 50)$, $(25, 75)$, $(50, 25)$, $(50, 75)$, $(75, 25)$, $(75, 50)$ and $(75, 75)$. Figure 6.5 shows examples of all three cases.

In addition, we updated the customers' time windows by adding the distance between the location of the central warehouse in the original instance set and the new central warehouse as an offset value. Lastly, we modified the capacities of

the electric vehicles and large trucks, considering the fact that instances labeled with C2, R2, and RC2 are more capacity constrained as compared to those labeled C1, R1, and RC1. In particular, we have fixed the capacity ratio between large trucks and electric vehicles to $4/0.5$ for instances of the first type, and to $2/0.25$ for instances of the second type. All benchmark instances generated in this study and the executable of the proposed algorithm are available at `https://github.com/manilakbay/2E-EVRP-TW`, accessed on (01/05/2024) .

6.3.2 Parameter Tuning

In order to determine well-working parameter values for our algorithms we have utilized the scientific tuning software `irace` [124]. Tables 6.5 and 6.6 summarize the parameters that are subject to tuning for our Clarke-Wright savings heuristic and for the VNS together with the considered value domains.

Due to large differences in instance size, we decided to tune our VNS algorithm (including the parameters of the Clarke-Wright savings heuristic) separately for small and for large problem instances. In the first case (small instances), instances `C101_C10`, `R102_C10`, `RC102_C10`, `C103_C15`, `R102_C15` and `RC103_C15` were used for tuning. In the case of the large instances, instances `C101_21`, `C201_21`, `R101_21`, `R201_21`, `RC101_21` and `RC201_21` were used. For each of the two tuning runs, the budget of `irace` was fixed to 2000 algorithm runs. In the context of small instances, the computation time limit of each run was fixed to 150 CPU seconds, while it was fixed to 900 CPU seconds in the case of the large problem instances. Tables 6.5 and 6.6 show the obtained parameter value settings for the two cases. It is worth noting, for example, that well-working ranges for the removal rates (in the context of the removal/destroy operators) are considerably smaller in the case of the large instances when compared to those for small instances. One reason for this may be that repairing a largely destroyed solution is time consuming and may lead to a rather bad quality solution. On the other hand, a high removal rate may be considered as a perturbation mechanism that helps to escape from local minima in the context of small instances.

Table 6.5 VNS parameters, their domains, and values determined by `irace`.

Parameter	Description	Domain	Small Instances	Large Instances
$rr1_{Lb}$	Customer removal rate lower bound	$\{0.0, \ldots, 1.0\}$	0.1	0.3
$rr1_{Ub}$	Customer removal rate upper bound	$\{0.0, \ldots, 1.0\}$	0.7	0.4
$rr2_{Lb}$	Route removal rate lower bound	$\{0.0, \ldots, 1.0\}$	0.1	0.4
$rr2_{Ub}$	Route removal rate upper bound	$\{0.0, \ldots, 1.0\}$	0.6	0.4
p_{init}	Initial penalty value	$\{10, 15, 20\}$	15	20
p_{min}	Minimum penalty value	$\{0.5, 1, 3, 5\}$	5	0.5
p_{max}	Maximum penalty value	$\{25, 30, 35, 40\}$	40	35
p_{iter}	Iteration count parameter for penalty procedure	$\{1, 2, 3\}$	1	2
p^{+}	Augmentation parameter for penalty	$\{3, 5, 7, 9\}$	9	5
p^{-}	Reduction parameter for penalty	$\{1.0, \ldots, 2.0\}$	1.6	1.4
T^{init}	Initial temperature	$\{50, 100, 150, 200\}$	200	100
t^{-}	Update parameter for the temperature	$\{1.0, \ldots, 2.0\}$	1.3	2
$iter\_ni_{max}$	Max non-improving iterations	$\{100, \ldots, 10,000\}$	1000	100
ζ	Routes in cyclic exchange	$\{1, 2, 3, 4\}$	2	4
θ_{max}	Nodes in cyclic exchange	$\{1, 2, 3, 4\}$	4	2

Table 6.6 Parameters of the Clarke-Wright savings heuristic, their domains, and values determined by `irace`.

Param.	Description	Domain	Small Ins.	Large Ins.
λ	Route redesign parameter	$\{0.0, \ldots, 1.0\}$	1.3	1.1
μ	Asymmetry of information	$\{0.0, \ldots, 1.0\}$	0.3	0.1
γ	Assignment priority	$\{1.0, \ldots, 2.0\}$	0.9	0.6

6.3.3 Numerical Results

In the following we provide a detailed comparison of the following methods. First, we applied both CPLEX (version 12.10) and our Clarke-Wright Savings Heuristic to all problem instances. Hereby, CPLEX was given a time limit of 2 h of CPU time for each problem instance. Next we also applied two versions of VNS. The full version of VNS is henceforth denoted by VNS_{full}. In contrast, VNS_{red} is a reduced version of VNS that only utilizes classical inter-route and intra-route shaking operators. A comparison of these two variants is interesting, because it shows how much the destroy and repair operators add to the performance of VNS. Note that both versions of VNS were applied with a computation time limit of 150 CPU seconds in the case of small problem instances, and 900 CPU seconds for large problem instances. Moreover, both versions of VNS were applied 10 times to each problem instance.

Tables 6.7–6.9 show the numerical results for small problem instances with 5, 10, and 15 customers, respectively. The structure of these tables is as follows. Instance names are given in the first column, and the maximum number of vehicles in the first and second echelons are provided in the second and third columns, respectively. These numbers are only necessary for the application of CPLEX. After the first three table columns, there are four blocks of columns, presenting the results of our four approaches. The first three columns of each block (with headings 'n', 'm', and 'dist') are the same for all four approaches. Hereby, columns 'n' and 'm' provide the number vehicles utilized by the respective solutions in the first echelon and the second echelon, respectively. In the case of VNS_{full} and VNS_{red} these numbers refer to the best solution found within 10 independent runs. Column 'dist' provides the objective function values of the solutions generated by the four approaches. In the case of VNS_{full} and VNS_{red}, 'dist' shows the objective function value of the best solution found in 10 runs, while an additional column with the heading 'avg' provides the average objective function value of the best solutions of each of the 10 runs. Next, columns

with heading '$\overline{t(s)}$' show the computation time of CPLEX, our Clarke-Wright Savings Heuristic and the average computation times of VNS_{full} and VNS_{red} to find the best solutions in each run. Finally, column 'gap(%)' provides the gap (in percent) between the best solution and the best lower bound found by CPLEX. Note that, in the case where the gap value is zero, CPLEX has found an optimal solution.

Table 6.7 Computational results for small-sized instances with 5 customers.

Instances			CPLEX					Clarke-Wright Savings Heuristic				VNS$_{red}$					VNS$_{full}$				
Name	nv_1	nv_2	m	n	Dist	Gap(%)	$\overline{t(s)}$	m	n	Dist	$\overline{t(s)}$	m	n	Dist	Avg	$\overline{t(s)}$	m	n	Dist	Avg	$\overline{t(s)}$
C101_C5	1	2	1	2	**385.49**	0	1.67	1	3	442.19	0.00021	1	2	**385.49**	**385.49**	0.989	1	3	**385.49**	**385.49**	12.509
C103_C5	1	1	1	1	**341.33**	0	0.09	1	2	360.94	0.00011	1	1	**341.33**	**341.33**	0.006	1	1	**341.33**	**341.33**	0.502
C206_C5	1	1	1	1	**417.31**	0	5.97	1	3	480.9	0.00017	1	1	**417.31**	**417.31**	0.001	1	1	**417.31**	**417.31**	0.001
C208_C5	1	1	1	1	**381.91**	0	0.31	1	1	383.07	0.00011	1	1	**381.91**	**381.91**	0.001	1	1	**381.91**	**381.91**	0.001
R104_C5	1	2	1	2	**317.02**	0	1.61	1	1	317.78	0.00012	1	1	**317.02**	**317.02**	0.001	1	1	**317.02**	**317.02**	0.001
R105_C5	1	3	1	2	**453.74**	0	9.57	1	1	677.61	0.00014	1	2	**453.74**	495.16	0.000	1	2	**453.74**	**453.74**	29.693
R202_C5	1	1	1	1	**347.82**	0	0.21	1	1	348.29	0.00010	1	1	**347.82**	**347.82**	0.001	1	1	**347.82**	**347.82**	0.001
R203_C5	1	1	1	1	**371.31**	0	0.21	1	1	387.92	0.00016	1	1	386.48	386.48	0.001	1	1	**371.31**	**371.31**	7.203
RC105_C5	1	3	1	3	**432.64**	0	28.84	1	3	496.72	0.00015	1	2	**432.64**	435.77	0.404	1	2	**432.64**	437.34	21.488
RC108_C5	1	2	1	2	**460.89**	0	24.24	1	2	702.23	0.00016	1	2	**460.89**	**460.89**	0.008	1	2	**460.89**	**460.89**	3.281
RC204_C5	1	1	1	1	**332.86**	0	0.64	1	1	649.44	0.00015	1	1	**332.86**	**332.86**	0.018	1	1	**332.86**	**332.86**	0.015
RC208_C5	1	1	1	1	**327.30**	0	0.37	1	1	331.77	0.00010	1	1	331.77	331.77	0.000	1	1	**327.30**	**327.30**	15.193
average			-	-	**380.80**	-	6.15	-	-	464.905	0.00014	-	-	382.44	386.15	0.119	-	-	**380.80**	381.19	7.491

Table 6.8 Computational results for small-sized instances with 10 customers.

Instances			CPLEX					Clarke-Wright Savings Heuristic				VNS$_{red}$					VNS$_{full}$				
Name	nv_1	nv_2	**m**	**n**	**Dist**	**Gap(%)**	$\overline{t(s)}$	**m**	**n**	**Dist**	$\overline{t(s)}$	**m**	**n**	**Dist**	**Avg**	$\overline{t(s)}$	**m**	**n**	**Dist**	**Avg**	$\overline{t(s)}$
C101_C10	1	4	1	4	**538.31**	0	3021.74	1	5	568.85	0.00017	1	3	**538.31**	538.74	0.568	1	4	**538.31**	**538.31**	13.233
C104_C10	1	3	1	2	**484.32**	0	5309.78	1	4	663.74	0.00024	1	2	**484.32**	**484.32**	0.979	1	2	**484.32**	**484.32**	7.119
C202_C10	1	3	1	2	**425.53**	0	152.011	1	5	625.02	0.00021	1	2	**425.53**	**425.53**	0.030	1	2	**425.53**	**425.53**	2.018
C205_C10	1	3	1	3	**415.48**	0	157.97	1	3	435.37	0.00024	1	2	**415.48**	419.64	0.005	1	3	**415.48**	**415.48**	1.006
R102_C10	1	4	1	3	**505.50**	0	6150.84	1	4	648.65	0.00019	1	3	**505.50**	**505.50**	2.477	1	3	**505.50**	524.59	22.556
R103_C10	1	3	1	2	**436.08**	9.27	6318.99	1	3	613.76	0.00024	1	2	**436.08**	**436.08**	2.396	1	2	**436.08**	437.51	3.600
R201_C10	1	2	1	2	**460.71**	0	2686.75	1	4	730.95	0.00017	1	2	**460.71**	**460.71**	2.838	1	2	**460.71**	**460.71**	16.455
R203_C10	1	2	1	1	**436.51**	0	2192.71	1	1	437.75	0.00021	1	1	**436.51**	**436.51**	0.002	1	1	**436.51**	**436.51**	0.002
RC102_C10	1	5	1	4	**618.75**	16.85	7079.09	1	5	684.93	0.00026	1	4	**618.75**	**618.75**	41.620	1	4	**618.75**	**618.75**	11.025
RC108_C10	1	4	1	4	637.23	24.28	6739.84	1	4	721.2	0.00020	1	3	**559.88**	**559.88**	0.097	1	3	**559.88**	**559.88**	0.016
RC201_C10	1	4	1	3	**495.54**	0	969.86	1	4	634.13	0.00021	1	2	**495.54**	497.04	0.004	1	3	**495.54**	**495.54**	2.528
RC205_C10	1	3	1	3	**576.17**	0	462.62	1	3	702.34	0.00025	1	2	**576.17**	577.76	0.130	1	3	**576.17**	**576.17**	15.366
average			-	-	502.51	-	3436.85	-	-	622.22	0.00022	-	-	**496.06**	496.70	4.262	-	-	**496.06**	497.77	7.910

Table 6.9 Computational results for small-sized instances with 15 customers.

Instances			CPLEX					Clarke-Wright Savings Heuristic				VNS$_{red}$					VNS$_{full}$				
Name	nv_1	nv_2	**m**	**n**	**Dist**	**Gap(%)**	$\overline{t(s)}$	**m**	**n**	**Dist**	$\overline{t(s)}$	**m**	**n**	**Dist**	**Avg**	$\overline{t(s)}$	**m**	**n**	**Dist**	**Avg**	$\overline{t(s)}$
C103_C15	1	5	-	-	-	-	-	1	6	690.99	0.00036	1	3	**575.18**	582.02	4.925	1	4	**575.18**	**575.18**	0.623
C106_C15	1	4	1	3	**500.32**	13.37	7182.91	1	6	681.31	0.00022	1	3	516.60	524.10	2.100	1	3	516.60	516.60	1.027
C202_C15	1	5	1	4	714.81	32.23	7183.04	1	6	729.87	0.00034	1	4	617.24	618.66	29.966	1	3	**550.32**	**550.32**	12.454
C208_C15	1	3	1	2	**550.02**	15.56	7182.95	1	4	737.61	0.00023	1	2	619.73	619.73	6.976	1	2	**550.02**	**550.02**	22.000
R102_C15	1	7	-	-	-	-	-	1	9	950.25	0.00026	1	5	**716.56**	**716.56**	9.523	1	5	**716.56**	**716.56**	12.056
R105_C15	1	5	-	-	-	-	-	1	8	777.77	0.00038	1	4	**607.96**	**607.96**	30.850	1	4	**607.96**	**607.96**	25.605
R202_C15	1	3	1	3	719.61	35.36	7198.17	1	6	990.37	0.00043	1	2	**593.69**	597.79	8.033	1	3	**593.69**	**593.69**	60.988
R209_C15	1	3	1	2	**475.10**	10.09	7182.43	1	5	711.09	0.00024	1	2	**475.10**	519.46	0.712	1	1	**475.10**	482.30	77.386
RC103_C15	1	5	-	-	-	-	-	1	7	745.82	0.00035	1	4	**616.32**	622.10	1.565	1	5	**616.32**	**616.32**	1.803
RC108_C15	1	5	-	-	-	-	-	1	7	716.22	0.00026	1	5	**603.87**	**603.87**	0.214	1	5	**603.87**	615.11	0.033
RC202_C15	1	3	1	3	**552.70**	16.06	7182.65	1	5	697.24	0.00033	1	3	601.86	601.86	2.395	1	2	**552.70**	587.11	11.600
RC204_C15	1	3	1	2	**485.34**	13.93	7183.03	1	3	604.05	0.00035	1	2	551.56	551.56	0.670	1	2	**485.34**	**485.34**	15.566
average			-	-	-	-	-	-	-	752.72	0.00031	-	-	591.31	597.14	8.161	-	-	**570.30**	574.71	20.095

The following observations can be made: First, apart from instances R103_C10, RC102_C10, and RC108_C10, CPLEX was able to solve the mathematical model—within 2 h of CPU time—for all instances with five and ten customers to optimality. For two of the remaining three cases, CPLEX was able to provide feasible solutions of the same quality as VNS_{full} and VNS_{red}, without being able to prove optimality. However, for the instances with 15 customers, the performance of CPLEX heavily starts to degrade. The reason for the rapidly decreasing performance of CPLEX is that the size and complexity of the MILP model sharply increase based on the instance size. For instance, the average number of variables and constraints of the MILP model for the instances containing five customers is 986 and 2235, respectively. These values increase to 4008 and 9363 for the instances with 10 customers and to 13125 and 31482 for the instances with 15 customers. In this latter case, CPLEX could only provide valid solutions (without being able to prove optimality) in seven out of 12 instances. Nevertheless, for one instance (C106_15), CPLEX produced a better solution than both VNS variants.

Both VNS variants performed comparably on small problem instances with 5 and 10 customers. They were able to find solutions with the same objective function values as those of CPLEX. However, the performance of the two VNS variants starts to differ on the instances with 15 customers. While VNS_{full} provides results at least as good as CPLEX for all instances except for C106_C15, VNS_{red} only does so in seven out of 12 cases. Considering those instances for which CPLEX was able to obtain a solution, both VNS variants improved the solution quality of CPLEX, on average, by 0.55% (VNS_{red}) and 6.86% (VNS_{full}). In fact, VNS_{full} outperforms VNS_{red} both in terms of best-performance (column 'dist') and in terms of average-performance (column 'avg'). Note also that the running times of both VNS_{full} and VNS_{red} are in the order of seconds. While the superiority of both VNS_{full} and VNS_{red} over CPLEX in terms of CPU time is more significant for the instances with 10 and 15 customers, see Tables 6.8 and 6.9, only VNS_{red} provides better CPU times for the instances with 5 customers. Finally, note that the results of the Clarke-Wright Savings Heuristic are, in the context of these small problem instances, approx. 20% worse than the best results obtained. This is, however, achieved in very low computation times of a fraction of a second, which shows that our Clarke-Wright Savings Heuristic is a good candidate for producing the initial solutions of VNS.

Next, we analyze the results of the four approaches when applied to the large problem instances of our benchmark set. These results are shown in Tables 6.10–6.12. The structure of these tables is slightly different to the one of the previous result tables. First, results of CPLEX are not provided, because CPLEX

was not able to generate a single valid solution within 2 h of computation time. Second, the additional column with heading 'imp(%)' provides the improvement (in percent) of the VNS variants over the results of the Clarke-Wright Savings Heuristic. In addition to the tables we also provide *critical difference* (CD) plots [31] as a statistical tool for assisting the interpretation of the obtained results. First, the Friedman test was used to compare the three approaches simultaneously. As a consequence of the rejection of the hypothesis that the techniques perform equally, the corresponding pairwise comparisons were performed using the Nemenyi post hoc test [74]. The obtained results are graphically shown by means of the above-mentioned CD plots in Figure 6.6. Note that each considered algorithm variant is placed on the horizontal axis according to its average ranking for the considered subset of problem instances. The performances of those algorithm variants that are below the critical difference threshold (computed with a significance level of 0.05) are considered as statistically equivalent; see the horizontal bars joining the markers of the respective algorithm variants.

The following observations can be made. For the large clustered instances (Table 6.10) and large random instances (Table 6.11), VNS_{full} significantly outperforms VNS_{red}, both in terms of best-performance and average-performance. This is also shown in Figure 6.6b Figure 6.6c. However, the opposite is generally the case in the context of random-clustered instances, as shown in Figure 6.6d. This means that the removal/destroy operators have a rather negative impact on the performance of VNS in these cases. This is most probably due to their elevated computation time requirements. Nevertheless, Figure 6.6d also shows that this difference is not statistically significant. Moreover, the superiority of VNS_{full} over VNS_{red} is much more pronounced in the context of instances with a long scheduling horizon (`R2* C2*` and `RC2*`) compared to the instances with a short scheduling horizon (`R1* C1*` and `RC1*`); see Figure 6.6e and Figure 6.6f. Finally, when considering all large instances together, VNS_{full} significantly outperforms VNS_{red} (see also Figure 6.6a).

When comparing the algorithms in terms of the average computation times required to find the best solutions of a run, it can be seen that VNS_{red} was able to provide solutions in lower CPU times than VNS_{full}. We can infer that destroy and repair type operators help to produce better solutions; however, repairing a destroyed solution prolongs the computation time.

Table 6.10 Computational results for large-sized clustered instances.

Instances			Clarke-Wright Savings Heuristic				VNS_{red}						VNS_{full}					
Name	n_k	n_v	**m**	**n**	**Dist**	$\overline{t(s)}$	**m**	**n**	**Dist**	**Avg**	**Imp(%)**	$\overline{t(s)}$	**m**	**n**	**Dist**	**Avg**	**Imp(%)**	$\overline{t(s)}$
C101_C21	3	25	3	39	1941.16	0.005	3	20	1513.91	1562.77	19.49	499.70	3	20	**1494.18**	**1538.74**	20.73	579.98
C102_C21	3	28	3	33	1822.02	0.005	3	21	1501.66	1506.95	17.29	537.57	3	19	**1447.86**	**1487.11**	18.38	572.03
C103_C21	3	26	3	29	1702.89	0.005	3	20	1447.98	1463.34	14.07	509.37	3	19	**1399.25**	**1425.80**	16.27	656.63
C104_C21	3	31	3	24	1580.07	0.005	3	20	1435.04	1446.17	8.47	405.19	3	19	**1400.52**	**1439.76**	8.88	540.97
C105_C21	4	43	3	36	1877.85	0.005	3	20	1522.97	1541.60	17.91	359.00	3	20	**1493.69**	**1521.13**	19.00	466.13
C106_C21	4	37	3	35	1791.74	0.004	3	20	1474.74	1491.70	16.75	361.26	3	20	**1429.75**	**1476.85**	17.57	536.93
C107_C21	4	41	3	34	1838.83	0.005	3	20	1499.81	1513.36	17.70	400.85	3	20	**1485.7**	**1513.18**	17.71	582.44
C108_C21	3	33	3	29	1687.15	0.005	3	20	1461.25	**1476.72**	12.47	483.00	3	20	**1450.96**	1489.63	11.71	523.87
C109_C21	4	31	3	26	1619.19	0.005	3	20	1447.36	1456.90	10.02	326.59	3	20	**1409.97**	**1455.93**	10.08	673.76
C201_C21	3	20	2	35	1794.83	0.004	2	11	1251.62	1276.42	28.88	422.74	2	12	**1208.76**	**1233.87**	31.25	545.02
C202_C21	4	20	2	31	1672.52	0.005	2	12	1228.61	1260.08	24.66	532.72	2	12	**1187.87**	**1232.74**	26.29	703.08
C203_C21	3	19	2	27	1554.96	0.005	2	12	**1197.45**	1223.16	21.34	356.27	2	11	1201.4	**1216.36**	21.78	767.58
C204_C21	4	18	2	22	1411.07	0.005	2	12	1178.14	1191.92	15.53	464.31	2	11	**1161.07**	**1181.40**	16.28	577.96
C205_C21	4	20	2	21	1470.73	0.005	2	12	1226.48	1249.59	15.04	460.15	2	12	**1205.23**	**1223.94**	16.78	664.39
C206_C21	3	19	2	19	1399.11	0.005	2	12	1202.52	1222.94	12.59	519.78	2	11	**1182.63**	**1198.54**	14.34	556.03
C207_C21	3	19	2	20	1406.55	0.005	2	12	1195.1	1211.55	13.86	262.84	2	11	**1173.7**	**1188.92**	15.47	562.88
C208_C21	3	17	2	19	1393.28	0.005	2	12	1193.01	1221.40	12.34	429.15	2	11	**1169.69**	**1188.85**	14.67	652.98
average					1644.94	0.005			1351.63	1371.56	16.38	431.21			**1323.66**	**1353.69**	17.48	597.80

Table 6.11 Computational results for large-sized random instances.

Instances			Clarke-Wright Savings Heuristic				VNS_{red}						VNS_{full}					
Name	n_k	n_v	**m**	**n**	**Dist**	$\overline{t(s)}$	**m**	**n**	**Dist**	**avg**	**Imp(%)**	$\overline{t(s)}$	**m**	**n**	**Dist**	**Avg**	**Imp(%)**	$\overline{t(s)}$
R101_C21	4	34	4	46	2546.45	0.006	4	25	**2164.26**	**2187.39**	14.10	573.91	4	26	2179.75	2306.91	9.41	589.70
R102_C21	3	32	4	37	2365.85	0.006	3	21	**1840.45**	**1894.10**	19.94	583.85	3	24	1843.45	2025.31	14.39	300.08
R103_C21	3	23	3	32	1974.87	0.006	3	19	**1696.36**	**1754.36**	11.17	657.28	3	19	1729.91	1829.33	7.37	350.64
R104_C21	2	20	3	23	1784.94	0.006	2	17	1473.50	1641.42	8.04	770.40	2	17	**1470.20**	**1628.00**	8.79	535.30
R105_C21	3	25	3	39	2216.4	0.005	3	23	**1842.34**	**1898.80**	14.33	539.41	3	22	1909.13	1975.78	10.86	463.62
R106_C21	3	28	3	32	2055.43	0.006	3	20	1737.88	**1870.36**	9.00	345.95	3	21	**1723.88**	1887.34	8.18	153.16
R107_C21	2	23	2	30	1725.89	0.007	2	18	1518.95	1671.82	3.13	287.01	2	18	**1490.01**	**1670.00**	3.24	323.49
R108_C21	2	21	2	23	1603.11	0.006	2	18	1454.99	**1553.47**	3.10	322.44	2	18	**1449.13**	1569.62	2.09	184.32
R109_C21	2	24	3	29	1947.31	0.006	2	18	1547.52	1694.54	12.98	356.45	2	19	**1529.71**	**1683.81**	13.53	386.16
R110_C21	2	23	2	26	1650.24	0.006	2	17	**1451.04**	**1486.15**	9.94	597.54	2	17	1470.57	1513.50	8.29	660.80
R111_C21	2	24	3	26	1805.69	0.006	2	18	**1487.83**	**1572.49**	12.91	579.21	2	17	1522.49	1593.35	11.76	483.22
R112_C21	2	23	2	20	1460.99	0.006	2	20	1457.06	1457.06	0.27	0.00	2	17	**1413.86**	**1452.74**	0.56	64.47
R201_C21	1	14	2	34	1912.59	0.006	1	12	1238.92	1265.99	33.81	486.39	1	9	**1218.88**	**1252.17**	34.53	639.80
R202_C21	1	12	2	29	1760.69	0.006	1	9	1158.64	1170.28	33.53	626.39	1	9	**1135.84**	**1166.67**	33.74	564.04
R203_C21	1	14	2	22	1587.76	0.007	1	8	**1064.16**	**1093.38**	31.14	513.09	1	7	1067.89	1096.69	30.93	542.58
R204_C21	1	9	2	17	1408.95	0.006	1	7	**962.16**	994.44	29.42	534.08	1	6	965.62	**977.19**	30.64	591.03
R205_C21	1	14	1	27	1522.45	0.006	1	9	1136.47	1167.99	23.28	711.65	1	7	**1134.35**	**1155.14**	24.13	452.03
R206_C21	1	12	1	23	1445.78	0.006	1	8	1106.23	1137.05	21.35	789.19	1	7	**1092.55**	**1117.88**	22.68	512.81
R207_C21	1	13	1	17	1334.95	0.006	1	7	1034.45	1072.21	19.68	588.03	1	7	**1025.08**	**1055.93**	20.90	512.99
R208_C21	1	12	1	16	1232.96	0.006	1	7	991.25	1018.02	17.43	484.88	1	7	**970.31**	**996.20**	19.20	519.00
R209_C21	1	15	1	23	1400.02	0.006	1	8	1078.55	1106.82	20.94	597.86	1	7	**1078.00**	**1089.28**	22.20	597.67
R210_C21	1	12	1	19	1350.21	0.006	1	8	1059.31	1090.53	19.23	515.34	1	7	**1045.64**	**1068.65**	20.85	480.40
R211_C21	1	9	1	18	1291.64	0.006	1	7	1030.77	1053.79	18.41	524.60	1	6	**999.26**	**1034.15**	19.93	421.96
average					1712.40	0.006			1371.00	**1428.37**	16.83	521.08			**1368.07**	1441.11	16.44	449.10

Table 6.12 Computational results for large-sized random-clustered instances.

Instances			Clarke-Wright Savings Heuristic				VNS_{red}						VNS_{full}					
Name	n_k	n_v	**m**	**n**	**Dist**	$\overline{t(s)}$	**m**	**n**	**Dist**	**Avg**	**Imp(%)**	$\overline{t(s)}$	**m**	**n**	**Dist**	**Avg**	**Imp(%)**	$\overline{t(s)}$
RC101_C21	3	28	4	38	2467.62	0.004	4	23	2044.99	2274.23	7.84	294.29	3	22	**1907.52**	**2106.42**	14.64	605.26
RC102_C21	3	29	4	36	2385.73	0.005	4	22	2004.78	**2035.60**	14.68	516.43	3	21	**1834.97**	2047.79	14.16	397.12
RC103_C21	3	28	4	29	2189.24	0.004	3	20	1747.98	1933.49	11.68	393.23	3	20	**1728.17**	**1846.23**	15.67	470.22
RC104_C21	2	26	2	26	1710.42	0.004	2	19	**1644.36**	**1686.88**	1.38	322.00	2	19	1645.35	1688.65	1.27	372.96
RC105_C21	3	23	5	33	2482.3	0.005	3	20	**1789.64**	**1821.53**	26.62	471.32	3	20	1802.85	1936.87	21.97	506.77
RC106_C21	3	23	3	33	2142.63	0.005	3	20	1760.23	**1797.62**	16.10	584.16	3	19	**1750.61**	1807.42	15.64	450.61
RC107_C21	3	24	3	28	1901.16	0.004	3	19	1687.90	**1713.75**	9.86	493.35	3	19	**1686.76**	1719.83	9.54	681.21
RC108_C21	3	25	2	26	1737.71	0.005	3	19	1672.75	1676.55	3.52	440.79	3	18	**1622.76**	**1655.21**	4.75	760.71
RC201_C21	1	15	1	35	1809.28	0.004	1	14	**1313.01**	**1341.92**	25.83	682.08	1	11	1318.73	1358.75	24.90	282.60
RC202_C21	1	13	1	30	1636.91	0.005	1	12	**1218.40**	1246.29	23.86	691.50	1	10	1200.59	**1230.97**	24.80	531.87
RC203_C21	1	11	1	22	1401.03	0.005	1	10	1119.62	1140.74	18.58	589.31	1	8	**1103.43**	**1138.82**	18.72	624.78
RC204_C21	1	14	1	16	1267.87	0.004	1	9	1045.72	1077.93	14.98	462.65	1	8	**1040.09**	**1054.96**	16.79	470.14
RC205_C21	1	17	1	25	1553.79	0.004	1	11	1223.37	1253.27	19.34	368.42	1	9	**1217.43**	**1245.16**	19.86	356.82
RC206_C21	1	16	1	25	1536.28	0.004	1	10	1216.70	1235.36	19.59	495.64	1	9	**1193.11**	**1216.17**	20.84	610.81
RC207_C21	1	12	1	21	1424.02	0.004	1	9	1116.30	**1133.88**	20.38	532.73	1	8	**1106.60**	1146.08	19.52	442.03
RC208_C21	1	14	1	14	1253.98	0.005	1	9	**1038.25**	1081.38	13.76	535.70	1	8	1049.42	**1067.86**	14.84	516.79
average					1806.25	0.004			1477.75	1528.15	15.50	492.10			**1450.52**	**1516.70**	16.12	505.04

Finally, VNS_{red} and VNS_{full} produced comparable results for small problem instances in terms of the number of utilized vehicles. In contrast, the increased effectiveness of VNS_{full} is shown in the context of large problem instances. Even though making use of a lower number of vehicles usually means that a better solution is obtained, note that a smaller fleet size does not always guarantee a better solution. For some of the instances (i.e., `C103_C15`, `R202_C15`), even though VNS_{red} provides solutions with a lower fleet size than VNS_{full}, the solutions of VNS_{full} are better. The reason for this is that the objective function only minimizes the traveled distance.

It is also worth noting that the average improvement rate with respect to the solutions of the Clarke-Wright savings heuristic for large clustered problem instances is lower than in the context of the random and random-clustered instances. One reason for this is possibly the assignment of each customer to the nearest satellite in the initial solution construction phase, which provides most probably a better customer-satellite assignment than in the context of random instances.

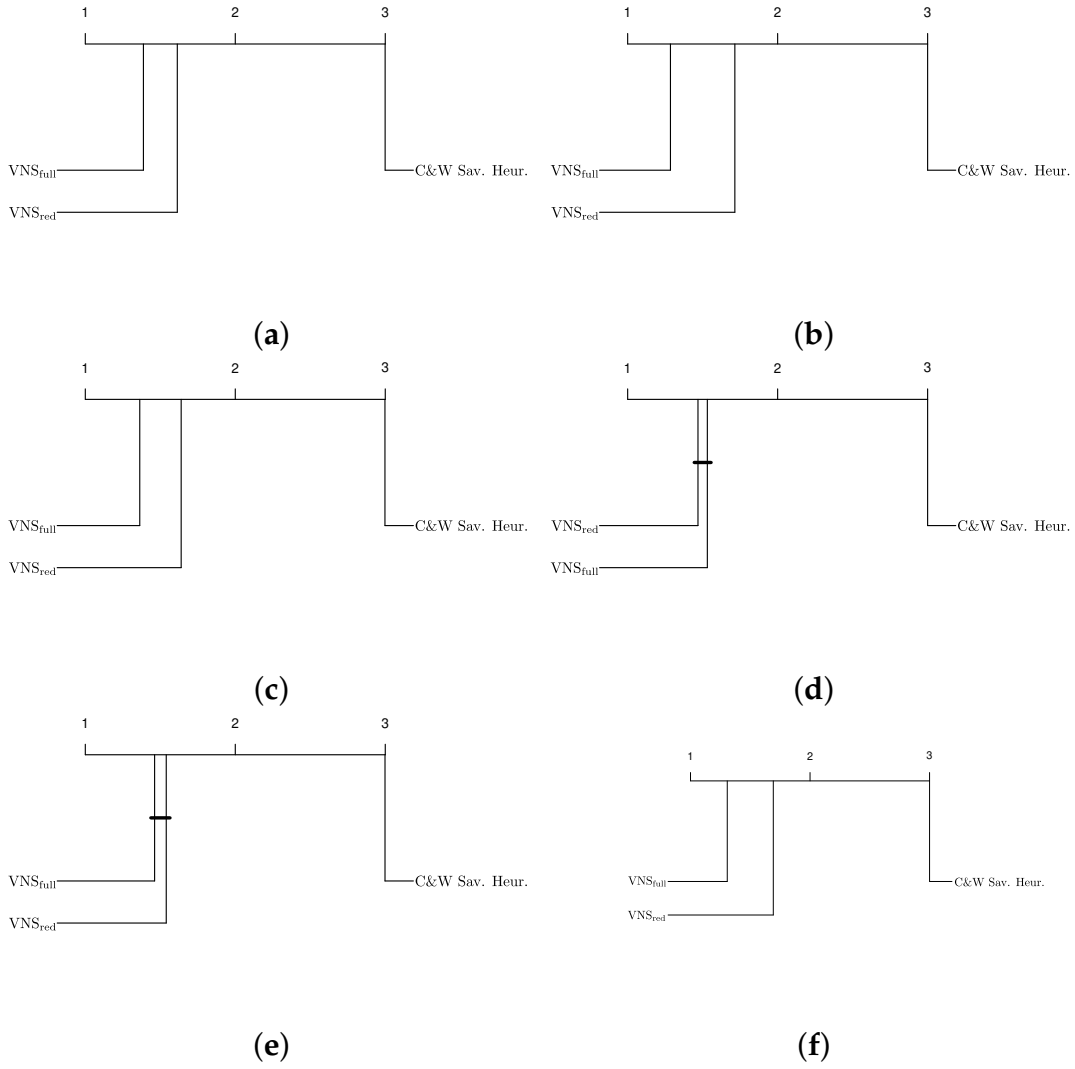


Fig. 6.6 Critical difference plots concerning the results for large instances. The graphic in (**a**) considers all large instances, while the other graphics consider subsets of the set of large instances. (**a**) All large instances; (**b**) clustered instances; (**c**) random instances; (**d**) random-clustered instances; (**e**) instances `R1*`; `C1*` and `RC1*`; (**f**) instances `R2*`, `C2*` and `RC2*`.

6.4 Conclusions

This chapter presented the two-echelon electric vehicle routing problem with time windows as a valuable concept for sustainable city logistics. A three-index node-based mixed-integer programming model was developed and solved using CPLEX for small instances. In addition, we proposed a variable neighborhood search metaheuristic making use of a wide range of classical and large neighborhood search operators. Moreover, our algorithm allows visiting unfeasible solutions, which is achieved by means of an extended objective

function for the evaluation of both feasible and unfeasible solutions. The local search step of our variable neighborhood search approach uses a variable neighborhood descent algorithm. Experimental tests were performed using new problem-specific instance sets generated based on available data sets from the literature. While CPLEX was able to solve the proposed mathematical model only for small problem instances with 5 and 10 customers, it started to struggle deriving even feasible solutions for larger instances. Our variable neighborhood search approach was able to find optimal or near-optimal solutions faster than CPLEX for all small problem instances. Moreover, numerical results showed that destroy-and-repair-type operators generally increased the algorithm's performance.

Continuing from the advances made in this chapter, several opportunities for further exploration remain. A promising direction for future work is the application of the Adapt-CMSA algorithm to the 2E-EVRP-TW comparing the performance of Adapt-CMSA with the variable neighborhood search approach.

CHAPTER 7

THE ELECTRIC VEHICLE ROUTING PROBLEM WITH ROAD JUNCTIONS AND ROAD TYPES: AN ANT COLONY OPTIMIZATION APPROACH

7.1 Introduction

This chapter introduces an enhanced EVRP model termed *EVRP with Road Junctions and Road Types* (EVRP-RJ-RT) that captures real-world complexities, such as road junctions and road type constraints. Additionally, our model utilizes an energy consumption-based objective function, offering a more realistic portrayal of the challenges electric vehicles face during logistics distribution. We have also expanded on existing problem instances from previous studies, tailoring them to our specific problem.

To handle the complexity of our advanced EVRP model, we have employed a construction heuristic which is a variant of the well-known Clark-Wright Savings algorithm. Additionally, we have implemented a more advanced metaheuristic—Ant Colony Optimization—on the basis of the well-known MAX-MIN Ant system [174]. The results presented in this chapter were previously published in the proceedings of GECCO 2024 – The Genetic and Evolutionary Computation Conference (`https://doi.org/10.1145/3638529.3653997`). Furthermore, it's worth mentioning that the initial version of the problem definition was developed in the Master's thesis of Michela Saliba, co-directed by the author of this thesis.

7.2 Problem Description

This section presents a detailed technical description of the new EVRP-RJ-RT problem. Our model shifts the focus from traditional distance-based objectives to an energy minimization framework, aligning with the real-world demands of electric vehicle logistics. In the following sections, we will outline the energy consumption calculations and the mathematical formulation of the problem in

detail.

7.2.1 Calculation of the Energy Consumption

The energy consumption of electric vehicles (EVs) is a critical factor for routing decisions. Our methodology for calculating energy consumption is adapted from the work presented in [19]. We begin by determining the total tractive power demand (PT) of an EV, measured in watts (W), using the following equation [17]:

$$\begin{aligned} PT =& M \cdot a \cdot v + M \cdot g \cdot v \cdot \sin(\theta) \\ &+ 0.5 \cdot C_d \cdot A_f \cdot \rho \cdot v^3 + M \cdot g \cdot C_r \cdot \cos(\theta) \cdot v \end{aligned} \tag{7.1}$$

where M represents the total mass of the vehicle, combining both the curb weight (w) and the weight of the cargo (m), in kilograms (kg). The variable v denotes the vehicle's speed in meters per second (m/sec), a is the acceleration (m/sec^2), g stands for the gravitational constant (9.81 m/sec^2), and θ is the road grade angle. The constants include the vehicle's frontal surface area (A_f in m^2), air density (ρ in kg/m^3), rolling resistance coefficient (C_r), and drag coefficient (C_d).

We assume all parameters in Eq. 7.1, except for speed and vehicle load, to be constant. However, speed and load may vary from one edge, resp. road segment, to another. For a vehicle traveling at an average speed of v_{ij} on a road segment between nodes i and j with distance d_{ij}, and with a cargo load m_j upon arrival at node j, the total mechanical energy required, E_{ij}, can be approximated as follows:

$$E_{ij} \approx PT \cdot \frac{d_{ij}}{v_{ij}} \approx [\alpha_{ij} \cdot (w + m_j) + \beta \cdot v_{ij}^2] \cdot d_{ij} \tag{7.2}$$

where the coefficient $\alpha_{ij} = a + g \cdot \sin(\theta_{ij}) + g \cdot C_r \cdot \cos(\theta_{ij})$ reflects the road segment specific resistance factor set to 0.0981, while $\beta = 0.5 \cdot C_d \cdot A \cdot \rho$, encapsulates the vehicle-specific aerodynamic and rolling resistance factor, which is a vehicle-related constant set to 2.11 [195]. This approximation provides the energy requirement in joules (J), which can then be converted into kilowatt-hours (kWh).

In this chapter, we relate the battery capacity stated for each problem instance directly with the mechanical energy, simplifying our calculations by not distinguishing between mechanical and battery electric energy. Thus, E_{ij} is directly used to calculate the energy consumption and to formulate the objective function.

7.2.2 Problem Formulation

The EVRP-RJ-RT involves a set of n customers indexed by $V = \{1, \ldots, n\}$, alongside a set of charging stations F, and road junctions J. To accommodate multiple visits, we introduce sets of dummy nodes for charging stations, $F^{'}$, and road junctions, $J^{'}$. We use indexes 0 and $n+1$ to denote a single depot, which acts as the starting and destination points, respectively. In case an index set is sub-indexed by 0, $n+1$, or both, the respective instances of the depot are added to the set. If, for example, $V^{'}$ is sub-indexed with 0 (that is, $V_0^{'}$), this means that 0 is added to $V^{'}$.

Using the sets and notations defined above, the problem is defined on a directed graph $G(V_{0,n+1}^{'}, A)$ which is not fully connected. Hereby, $V^{'} = V \cup F^{'} \cup J^{'}$ includes all customers, dummy charging stations, and dummy road junctions. The set of arcs is defined as

$$A = \{(i,j) \mid i,j \in V_{0,n+1}^{'}, i \neq j \text{ and } (i,j) \text{ is an existing road segment}\}. \tag{7.3}$$

Moreover, each arc has an associated distance d_{ij} and a road type implying specific speed limits, where v_{ij}^{ub} is the speed upper bound and v_{ij}^{lb} is the speed lower bound.

To more conveniently model the connectivity within the network, we introduce the set $\mathcal{N}(i)$, which represents the neighborhood of a given node i. This neighborhood consists of all nodes that are directly connected to node i. The set is formally defined as follows:

$$\mathcal{N}(i) = \{j \in V \mid (i,j) \in A\} \tag{7.4}$$

This means that a node j is in the neighborhood $\mathcal{N}(i)$ of node i if and only if there exists a direct connection between i and j in the network, as represented by the arc (i,j) in set A.

To address specific subsets of the network, we further define subscripted versions of $\mathcal{N}(i)$ based on predefined node sets, such as $V_{0,N+1}^{'}$. For example:

$$\mathcal{N}_{V_{0,N+1}^{'}}(i) = \{j \in V_{0,N+1}^{'} \mid (i,j) \in A\} \tag{7.5}$$

This definition signifies that $\mathcal{N}_{V_{0,N+1}^{'}}(i)$ includes those nodes from $V_{0,N+1}^{'}$ to which node i has a direct connection. Thus, $j \in \mathcal{N}_{V_{0,N+1}^{'}}(i)$ if and only if (i,j) is an element of set A, indicating a direct arc between nodes i and j.

A fleet of electric vehicles with identical loading capacity C and battery capacity Q is based at the depot. Customer demands q_i are met upon vehicle arrival, with a service time s_i. Furthermore, when the vehicle stops at a charging

station, its battery is charged with a constant charging rate of $g > 0$ up to full battery level. Note that the completion time of each vehicle tour is restricted by a predefined time limit T_{max}.

Our model incorporates the following decision variables: x_{ij}, a binary variable indicating the inclusion of arc (i, j) in the solution; v_{ij}, representing the vehicle's travel speed between nodes i and j; m_j, denoting the cargo load and y_j is the remaining battery level upon arrival at node j; f_j, specifying the vehicle's arrival time at node j; and E_{ij}, reflecting the energy expenditure between nodes i and j. Additionally, t_{ij} is introduced as the time required to travel from node i to node j, calculated as $t_{ij} = \frac{d_{ij}}{v_{ij}}$. Due to the nonlinear relationship among these variables, particularly in the context of energy consumption calculations, our model is formulated as a Mixed Integer Nonlinear Programming (MINLP) model, as follows:

$$\textbf{Min} \sum_{i \in V_0^{'}, j \in \mathcal{N}_{V_{n+1}^{'}}(i)} E_{ij} x_{ij} + \sum_{j \in \mathcal{N}_{V_{n+1}^{'}}(0)} x_{0j}\, K \tag{7.6}$$

$$\sum_{j \in \mathcal{N}_{V_{n+1}^{'}}(i)} x_{ij} = 1 \qquad \forall i \in V \tag{7.7}$$

$$\sum_{j \in \mathcal{N}_{V_{n+1}^{'}}(i)} x_{ij} \leq 1 \qquad \forall i \in F^{'} \cup J^{'} \tag{7.8}$$

$$\sum_{i \in \mathcal{N}_{V_0^{'}}(j)} x_{ij} - \sum_{i \in \mathcal{N}_{V_{n+1}^{'}}(j)} x_{ji} = 0 \qquad \forall j \in V^{'} \tag{7.9}$$

$$0 \leq f_0 \leq T_{max} \tag{7.10}$$

$$f_i + (t_{ij} + s_i) x_{ij} - T_{max}(1 - x_{ij}) \leq f_j \qquad \forall i \in V_0^{'}, j \in \mathcal{N}_{V_{n+1}^{'}}(i) \tag{7.11}$$

$$f_i + t_{ij} x_{ij} + g(Q - y_i) - (T_{max} + gQ)(1 - x_{ij}) \leq f_j \qquad \forall i \in F^{'}, j \in \mathcal{N}_{V_{n+1}^{'}}(i) \tag{7.12}$$

$$t_{0j} \leq f_j \leq T_{max} - (t_{0j} + s_j) \qquad \forall j \in V^{'} \tag{7.13}$$

$$v_{ij}^{lb} \leq v_{ij} \leq v_{ij}^{ub} \qquad \forall i \in V_0^{'}, j \in \mathcal{N}_{V_{n+1}^{'}}(i) \tag{7.14}$$

$$0 \leq m_0 \leq C \tag{7.15}$$

$$0 \leq m_j \leq m_i - q_i x_{ij} + C(1 - x_{ij}) \qquad \forall i \in V_0^{'}, j \in \mathcal{N}_{V_{n+1}^{'}}(i) \tag{7.16}$$

$$0 \leq y_j \leq y_i - E_{ij} + Q(1 - x_{ij}) \qquad \forall i \in V, j \in \mathcal{N}_{V_{n+1}^{'}}(i) \tag{7.17}$$

$$0 \leq y_j \leq Q - E_{ij} + Q(1 - x_{ij}) \qquad \forall i \in F_0^{'}, j \in \mathcal{N}_{V_{n+1}^{'}}(i) \tag{7.18}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V_0^{'}, j \in \mathcal{N}_{V_{n+1}^{'}}(i) \tag{7.19}$$

The problem addressed in this study gives priority to solutions that utilize fewer vehicles, even if this may lead to higher energy consumption. To reflect this preference in the objective function, we introduce an additional cost factor $K > 0$ for each vehicle used. Practically, the number of vehicles in a solution is represented by the sum of variables on outgoing arcs from the depot (node 0) that have a value of 1. Therefore, the objective is to minimize the function described in (Eq. 7.6). Constraints (7.7) and (7.8) manage the network's connectivity, ensuring that each customer is visited exactly once and that charging stations and road junctions are visited only as needed. Constraints (7.9) balance the flow by equating the number of incoming arcs to outgoing arcs at each node. Constraints (7.10-7.12) determine the arrival times at each node, taking into account service times at customer locations, battery charging durations, and travel times. Constraint (7.13) limits the tour compilation time. Constraint (7.14) forces the vehicle's speed to be between the lower and upper-speed limit of the traveled edge (resp. road segment). Constraints (7.15) and constraints (7.16) guarantee that customers' demands are satisfied. Finally, constraints (7.17)-(7.18) calculate the battery states.

7.3 The Solution Approach for the EVRP-RJ-RT

This section outlines the solution approach for the EVRP-RJ-RT, which is structured into three distinct phases: preprocessing, the main algorithm, and postprocessing. Initially, the preprocessing phase establishes the most energy-efficient paths between all pairs of nodes. Building on this, the main algorithms generate preliminary solutions, which are then refined during the postprocessing phase to possibly reduce energy spending. Before delving into the details of each phase, we first present how a solution is represented and evaluated within our approach.

7.3.1 Solution Representation & Evaluation

Any solution S produced by the algorithm is a collection of feasible tours, where each tour represents a complete route that starts and ends at the depot for a single vehicle.

To illustrate, consider a small problem instance represented by the vector $\mathbf{I}$, which includes the indices of five customers, three charging stations, and five road junctions. The depot is denoted by indices 0 and 6.

$$\mathbf{I} = \left(\overbrace{0}^{\text{depot}}, \overbrace{1,2,3,4,5}^{\text{customers}}, \overbrace{6}^{\text{depot}}, \overbrace{7,8,9}^{\text{charging stations}}, \overbrace{10,11,12,13,14}^{\text{road junctions}} \right)$$

Given the above instance, a solution S consisting of two tours T_1 and T_2, where $T_1 = \langle 0 \to 9 \to 12 \to 1 \to 13 \to 4 \to 10 \to 6 \rangle$ and $T_2 = \langle 0 \to 2 \to 8 \to 10 \to 3 \to 7 \to 14, \to 5 \to 6 \rangle$, is represented as follows:

$$S = \begin{cases} T_1 = \langle 0 \to 9 \to 12 \to 1 \to 13 \to 4 \to 10 \to 6 \rangle, \\ T_2 = \langle 0 \to 2 \to 8 \to 10 \to 3 \to 7 \to 14 \to 5 \to 6 \rangle \end{cases}$$

Our algorithm only considers feasible solutions, adhering to all constraints. The evaluation of these solutions makes use of two criteria, in a lexicographic order. The first priority is minimizing the number of electric vehicles utilized, which corresponds to the number of tours in the solution. The second objective is the minimization of the total energy consumption of all tours combined.

7.3.2 Preprocessing

In many conventional EVRPs from the existing literature, a problem instance consists of a fully connected graph. However, in the problem addressed in this study, the graph is generally not fully connected. Some node pairs might be directly connected by road segments, while for others a connecting path—in which intermediate nodes are road junctions—can be found. Note that charging stations and customers cannot serve as intermediate connection nodes. Each edge or road segment in the graph is associated with a specific road type, which imposes speed limitations (both minimum and maximum). Given that a vehicle's energy consumption is influenced by the travel speed and load and that the travel speed on a specific edge must adhere to its respective speed limitations, it is crucial to identify the most energy-efficient paths connecting each node pair.

For this purpose, we introduce a preprocessing step before the execution of the main algorithm. The main goal of this phase is to determine the most energy-efficient paths between pairs of nodes. To achieve this, we employ Dijkstra's algorithm, a well-established shortest-path algorithm [54]. Note that for the execution of the Dijkstra algorithm, the maximally allowed speed on each edge is considered. After this preprocessing phase, we obtain a complete set of paths $\mathcal{P}$ containing the most energy-efficient paths between all node pairs in the EVRP graph.

7.3.3 Postprocessing

After generating a valid solution within one of our main algorithms, there might remain a potential for reducing its energy consumption. This is because our algorithmic approaches—in an attempt to reduce the complexity of the problem—consider the maximum allowed speed on each traveled edge, resp. road segment. Note that this approach does not always yield the most energy-efficient routes.

Our postprocessing step aims to address this issue by considering the possibility of reducing travel speed to further reduce energy consumption. However, this poses a challenge: while lower speeds are more energy-efficient, they also extend the tour completion time, potentially violating time constraints.

The process begins by determining the slack time, which is the difference between the time limit and the actual time taken. A positive slack time indicates room for speed reduction. We then rank the route's edges in descending order based on their current speeds, targeting edges traversed with a higher speed first for potential reductions.

We systematically reduce the speed for each edge in a predefined step, recalculating the route's duration after each adjustment. This reduction continues until the slack time is exhausted, the edge's speed hits its minimum limit, or the route's duration reaches the time limit. This process ensures we remain within time constraints while aiming for the most energy-efficient route. This speed adjustment is applied to all routes of all solutions produced by the main algorithms.

7.3.4 A Construction Heuristic Based on the Clarke-Wright Savings Algorithm

Initially, a set of routes R is created, defined as $R = \{(p_{0i} \rightarrow p_{i(n+1)}) \mid i \in V, p_{0i}, p_{i(n+1)} \in \mathcal{P}\}$. For each customer i, this set concatenates the two most energy-efficient paths from $\mathcal{P}$: p_{0i}, the path from the depot (0) to customer i; and $p_{i(n+1)}$, the path from customer i back to the depot. Subsequently, a savings list $\mathcal{L}$ is generated for each pair of nodes $(i, j) \in V'$ under three conditions: (1) i and j are part of two different tours, (2) both i and j must be the starting or, respectively, the endpoints of their respective tours, and (3) the total load carried in the tours to which i and j belong must not exceed the vehicle capacity. Each entry in the list is associated with a savings value σ_{ij}, calculated using the following equation:

$$\sigma_{ij} := E_{0i} + E_{0j} - E_{ij} \quad , \tag{7.20}$$

Table 7.1 Cases for tour merging w.r.t. nodes i and j

Case	Tours	Merging Procedure	Result
1	$T_1 : \{0 \to i \to \ldots \to n+1\}$ $T_2 : \{0 \to j \to \ldots \to n+1\}$	Reverse T_1, $\mathsf{rev}(T_1)$ Concatenate with T_2	$T_m : \{0 \to \ldots \to p_{ij} \to \ldots \to n+1\}$
2	$T_1 : \{0 \to i \to \ldots \to n+1\}$ $T_2 : \{0 \to \ldots \to j \to n+1\}$	Reverse both T_1 and T_2 $\mathsf{rev}(T_1)$, $\mathsf{rev}(T_2)$	$T_m : \{0 \to \ldots \to p_{ij} \to \ldots \to n+1\}$
3	$T_1 : \{0 \to \ldots \to i \to n+1\}$ $T_2 : \{0 \to j \to \ldots \to n+1\}$	Concatenate T_1 and T_2	$T_m : \{0 \to \ldots \to p_{ij} \to \ldots \to n+1\}$
4	$T_1 : \{0 \to \ldots \to i \to n+1\}$ $T_2 : \{0 \to \ldots \to j \to n+1\}$	Reverse T_2, $\mathsf{rev}(T_2)$ Concatenate with T_1	$T_m : \{0 \to \ldots \to p_{ij} \to \ldots \to n+1\}$

where E_{0i}, E_{0j} and E_{ij} depend on the current vehicle load and it is assumed that the vehicle travels at the maximum allowed speed. Note that the entries in the savings list $\mathcal{L}$ are kept sorted in descending order based on their σ_{ij}. After creating the initial tours and the savings list $\mathcal{L}$, the algorithm executes the following steps until no more entry remains in $\mathcal{L}$:

1. The top most entry (i, j) is chosen from $\mathcal{L}$. Then, the tours corresponding to nodes i and j are merged considering the most energy-efficient path between i and j, that is, $p_{ij} \in \mathcal{P}$.

   The merging process is determined by one of the four possible cases shown in Table 7.1, depending on the positions of nodes i and j in the tour. It may be required to reverse one or both of the tours selected to ensure a direct connection from i to j. In such a case, the reversed tour T_1 is denoted by $\mathsf{rev}(T_1)$.

2. Following the merging process, the feasibility of the merged tour T_m is checked in terms of battery capacity and the time limit. Routes that violate time limit constraints are discarded, and the respective entry is removed from the savings list. On the other hand, battery-infeasible tours are attempted to be repaired by inserting a charging station into the tour. This procedure initially identifies the first node in the tour where the electric vehicle arrives with a depleted battery. Then, the charging station, which minimizes the increase in overall energy consumption, is inserted between this node and its predecessor. If the tour remains infeasible, then the same procedure is repeated for the previous arcs. Suppose infeasibility persists after multiple charging station insertions. In that case, the merged tour is discarded, its associated nodes are removed from the savings list, and the next node pair is selected from the top of $\mathcal{L}$.

3. Finally, the savings list $\mathcal{L}$ is updated as described above.

7.3.5 The Ant Colony Optimization Algorithm

The second algorithmic approach we employ to address the problem is Ant Colony Optimization (ACO) [56]. ACO is a swarm intelligence technique that mimics the foraging behavior of real ants to find optimal or near-optimal solutions to complex problems. In our implementation, we specifically utilize the MAX-MIN Ant System (MMAS) implemented in the hyper-cube framework [25]. The core principle of ACO revolves around constructing solutions probabilistically, guided by a pheromone model. This model represents the accumulated knowledge or experience of the ant colony. As ants traverse the solution space, they deposit and react to pheromone trails, which influence the solution construction and are updated based on the quality of the solutions found. Algorithm 7.1 shows the general framework of our implementation. The subsequent sections delve deeper into the specific components and mechanisms of the algorithm.

7.3.5.1 Solution Construction

The solution construction phase commences with an empty solution and a set of unvisited customers $\mathcal{U}$. First, a route is initialized with the depot as starting point. At each step, the route is augmented by inserting a new node j which is chosen from the candidate list of the current node i, which is defined as:

$$\mathcal{N}_i = \{j \mid j \in \mathcal{U} \cup F^{'} \cup \{n+1\}, \text{and } p_{ij} \neq \emptyset\}$$

.

A node j is added to the route only if it remains feasible in terms of loading, battery, and time limit constraints. If the insertion is not feasible, j is excluded from $\mathcal{N}_i$, prompting the selection of an alternative candidate. If $\mathcal{N}_i$ is exhausted or if $j = n+1$, we kick-start a new route. It is pivotal to note that when introducing the next node, we always opt for the most energy-efficient connection between nodes i and j, as explained in previous sections.

The probability of selecting a candidate node is calculated using the following formula:

$$P(j|i) = \frac{\tau_{ij}^{\alpha} \cdot \eta_j^{\beta}}{\sum_{j \in \mathcal{N}_i} \tau_{ij}^{\alpha} \cdot \eta_j^{\beta}} \tag{7.21}$$

Here, $\tau_{ij} \in \mathcal{T}$ denotes the pheromone value between the current node i and j. Moreover, η_j stands for heuristic information. The parameters α and β are

Algorithm 7.1 Ant Colony Optimization for the EVRP-RJ-RT

input: a problem instance $G(V'_{0,n+1}, A)$
$S^{bsf} := \emptyset$, $S^{rb} := \emptyset$, $cf := 0$, `bs_update` := FALSE
$\mathcal{P}$:=Preprocessing(G)
InitializePheromones($\mathcal{T}$)
while termination condition not met **do**
  $S^{ib} := \emptyset$
  for $k := 1, \ldots, n_a$ **do**
    $S^k :=$ ConstructSolution($\mathcal{T}, \mathcal{P}$)
    Postprocessing(S^k)
    if $f(S^k) < f(S^{ib})$ **then** $S^{ib} := S^k$
  end for
  if $f(S^{ib}) < f(S^{rb})$ **then** $S^{rb} := S^{ib}$
  if $f(S^{ib}) < f(S^{bsf})$ **then** $S^{bsf} := S^{ib}$
  UpdatePheromones(cf, `bs_update`, S^{ib},S^{rb},S^{bsf})
  $cf :=$ ComputeConvergenceFactor($\mathcal{T}$)
  if $cf > 0.999$ **then**
    if `bs_update` = TRUE **then**
      $S^{rb} := \emptyset$, and `bs_update` := FALSE
      InitializePheromones($\mathcal{T}$)
    else
      `bs_update` := TRUE
    end if
  end if
end while
output: S^{bsf}, the best solution found by the algorithm

used to weigh the relative importance of the pheromone value and heuristic information, respectively. Well-working values of α and β are determined by parameter tuning. The heuristic information is calculated using Eq. 7.22.

$$\eta_{ij} = \begin{cases} \dfrac{1}{E_{ij} \cdot (\mathrm{Y_i})} & \text{, if} j \in F' \\ \dfrac{1}{E_{ij}} \cdot \dfrac{1}{(\mathrm{Q} - \mathrm{Y_i})} & \text{, otherwise} \end{cases} \tag{7.22}$$

Again, remember that E_{ij} depends on the current vehicle load. Moreover, the maximum allowed speed is assumed. This piece-wise function calculates the desirability of the next nodes based on the node type and the battery level of the vehicle when departing from node i (denoted by Y_i). As a result, as the battery level decreases, it is more likely to choose a charging station [95].

The actual selection of a candidate node follows a two-step approach. First, a random number r from the interval [0,1] is generated. If r is less than or equal

Table 7.2 Weight settings for solutions based on *cf* and `bs_update`.

	`bs_update` = False				`bs_update`
	$cf < 0.4$	$cf \in [0.4, 0.6)$	$cf \in [0.6, 0.8)$	$cf \geq 0.8$	= True
κ_{ib}	1	2/3	1/3	0	0
κ_{rb}	0	1/3	2/3	1	0
κ_{bs}	0	0	0	0	1

to d_{rate}, the option with the highest probability is chosen. Otherwise, the roulette wheel selection method determines the next node.

The procedure continues until $\mathcal{U}$ is depleted. After creating a valid solution, the post-processing procedure described in Section 7.3.3 is applied to possibly reduce the energy consumption.

7.3.5.2 Pheromone Update

The pheromone update mechanism in our algorithm employs the same methodology as in any other MMAS algorithm applied in the hypercube framework. The first part of this mechanism updates each pheromone value $\tau_{ij} \in \mathcal{T}$ based on three key solutions kept during each algorithm's iteration:

- S^{ib}: best solution identified in the current iteration.
- S^{rb}: best solution since the last restart.
- S^{bsf}: best solution discovered since the algorithm's start.

In general, these three solutions play a crucial role in the pheromone update. At the start of the algorithm, S^{rb} and S^{bsf} are set to $\emptyset$. Moreover, we define $f(\emptyset) := \infty$. In addition, the pheromone levels τ_{ij} in $\mathcal{T}$ are initialized to 0.5 in function InitializePheromones($\mathcal{T}$).

The solutions S^{ib}, S^{rb}, and S^{bsf} are utilized as follows. Each solution is associated with a weight, namely κ_{ib}, κ_{rb}, and κ_{bs}. These weights are determined based on the convergence factor *cf* and a boolean variable `bs_update`. The table below presents the standard weight settings based on *cf* and `bs_update`; see also Algorithm 7.1.

With the solution weights established, each pheromone value is updated using the following equation:

$$\tau_i := \tau_i + \phi \cdot (\xi_i - \tau_i)$$

Here, ϕ is the learning rate, and the weighted average of solutions ξ_i is computed as:

$$\xi_i := \kappa_{ib} \cdot \Delta(S^{ib}, i, j) + \kappa_{rb} \cdot \Delta(S^{rb}, i, j) + \kappa_{bs} \cdot \Delta(S^{bsf}, i, j)$$

The function $\Delta(S, i, j)$ returns 1 if nodes i and j are consecutive nodes (in this order) in some tour of S, and 0 otherwise.

After the pheromone update, values exceeding $\tau_{\max} = 0.999$ are reset to $\tau_{\max}$, and those falling below $\tau_{\min} = 0.001$ are adjusted to $\tau_{\min}$. This adjustment ensures the algorithm avoids a state of complete convergence.

The convergence factor *cf* is computed consistently across all MMAS algorithms in the hypercube framework:

$$cf := 2 \left(\frac{\sum_{\tau \in \mathcal{T}} \max\{\tau_{\max} - \tau, \tau - \tau_{\min}\}}{|\mathcal{T}| \cdot (\tau_{\max} - \tau_{\min})} - 0.5 \right)$$

This formulation implies that *cf* is zero when all pheromones are at 0.5. Conversely, when all pheromones are either $\tau_{\min}$ or $\tau_{\max}$, *cf* is one. For all other scenarios, *cf* lies between 0 and 1.

7.4 Experimental Evaluation

All experiments were performed on a cluster of machines with Intel® Xeon® 5670 CPUs with 12 cores of 2.933 GHz and a minimum of 32 GB RAM.

7.4.1 Problem Instances

To test the proposed algorithms, we generated a set of problem instances based on the EVRP-TW instances introduced by Schneider et al. [166]. These instances span a range of scenarios with 5 to 100 customers and include a predetermined number of charging stations. The spatial distribution of customers is categorized by the prefixes “c” for clustered, “r” for random, and “rc” for a mix of both.

In the process of adapting these instances to the EVRP-RJ-RT, we preserved the original depot, customer locations, and charging stations. We then expanded the road network with additional nodes to symbolize road junctions. In particular, we chose the number of added road junctions to be equal to the number of original nodes (depot, customers, and charging stations). The coordinates for these new nodes were generated randomly within the bounds set by the extremities of the original node’s coordinates. To obtain road networks of different densities, we generated two distinct variations of each instance, characterized by the minimum number of neighboring nodes to which each node is connected (5 and 15). In the case of the 5-neighbors model, each node is linked to its five closest neighbors, ensuring that at least two of these are the nearest road junctions, with the rest

being chosen from the closest nodes. The connectivity information is stored as a matrix in the problem instance files. Additionally, this connection matrix provides information about the road type of each edge (1, 2, or 3), representing highways, main roads, and urban roads, respectively. These distinctions help simulate different speed constraints similar to real-world speed limits. The resulting instances are provided upon request.

To further align the instances more closely with real-world conditions, we implemented the following adjustment [146]:

- Customer demands were adjusted to fall between 50-450 kg.
- The vehicle's loading capacity was set at 3650 kg.
- The curb weight of the vehicle was defined as 6350 kg.
- The recharging rate was fixed to 0.0545 - indicating the hours required to charge 1 kWh of energy.
- The battery capacity was standardized at 110 kWh.
- T_{max} limited to 8 hours.

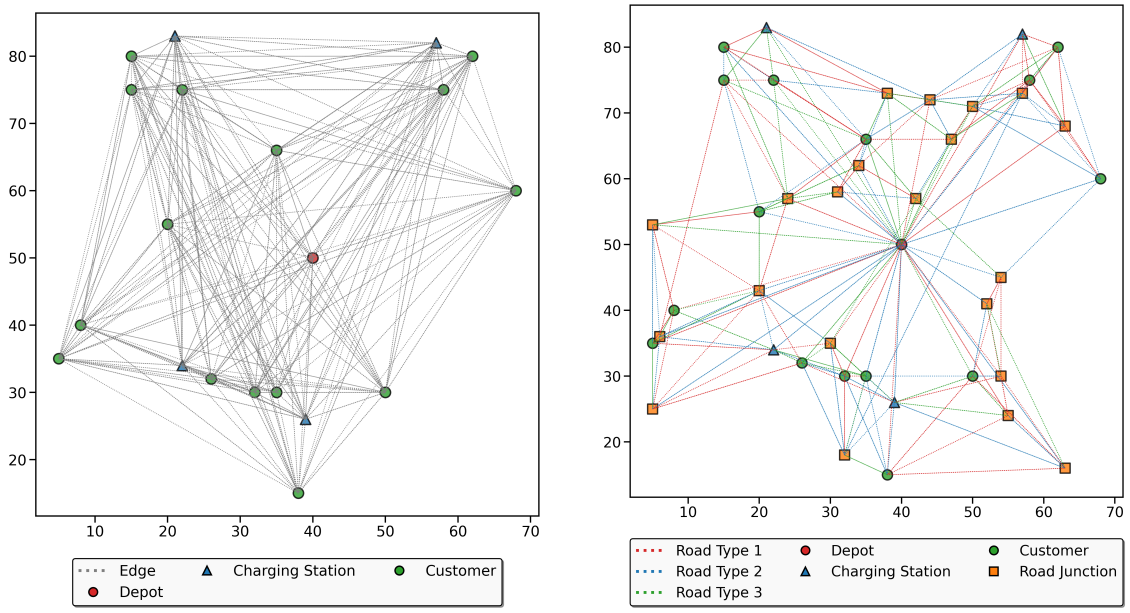
Figures 7.1a and 7.1b illustrate an example of both original and modified problem instances, respectively.

7.4.2 Parameter Tuning

We employed the scientific tuning software irace [124] to determine the optimal parameter values for ACO. The tuning process was executed using six instances, namely `r101_21_n15`, `r201_21_n15`, `r103_21_n10`, `rc201_21_n10`, `rc201_21_n15`, and `rc103_21_n5`. The budget of irace was set to 2500 algorithm runs, and each problem instance was subjected to a time limit of 2400 CPU seconds. The precision of irace was set to two decimal places for numerical parameters. Parameter tuning results are provided in Table 7.3.

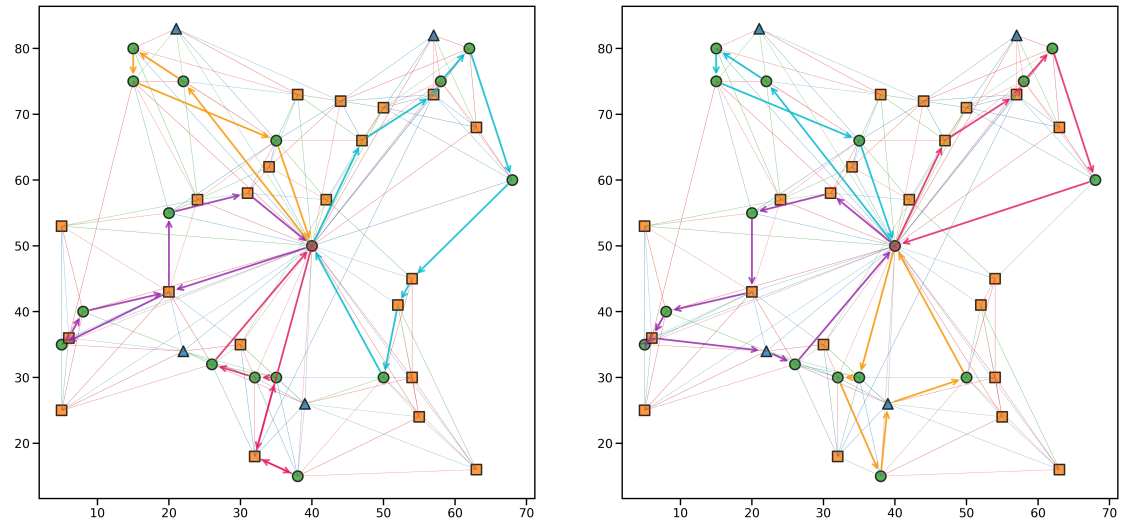
7.4.3 Computational Results

This section presents comparative results of the Clarke-Wright Savings algorithm and the ACO algorithm. As they are the most interesting ones, we only present results for large-sized EVRP-RJ-RT problem instances with 100 customers. Given its deterministic nature, the Clarke-Wright Savings algorithm was applied once to each instance. On the other hand, the ACO algorithm was run 10 times for



(a) EVRP instance

(b) Extended EVRP-RJ-RT instance



(c) Clarke-Wright solution

(d) ACO solution

Fig. 7.1 Overview of example instances and corresponding solutions. Panels (a) and (b) compare a base EVRP instance with a depot, fifteen customers, and four charging stations on a fully connected graph, to its extended version for EVRP-RJ-RT with twenty road junctions and a non-connected graph, highlighting road types with speed limits: highways (80-100 km/h), urban roads (55-70 km/h), and city streets (30-50 km/h). Panels (c) and (d) illustrate the solutions obtained from the Clarke-Wright Savings Heuristic and the ACO algorithm, respectively, with the former resulting in energy consumption of 112.36 kWh and the latter 99.97 kWh, showcasing the routes on the same map with four unique routes marked by different colored arrows.

each instance to account for solution variability. The computational time limit was set to 2400 CPU seconds for each execution of ACO.

Table 7.3 Parameters, their domains, and the chosen values as determined by irace.

Parameter	Domain	Value	Description
α	$[1, 3]$	1.66	Relative importance of the pheromone
β	$[1, 3]$	1.91	Relative importance of the heuristic information
d_{rate}	$[0.01, 0.99]$	0.62	Determinism rate
ϕ	$[0.01, 0.99]$	0.42	Learning rate
n_{a}	$\{2, 5, 10, 20, 50, 100, 200\}$	10	Number of generated solution

The numerical results are provided in Tables 7.4-7.6. Each table deals with a specific group of instances. More specifically, Table 7.4 covers instances with a neighborhood size of 5—that is, instances with sparser road networks—while Table 7.5 and Table 7.6 deal with instances generated with a neighborhood size of 10 and 15, respectively. The structure of these tables is straightforward. The left-most column lists the instance names. A subsequent multi-column, labeled "C&W," outlines the results obtained with the Clarke-Wright Savings algorithm, detailing the number of electric vehicles used, the energy consumed, and the computation time. Another multi-column (labeled "ACO") provides insights into the ACO algorithm's outcomes. Here, the 'ev' column specifies the number of electric vehicles in the best solution among 10 runs. The columns "best energy" and "avg. energy" show the energy consumption of the best solution and average energy consumption values across the 10 runs, respectively. "best imp. %" and "avg. imp. %" columns provide the percentage improvements of ACO over C&W, both concerning the best solution found and concerning the average over the best solutions from 10 runs. Finally, the "time" column reveals the average CPU time required to find the best solution in each ACO run. Figures 7.1c and 7.1d illustrate example solutions derived by the Clarke-Wright Savings Heuristic and ACO, respectively.

From our analysis, it is evident that ACO outperforms the Clarke-Wright Savings Heuristic not only concerning the best solution found over 10 runs but also concerning the averages of the energy consumption and the number of electric vehicles used over 10 runs. The performance gap is more pronounced in sparse road networks (Table 7.4) than in denser ones (Tables 7.5 and 7.6). The reason is that with an increased number of road junctions and, consequently, more routing options, Clarke-Wright tends to yield better solutions.

The boxplots presented in Figure 7.2 provide a detailed comparative analysis of the performance between the Clarke-Wright Savings Heuristic and ACO across various instance groups. These graphics again illustrate that the ACO algorithm outperforms the Clarke-Wright Savings Algorithm in terms of both best and

Table 7.4 Comparison of ACO and Clarke-Wright for sparse EVRP-RJ-RT instances.

Instance	C&W			ACO					
name	ev	best	time	ev	best	avg.	best imp. %	avg. imp. %	time
c101_21_n5	27	582.76	0.68	**26**	**524.16**	540.48	10.06	7.25	2099.28
c102_21_n5	**26**	627.36	1.38	**26**	**530.49**	543.85	15.44	13.31	2278.22
c103_21_n5	**26**	547.97	0.67	**26**	**488.95**	501.24	10.77	8.53	2074.05
c104_21_n5	27	567.22	0.73	**26**	**503.66**	520.89	11.21	8.17	2001.83
c201_21_n5	27	587.48	0.70	**26**	**546.44**	551.89	6.99	6.06	2205.59
c202_21_n5	**27**	592.33	0.79	**27**	**542.20**	549.66	8.46	7.20	2098.54
c204_21_n5	28	572.85	1.23	**26**	**544.87**	561.51	4.88	1.98	2070.88
r101_21_n5	**5**	331.16	0.69	**5**	**267.21**	282.63	19.31	14.65	2099.08
r102_21_n5	**5**	333.41	0.62	**5**	**278.43**	290.95	16.49	12.73	2169.35
r103_21_n5	**5**	287.32	0.57	**5**	**258.76**	268.60	9.94	6.52	2069.87
r104_21_n5	**5**	330.87	0.67	**5**	**247.07**	266.47	25.33	19.46	2277.53
r201_21_n5	**5**	301.41	0.65	**5**	**244.68**	254.87	18.82	15.44	2236.51
r203_21_n5	**5**	311.59	0.67	**5**	**244.39**	256.35	21.57	17.73	2293.71
r204_21_n5	**5**	312.35	0.61	**5**	**250.39**	259.02	19.84	17.07	1958.23
rc101_21_n5	**5**	312.55	0.58	**5**	**257.81**	276.20	17.51	11.63	2021.72
rc102_21_n5	**5**	329.64	0.67	**5**	**256.92**	275.82	22.06	16.33	1919.27
rc103_21_n5	**5**	293.90	0.58	**5**	**266.16**	276.97	9.44	5.76	1758.45
rc104_21_n5	6	328.90	0.43	**5**	**292.46**	308.37	11.08	6.24	2174.22
rc202_21_n5	6	318.44	0.61	**5**	**276.21**	286.28	13.26	10.10	2260.67
rc203_21_n5	6	296.86	0.42	**5**	**260.58**	280.19	12.22	5.62	1814.73
rc204_21_n5	**5**	338.59	0.60	**5**	**264.43**	275.64	21.90	18.59	2209.32
Average	12.43	405.00	0.69	**12.05**	**349.82**	363.23	14.60	10.97	2099.57

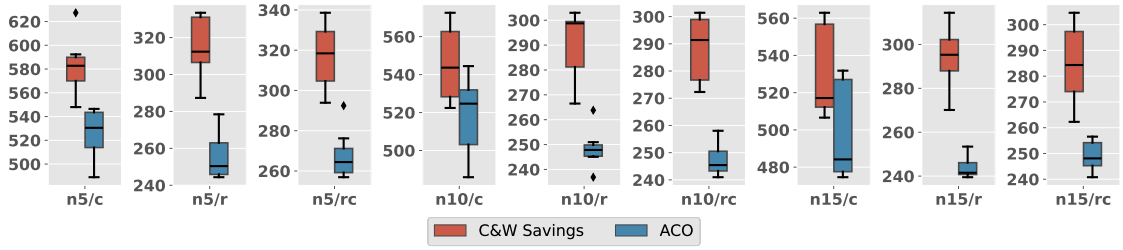
average energy consumption across all instance groups.

Table 7.5 Comparison of ACO and Clarke-Wright for medium-density EVRP-RJ-RT instances.

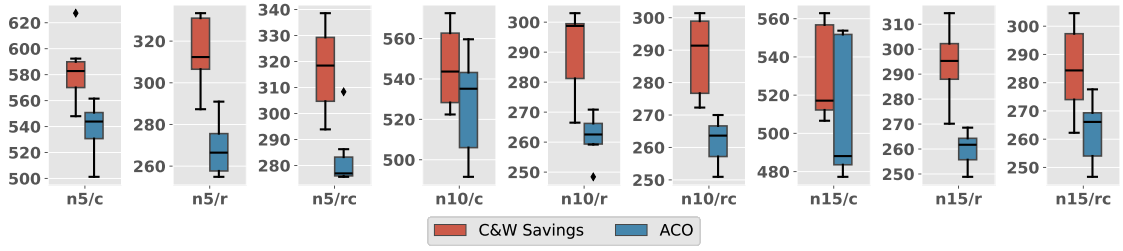
Instance	C&W			ACO					
name	ev	best	time	ev	best	avg.	best imp. %	avg. imp. %	time
c101_21_n10	**26**	522.42	0.59	26	**485.93**	491.54	6.98	5.91	2275.73
c102_21_n10	**26**	558.36	0.60	**26**	**514.19**	517.42	7.91	7.33	2250.59
c104_21_n10	27	529.46	0.58	**26**	**492.06**	494.58	7.06	6.59	2229.82
c201_21_n10	**26**	543.67	0.59	**26**	**531.03**	559.66	2.32	-2.94	2221.20
c202_21_n10	**26**	572.58	0.61	**26**	**544.47**	545.21	4.91	4.78	2231.86
c203_21_n10	**26**	527.16	0.62	**26**	**524.69**	535.16	0.47	-1.52	2150.51
c204_21_n10	**26**	567.04	0.65	**26**	**532.86**	541.20	6.03	4.56	2236.61
r101_21_n10	**5**	298.91	0.48	**5**	**263.80**	270.84	11.75	9.39	2215.93
r103_21_n10	**5**	279.91	0.52	**5**	**236.88**	248.43	15.37	11.25	2112.57
r104_21_n10	**5**	282.53	0.50	**5**	**248.72**	269.11	11.97	4.75	2068.15
r201_21_n10	**5**	298.75	0.51	**5**	**250.97**	262.55	15.99	12.12	2020.13
r202_21_n10	**5**	266.49	0.53	**5**	**247.83**	263.37	7.00	1.17	2118.18
r203_21_n10	**5**	303.00	0.51	**5**	**245.07**	259.54	19.12	14.34	2260.36
r204_21_n10	**5**	300.07	0.50	**5**	**245.59**	259.19	18.16	13.62	2225.41
rc102_21_n10	**5**	301.40	0.58	**5**	**258.04**	267.82	14.39	11.14	2114.35
rc103_21_n10	**5**	297.07	0.54	**5**	**245.43**	261.52	17.38	11.97	2082.76
rc104_21_n10	**5**	291.39	0.50	**5**	**247.61**	270.00	15.02	7.34	2179.50
rc201_21_n10	**5**	272.30	0.49	**5**	**244.94**	252.88	10.05	7.13	1996.98
rc202_21_n10	**5**	279.43	0.49	**5**	**253.44**	263.65	9.30	5.65	2064.93
rc203_21_n10	**5**	300.81	0.49	**5**	**240.99**	250.94	19.89	16.58	2233.38
rc204_21_n10	**5**	273.96	0.51	**5**	**241.63**	265.46	11.80	3.10	2182.55
Average	12.05	374.61	0.54	**12.00**	**337.91**	350.00	11.09	7.35	2165.31

Table 7.6 Comparison of ACO and Clarke-Wright for high-density EVRP-RJ-RT instances.

Instance	C&W			ACO					
name	ev	best	time	ev	best	avg.	best imp. %	avg. imp. %	time
c101_21_n15	**26**	517.12	0.55	**26**	**484.16**	488.11	6.37	5.61	2260.81
c102_21_n15	**26**	508.07	0.56	**26**	**480.21**	487.34	5.48	4.08	2141.38
c103_21_n15	**26**	516.42	0.55	**26**	**474.93**	477.25	8.03	7.58	2165.13
c104_21_n15	**26**	506.60	0.55	**26**	**474.61**	479.81	6.31	5.29	2245.11
c202_21_n15	**26**	553.27	0.55	**26**	**524.08**	550.07	5.28	0.58	2165.83
c203_21_n15	27	560.28	0.58	**26**	**530.08**	553.74	5.39	1.17	2108.68
c204_21_n15	**26**	562.90	0.59	**26**	**531.81**	553.47	5.52	1.68	2205.35
r101_21_n15	**5**	283.53	0.46	**5**	**241.53**	248.86	14.81	12.23	2099.71
r102_21_n15	**5**	295.29	0.46	**5**	**241.21**	261.71	18.31	11.37	2162.46
r103_21_n15	**5**	314.42	0.46	**5**	**244.26**	266.03	22.31	15.39	2211.48
r201_21_n15	**5**	270.14	0.47	**5**	**239.51**	260.76	11.34	3.47	2262.17
r202_21_n15	**5**	292.38	0.48	**5**	**253.44**	268.56	13.32	8.15	2200.11
r203_21_n15	**5**	306.57	0.46	**5**	**240.26**	250.67	21.63	18.23	2124.56
r204_21_n15	**5**	297.87	0.46	**5**	**247.99**	262.55	16.75	11.86	2285.87
rc101_21_n15	**5**	294.37	0.48	**5**	**256.53**	277.64	12.85	5.68	2266.00
rc102_21_n15	**5**	304.61	0.50	**5**	**254.52**	266.28	16.44	12.58	2141.76
rc104_21_n15	**5**	300.31	0.46	**5**	**253.86**	272.38	15.47	9.30	2023.89
rc201_21_n15	**5**	262.27	0.46	**5**	**244.16**	251.99	6.91	3.92	1815.69
rc202_21_n15	**5**	267.71	0.48	**5**	**240.79**	246.67	10.06	7.86	1972.90
rc203_21_n15	**5**	280.32	0.48	**5**	**246.35**	256.12	12.12	8.63	2099.69
rc204_21_n15	**5**	284.34	0.46	**5**	**248.09**	266.11	12.75	6.41	1972.19
Average	12.05	370.42	0.50	**12.00**	**331.07**	345.05	11.78	7.67	2139.56



(a) Best Energy Consumption



(b) Average Energy Consumption

Fig. 7.2 Comparative analysis of the Clarke-Wright Savings Heuristic and the ACO algorithms across various instance groups, such as n5/c, n5/r, and n5/rc. Here, "n5" denotes the number of neighbors of each node (road network density), while "c", "r", and "rc" indicate the spatial distribution of the nodes. For each instance group, paired boxplots show the distribution of both the best (a) and average (b) energy consumption results obtained by the two algorithms.

7.5 Conclusion

This chapter introduced an advanced extension of the Electric Vehicle Routing Problem (EVRP) which we labeled the EVRP with Road Junctions and Road Types (EVRP-RJ-RT). This problem variant moves the research on EVRPs closer to considering the complexities of real-world urban logistics. Our approach expands the conventional EVRP framework by incorporating road junctions and various road types with unique speed constraints, such as highways, urban roads, and city streets. This not only increases the model's complexity in comparison to existing EVRP variants but also makes it more realistic by reflecting the actual transportation networks of urban areas. A distinctive feature of our approach is the treatment of speed as a decision variable, which introduces an additional layer of complexity to the problem.

We have formulated the problem as a Mixed Integer Non-Linear Programming Model (MINLP) that utilizes an objective function and battery constraints that accurately reflect the energy consumption of electric vehicles, considering load, speed, and distance traveled. To address the computational challenges posed by this enhanced EVRP, we have proposed a construction heuristic based on the Clark & Wright Savings algorithm and an Ant Colony Optimization algorithm based on the MAX-MIN Ant System. Our results demonstrate the effectiveness of these methods in producing energy-efficient routing plans for electric vehicles in realistic scenarios.

CHAPTER 8

CONCLUSIONS AND OUTLOOK

8.1 Conclusions

Grounded in the pressing need to mitigate environmental pollution—a direct consequence of uncontrolled industrialization and urban expansion—this thesis delves into the domain of sustainable logistics with an emphasis on the urban landscape. The motivation for this thesis is to be found in the critical challenge of air pollution, exacerbated by the extensive use of fossil fuels within the transportation sector. This has not only compromised environmental quality but also posed severe health risks, highlighting the urgent need for sustainable transportation solutions.

This work has contributed to both the theoretical and practical aspects of urban logistics optimization through an extensive exploration of Electric Vehicle Routing Problems (EVRP), Two-Echelon Vehicle Routing Problems (2E-VRP), and their novel integration into the Two-Echelon Electric Vehicle Routing Problem (2E-EVRP). By considering environmentally friendly vehicles, such as electric vehicles (EVs), and implementing strategic multi-echelon distribution systems, this thesis points out pivotal strategies for enhancing sustainability within the logistics sector. The application of advanced optimization techniques and the development of novel solution approaches such as the self-adaptive version of the Construct, Merge, Solve, and Adapt (CMSA) algorithm, Adapt-CMSA, and Variable Neighborhood Search offer practical solutions to complex logistics challenges.

The findings from this study underscore the potential of integrating EVs into urban logistics systems and highlight the benefits of multi-echelon distribution models in reducing traffic congestion, air pollution, and overall carbon footprint. The exploration of EVRPs with additional real-world constraints, such as time windows, simultaneous pickups and deliveries, partial battery charging, and the consideration of road junctions and types, has advanced our understanding of sustainable city logistics.

- **Chapter 2** presented a variant of the hybrid metaheuristic CMSA called Adapt-CMSA. This algorithm variant was introduced in this thesis to mitigate some potential disadvantages of the standard CMSA algorithm, including a rather high sensitivity to parameter value settings, which may cause a need for specific parameter tuning applications in the context of diverse problem instance sets.

- **Chapter 3** presented the comparison of standard CMSA with the newly developed Adapt-CMSA variant in the context of the Minimum Positive Influence Dominating Set Problem (MPIDS). The MPIDS was chosen for this purpose due to its relative simplicity compared to complex vehicle routing problems. Indeed, the results of CMSA obtained for the MPIDS problem showed that, despite its superiority over other greedy algorithms and metaheuristics from the literature on many small and large instances, its performance starts to decrease considerably in the context of large instances. Adapt-CMSA, on the contrary, due to a dynamical adjustment of its parameters during run-time, was shown to be able to solve problem instances of all sizes without further tuning efficiently. Our experiments focused on the MPIDS problem revealed Adapt-CMSA's superiority over standard CMSA.

- **Chapter 4** explored Adapt-CMSA's application to Electric Vehicle Routing Problems with Time Windows, Simultaneous Pickup and Delivery, and Partial Recharging. Initially, the problem is formulated as Mixed-Integer Linear Programming (MILP), enabling its solution using any ILP solver. To address the problem, we developed two Adapt-CMSA variants. The first variant, named Adapt-Cmsa-Std, employs two-dimensional binary matrices for solution and sub-instance representation, with standard assignment-type MILP formulations utilized for solving sub-instances. Computational experiments revealed that assignment-type MILP models generally do not provide good lower bounds, leading to difficulties for the ILP solver in solving even small-sized sub-instances within restricted time constraints. Consequently, we introduced a second algorithm variant, Adapt-Cmsa-SetCov. In this approach, solutions and sub-instances are considered as combinations of routes, meaning solution components are the routes themselves. To solve a sub-instance, we utilized a set-covering-based formulation, which is based on the routes generated during the construction step and then selecting the most promising combination of routes so that each customer is visited in one of the

selected routes. The results convincingly demonstrated that CMSA variants employing set-covering-based models significantly outperform those using standard assignment-type models. In our view, CMSA algorithms present an ideal framework for leveraging set-covering-based models to solve optimization problems that can be modeled in this manner. This is attributed to the fact that CMSA algorithms are less sophisticated and easier to implement when compared for example, to column generation approaches. Moreover, CMSA algorithms are capable of exploring search spaces, unlike simpler heuristic methods from the literature designed to benefit from set-covering-based models.

- In **Chapter 5**, we addressed the Two-Echelon Electric Vehicle Routing Problem with Simultaneous Pickups and Deliveries (2E-EVRP-SPD), which reflects an integration of sustainable logistics practices through a two-echelon framework and the use of electric vehicles. In this problem, we consider that large trucks with internal combustion engines transport products from central warehouses to satellites in the surroundings of cities. Subsequently, smaller electric vehicles distribute goods from these satellites to customers located in the cities. Moreover, as the name of the problem already indicates, it also considers simultaneous pickup and delivery (SPD) constraints for the delivery of goods to customers. When SPD constraints are considered, each customer may have two different demands: (1) the goods to be delivered to the demand point (*delivery demand*), and (2) the goods to be collected from the demand point (*pickup demand*). So, once a vehicle visits a certain customer, both demands must be met simultaneously. This approach usually arises as a reverse logistics practice. However, despite the importance of reverse logistics in terms of sustainability, the number of publications on EVRP-SPD variants is rather limited.

  First, the addressed problem was formulated as a mixed integer linear program (MILP). Any general-purpose MILP solver may be used to solve this model. However, due to the multi-tier structure of the distribution network, the limited driving range of electric vehicles, and the SPD constraints, the 2E-EVRP-SPD problem is rather complex. In fact, our computational experiments showed that CPLEX struggled to solve even small-sized problems to optimality. In fact, in most cases, CPLEX was only able to derive valid solutions with large optimality gaps. Therefore, two probabilistic construction heuristics based on a Clarke-Wright Savings and a Sequential insertion algorithm were implemented. Moreover, in line with

the findings presented in **Chapter 4**, we developed an Adapt-CMSA variant in which a set-covering-based approached is used to solve the sub-instances. The proposed hybrid approach performed better than both CPLEX and the two construction heuristics.

- **Chapter 6** considered another variant of the Two-Echelon Electric Vehicle Routing Problem (2E-EVRP), this time incorporating time window constraints instead of SPD constraints. The chapter first technically described the problem by means of modelling it through a three-index node-based MILP model. The multi-tier distribution network, combined with the limited driving range of electric vehicles and the specific demands of time window constraints, presents a complex challenge. Our computational findings revealed that the MILP solver CPLEX was only able to optimally solve problems of smaller size. To address larger instances, we introduced a Variable Neighborhood Search (VNS) approach, enhanced by an initial solution generation technique inspired by Clarke-Wright Savings algorithm, tailored to the unique requirements of the 2E-EVRP-TW.

  The developed VNS approach utilized a wide range of both classical and large neighborhood search operators. Moreover, the algorithm also allows for the exploration of not only feasible but also unfeasible solutions through an augmented objective function. Our approach employed a variable neighborhood descent algorithm during the local search phase, with a notable distinction between two VNS variants: the comprehensive version, which incorporates destroy and repair mechanisms alongside standard routing operators, and a simplified version focusing solely on standard inter-route and intra-route shaking operators. This comparative analysis underscored the significant value added by the destroy and repair mechanisms in enhancing VNS performance.

  The results obtained by applying our VNS algorithm to the instances were promising, consistently outperforming CPLEX in speed and efficiency for all small-sized instances. Moreover, the inclusion of destroy-and-repair operators markedly improved the algorithm's overall effectiveness, showcasing its potential in solving complex routing problems.

- Finally, **Chapter 7** introduced the Electric Vehicle Routing Problem with Road Junctions and Road Types (EVRP-RJ-RT), an extension that significantly narrows the gap between theoretical EVRP models and the intricacies of real-world urban logistics. This model integrates road junctions and diverse road types, each with unique speed

constraints—including highways, urban roads, and city streets—thereby elevating the complexity and realism of the model to more accurately mirror the transportation networks found in urban environments. A novel aspect of our model is the consideration of speed as a decision variable, adding an extra dimension of complexity by directly affecting energy consumption based on load, speed, and distance.

We formulated this problem as a Mixed Integer Non-Linear Programming (MINLP) model, incorporating an objective function alongside battery constraints that closely simulate the energy expenditure of electric vehicles under varying conditions. To tackle the computational challenges inherent in this more sophisticated EVRP variant, we introduce a construction heuristic inspired by the Clarke-Wright Savings algorithm, coupled with an Ant Colony Optimization (ACO) technique derived from the MAX-MIN Ant System. These methods have proven to be effective in devising energy-efficient routing plans that are viable in practical urban settings, highlighting their potential to significantly improve the sustainability and efficiency of urban logistics involving electric vehicles.

This thesis has made a range of contributions to the field of sustainable logistics by thoroughly examining and addressing the complexities of electric vehicle routing in urban contexts. The presented work employs a multi-faceted approach that integrates environmental considerations with advanced optimization techniques to reduce pollution and greenhouse gas emissions while providing scalable and efficient solutions for urban logistics systems. The thesis shows the innovative application of the newly developed Adapt-CMSA algorithm, along with the exploration of novel problem formulations such as the 2E-EVRP with simultaneous pickups and deliveries and the inclusion of real-world constraints like road junctions and types. The findings presented across the chapters provide valuable insights into the adoption of electric vehicles and multi-echelon distribution systems, demonstrating their effectiveness in minimizing environmental impact while maintaining operational efficiency.

8.2 Outlook

As mentioned in the previous section, this thesis has paved the way for a comprehensive exploration of sustainable logistics within urban contexts, employing advanced optimization techniques to address the complexities of electric vehicle routing. While these are numerous contributions to be found in this work, there are also several avenues for further research and development,

particularly from an algorithmic perspective. The outlook for future work is discussed below, focusing on improvements regarding algorithmic approaches, and the development of realistic datasets.

From the algorithmic perspective, the CMSA algorithm stands as a cornerstone in this thesis. It is grounded on the idea of the generation of sub-instances of the original problem instance that contain high-quality solutions. It employs a systematic approach to generating sub-instances of original problem instances and solving these sub-instances using an exact method or solver. In this line, three primary areas for improvement can be mentioned:

- **Creation of High-Quality Sub-Instances:** The quality of sub-instances, key to the CMSA's efficiency, depends significantly on the construct step's ability to generate useful solutions. While a direct correlation between the construct step's output and the solve step's solution quality is anticipated, a more nuanced understanding reveals that the combinations of solution components may yield high-quality sub-instances irrespective of individual solution quality. This indicates the potential for employing diverse solution construction mechanisms to foster a rich variety of sub-instances, potentially enhancing search space exploration. However, achieving a balance between diversity and coherence in sub-instance composition is crucial. It may not always be the case that having a very diverse sub-instance includes high-quality solutions. Here, the most crucial issue is merging the most complementary solution components. And this may not always be directly related to the quality of the solutions generated in the construct step. Advanced machine learning techniques could play a pivotal role here, guiding the solution construction process by analyzing patterns and qualities of previously successful sub-instances, thereby optimizing the selection and combination of solution components.

- **Efficient Management of Sub-Instance Size:** As sub-instances grow in size, the performance of exact solvers like CPLEX may diminish due to increased computational complexity. The Adapt-CMSA attempts to mitigate this by adjusting solution similarity parameters, yet this approach can inadvertently limit diversity and exploration capabilities. An alternative strategy might involve refining the sub-instance through selective exclusion of components unlikely to contribute to high-quality solutions, based on compatibility and utility rather than individual merit. In this way, the diversity of sub-instances can also be maintained. Incorporating techniques such as machine learning for dynamic refinement could enhance the solver's

efficiency without compromising the algorithm's exploratory depth.

- **Enhanced Problem Formulation:** As mentioned above, the transition from assignment-type MILP models to set-covering models significantly improved solver performance within this thesis. This suggests that exploring alternative, more efficient modeling techniques could further leverage solver capabilities, potentially offering more scalable and accurate solutions to complex routing problems. Furthermore, employing more sophisticated exact solution approaches would undoubtedly result in solving sub-instances more efficiently.

In addition to the algorithmic enhancements, another future research line is the generation of realistic datasets. A critical aspect of validating and testing the proposed methodologies involves employing datasets that closely mirror real-world scenarios. Future research should focus on creating or augmenting datasets to include authentic road networks and geographic locations. This entails not just the physical layout but also the dynamic elements such as traffic patterns, varying road conditions, and logistical constraints, thereby ensuring that solution methodologies are tested against the multifaceted challenges present in actual urban logistics scenarios.

To this end, the author is currently in the process of developing a generator for realistic EVRP-RJ-RT instances based on geographical information extracted from OpenStreetMap [141]. An example instance generated by the preliminary version of this generator is presented in Figure 8.1.

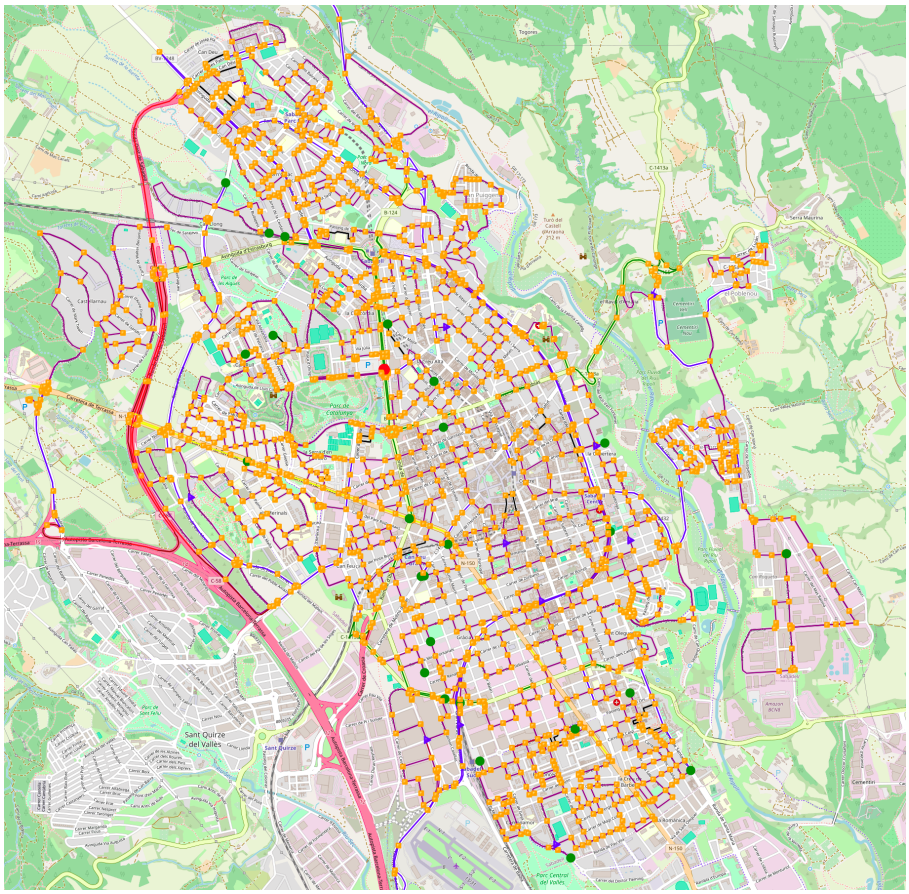


Fig. 8.1 Example instance of EVRP-RJ-RT generated using preliminary version of the instance generator.

References

[1] Rami Abousleiman and Osamah Rawashdeh. Tabu search based solution to the electric vehicle energy efficient routing problem. In *2014 IEEE Transportation Electrification Conference and Expo (ITEC)*, pages 1–6. IEEE, 2014.

[2] Mehmet Anıl Akbay and Christian Blum. Application of CMSA to the minimum positive influence dominating set problem. In *Artificial Intelligence Research and Development*, pages 17–26. IOS Press, 2021.

[3] Mehmet Anıl Akbay, Can Berk Kalayci, Christian Blum, and Olcay Polat. Variable neighborhood search for the two-echelon electric vehicle routing problem with time windows. *Applied Sciences*, 12(3):1014, 2022.

[4] Mehmet Anıl Akbay, Albert López Serrano, and Christian Blum. A self-adaptive variant of CMSA: Application to the minimum positive influence dominating set problem. *International Journal of Computational Intelligence Systems*, 15(1):1–13, 2022.

[5] Mehmet Anıl Akbay, Can Berk Kalayci, and Christian Blum. Application of CMSA to the electric vehicle routing problem with time windows, simultaneous pickup and deliveries, and partial vehicle charging. In *Metaheuristics. MIC 2022*, volume 13838 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2023. doi: 10.1007/978-3-031-26504-4_1.

[6] Mehmet Anıl Akbay, Can Berk Kalayci, and Christian Blum. Application of Adapt-CMSA to the two-echelon electric vehicle routing problem with simultaneous pickup and deliveries. In *Evolutionary Computation in Combinatorial Optimization. EvoCOP 2023*, volume 13987 of *Lecture Notes in Computer Science*, pages 16–33. Springer, 2023. doi: 10.1007/978-3-031-30035-6_2.

[7] İ. Kuban Altınel and Temel Öncan. A new enhancement of the clarke and wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 56(8):954–961, 2005.

[8] Rimmi Anand, Divya Aggarwal, and Vijay Kumar. A comparative analysis of optimization solvers. *Journal of Statistics and Management Systems*, 20(4): 623–635, 2017.

[9] Enrico Angelelli and Renata Mansini. The vehicle routing problem with time windows and simultaneous pick-up and delivery. In *Quantitative approaches to distribution logistics and supply chain management*, pages 249–267. Springer, 2002.

[10] Divansh Arora, Parikshit Maini, Pedro Pinacho-Davidson, and Christian Blum. Route planning for cooperative air-ground robots with fuel constraints: An approach based on CMSA. In *Proceedings of GECCO 2019 – Genetic and Evolutionary Computation Conference*, page 207–214, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Mohammad Asghari and S. Mohammad J. Mirzapour Al-e-hashem. Green vehicle routing problem: A state-of-the-art review. *International Journal of Production Economics*, 231:107899, 2021.

[12] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz. *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 2018.

[13] Qinghai Bai. Analysis of particle swarm optimization algorithm. *Computer and Information Science*, 3(1):180, 2010.

[14] Roberto Baldacci, Aristide Mingozzi, Roberto Roberti, and Roberto Wolfler Calvo. An exact algorithm for the two-echelon capacitated vehicle routing problem. *Operations Research*, 61(2):298–314, 2013.

[15] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[16] Jonathan F. Bard, Liu Huang, Moshe Dror, and Patrick Jaillet. A branch and cut algorithm for the VRP with satellite facilities. *IIE Transactions*, 30 (9):821–834, 1998.

[17] Matthew Barth, Theodore Younglove, and George Scora. Development of a heavy-duty diesel modal emissions and fuel consumption model. California PATH Research Report UCB-ITS-PRR-2005-1, California PATH Program, Institute of Transportation Studies, University of California, Berkeley, January 2005.

[18] John E. Beasley. Route first—cluster second methods for vehicle routing. *Omega*, 11(4):403–408, 1983.

[19] Tolga Bektaş and Gilbert Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, 2011.

[20] Onder Belgin, Ismail Karaoglan, and Fulya Altiparmak. Two-echelon vehicle routing problem with simultaneous pickup and delivery: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, 115:1–16, 2018.

[21] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[22] Houssem Eddine Ben-Smida, José-Francisco Chicano-García, and Saoussen Krichen. Construct, merge, solve and adapt for taxi sharing. *RIUMA (Repositorio Institucional de la Universidad de Malaga)*, 2019. URL `https://hdl.handle.net/10630/18106`.

[23] Dimitris J. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40(3):574–585, 1992.

[24] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353–373, 2005.

[25] Christian Blum and Marco Dorigo. The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (cybernetics)*, 34(2):1161–1172, 2004.

[26] Christian Blum, Jakob Puchinger, Günther R. Raidl, Andrea Roli, et al. A brief survey on hybrid metaheuristics. *Proceedings of BIOMA*, pages 3–18, 2010.

[27] Christian Blum, Pedro Pinacho Davidson, Manuel López-Ibáñez, and José A. Lozano. Construct, merge, solve & adapt: A new general algorithm for combinatorial optimization. *Computers & Operations Research*, 68:75–88, 2016.

[28] Salim Bouamama and Christian Blum. An improved greedy heuristic for the minimum positive influence dominating set problem in social networks. *Algorithms*, 14(3):79, 2021.

[29] Ulrich Breunig, Roberto Baldacci, Richard F. Hartl, and Thibaut Vidal. The electric two-echelon vehicle routing problem. *Computers & Operations Research*, 103:198–210, 2019.

[30] Maurizio Bruglieri, Simona Mancini, Ferdinando Pezzella, Ornella Pisacane, and Stefano Suraci. A three-phase matheuristic for the time-effective electric vehicle routing problem with partial recharges. *Electronic Notes in Discrete Mathematics*, 58:95–102, 2017.

[31] B. Calvo and G. Santafé. scmamp: Statistical comparison of multiple algorithms in multiple problems. *The R Journal*, 8(1), 2016.

[32] Siqi Cao, Wenzhu Liao, and Yuqi Huang. Heterogeneous fleet recyclables collection routing optimization in a two-echelon collaborative reverse logistics network from circular economic and environmental perspective. *Science of the Total Environment*, 758:144062, 2021.

[33] Craig R. Carter and Dale S. Rogers. A framework of sustainable supply chain management: moving toward new theory. *International Journal of Physical Distribution & Logistics Management*, 38(5):360–387, 2008.

[34] Alberto Ceselli, Ángel Felipe, M Teresa Ortuño, Giovanni Righini, and Gregorio Tirado. A branch-and-cut-and-price algorithm for the electric vehicle routing problem with multiple technologies. In *Operations Research Forum*, volume 2, pages 1–33. Springer, 2021.

[35] Camilo Chacón Sartori, Christian Blum, and Gabriela Ochoa. STNWeb: A new visualization tool for analyzing optimization algorithms. *Software Impacts*, 17:100558, 2023.

[36] I. Ming Chao, Bruce L. Golden, and Edward Wasil. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *American Journal of Mathematical and Management Sciences*, 13(3-4): 371–406, 1993.

[37] Wei-Neng Chen, Jun Zhang, Henry S.H. Chung, Wen-Liang Zhong, Wei-Gang Wu, and Yu-hui Shi. A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Transactions on Evolutionary Computation*, 14(2):278–300, 2009.

[38] Weidong Chen, Hao Zhong, Lidong Wu, and Ding-Zhu Du. A general greedy approximation algorithm for finding minimum positive influence dominating sets in social networks. *Journal of Combinatorial Optimization*, pages 1–20, 2021.

[39] Young Sul Cho, Jin Seop Kim, Juyong Park, Byungnam Kahng, and Doochul Kim. Percolation transitions in scale-free networks under the achlioptas process. *Physical Review Letters*, 103(13):135702, 2009.

[40] Nicos Christofides and Samuel Eilon. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20: 309–318, 1969.

[41] Geoff Clarke and John W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[42] Leandro C. Coelho and Gilbert Laporte. Classification, models and exact algorithms for multi-compartment delivery problems. *European Journal of Operational Research*, 242(3):854–864, 2015.

[43] Ryan G. Conrad and Miguel Andres Figliozzi. The recharging vehicle routing problem. In *Proceedings of the 2011 industrial engineering research conference*, page 8. IISE Norcross, GA, 2011.

[44] David L. Cortés-Murcia, Caroline Prodhon, and H. Murat Afsar. The electric vehicle routing problem with time windows, partial recharges and satellite customers. *Transportation Research Part E: Logistics and Transportation Review*, 130:184–206, 2019.

[45] Teodor Gabriel Crainic, Nicoletta Ricciardi, and Giovanni Storchi. Advanced freight transportation systems for congested urban areas. *Transportation Research Part C: Emerging Technologies*, 12(2):119–137, 2004.

[46] Teodor Gabriel Crainic, Nicoletta Ricciardi, and Giovanni Storchi. Models for evaluating and planning city logistics systems. *Transportation Science*, 43 (4):432–454, 2009.

[47] Teodor Gabriel Crainic, Guido Perboli, and Mariangela Rosano. Simulation of intermodal freight transportation systems: a taxonomy. *European Journal of Operational Research*, 270(2):401–418, 2018.

[48] Monica Crippa, Diego Guizzardi, Manjola Banja, Efisio Solazzo, Marilena Muntean, Edwin Schaaf, Federico Pagani, Fabio Monforti-Ferrario, JGJ Olivier, Roberta Quadrelli, et al. Co2 emissions of all world countries. *JRC Science for Policy Report, European Commission, EUR*, 31182, 2022.

[49] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *Interjournal, Complex Systems*, 1695(5):1–9, 2006.

[50] George B. Dantzig and John H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[51] Nico Dellaert, Fardin Dashty Saridarq, Tom Van Woensel, and Teodor Gabriel Crainic. Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows. *Transportation Science*, 53(2):463–479, 2019.

[52] Emrah Demir, Tolga Bektaş, and Gilbert Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–359, 2012.

[53] Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405, 2016.

[54] Edsger W. Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290. 2022.

[55] Marko Djukanović, Aleksandar Kartelj, and Christian Blum. Self-adaptive CMSA for solving the multidimensional multi-way number partitioning problem. *Expert Systems with Applications*, page 120762, 2023.

[56] Marco Dorigo and Thomas Stützle. *Ant colony optimization*. MIT press, 2004.

[57] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1991.

[58] Moshe Dror and Pierre Trudeau. Savings by split delivery routing. *Transportation Science*, 23(2):141–145, 1989.

[59] Moshe Dror and Pierre Trudeau. Split delivery routing. *Naval Research Logistics (NRL)*, 37(3):383–402, 1990.

[60] Ece Naz Duman, Duygu Taş, and Bülent Çatay. Branch-and-price-and-cut methods for the electric vehicle routing problem with time windows. *International Journal of Production Research*, 60(17):5332–5353, 2022.

[61] Nicolas Dupin and El-Ghazali Talbi. Matheuristics to optimize refueling and maintenance planning of nuclear power plants. *Journal of Heuristics*, 27 (1):63–105, 2021.

[62] Sevgi Erdoğan and Elise Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1): 100–114, 2012.

[63] European Environment Agency. Contribution to eu-27 emissions, 2023. URL `https://www.eea.europa.eu/data-and-maps/daviz/contribution-to-eu-27-emissions/#tab-googlechartid_googlechartid_chart_112`. Accessed April 10, 2024.

[64] Mai Fei and Chen Weidong. An improved algorithm for finding minimum positive influence dominating sets in social networks. *Journal of South China Normal University*, 48(3):59–63, 2016.

[65] Dominique Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424, 2010.

[66] Ángel Felipe, M. Teresa Ortuño, Giovanni Righini, and Gregorio Tirado. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, 71:111–128, 2014.

[67] Martina Fischetti and Matteo Fischetti. *Matheuristics*, pages 121–153. Springer International Publishing, 2018.

[68] Lester Randolph Jr. Ford and Delbert R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101, 1958.

[69] Angela K. Fournier, Erin Hall, Patricia Ricke, and Brittany Storey. Alcohol and the social network: Online social networking sites and college students' perceived drinking norms. *Psychology of Popular Media Culture*, 2(2):86, 2013.

[70] Peter M. Francis, Karen R. Smilowitz, and Michal Tzur. The period vehicle routing problem and its extensions. *the Vehicle Routing Problem: Latest Advances and New Challenges*, pages 73–102, 2008.

[71] Aurélien Froger, Jorge E. Mendoza, Ola Jabali, and Gilbert Laporte. *A matheuristic for the electric vehicle routing problem with capacitated charging*

stations. PhD thesis, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport (CIRRELT), 2017.

[72] Piotr Fronczak. Scale-free nature of social networks. In Reda Alhajj and Jon Rokne, editors, *Encyclopedia of Social Network Analysis and Mining*, pages 2300–2309. Springer New York, 2018.

[73] Ahmed G. Gad. Particle swarm optimization algorithm and its applications: a systematic review. *Archives of Computational Methods in Engineering*, 29(5): 2531–2561, 2022.

[74] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677 – 2694, 2008.

[75] Sotiris P. Gayialis, Grigorios D. Konstantakopoulos, and Ilias P. Tatsiopoulos. Vehicle routing problem for urban freight transportation: A review of the recent literature. *Operational Research in the Digital Era–ict Challenges*, pages 89–104, 2019.

[76] Michel Gendreau, Gilbert Laporte, and René Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.

[77] Robert B. Gibson. Sustainability assessment: basic components of a practical approach. *Impact Assessment and Project Appraisal*, 24(3):170–182, 2006.

[78] Billy E. Gillett and Leland R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974.

[79] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[80] Fred Glover and Manuel Laguna. *Tabu search*. Springer, 1998.

[81] Marc Goetschalckx and Charlotte Jacobs-Blecha. The vehicle routing problem with backhauls. *European Journal of Operational Research*, 42(1): 39–51, 1989.

[82] Bruce Golden, Arjang Assad, Larry Levy, and Filip Gheysens. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11 (1):49–66, 1984.

[83] Yue-Jiao Gong, Jun Zhang, Ou Liu, Rui-Zhang Huang, Henry Shu-Hung Chung, and Yu-Hui Shi. Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (applications and Reviews)*, 42(2):254–267, 2011.

[84] Philippe Grangier, Michel Gendreau, Fabien Lehuédé, and Louis-Martin Rousseau. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research*, 254(1):80–91, 2016.

[85] Wei-Jie Guan, Xue-Yan Zheng, Kian Fan Chung, and Nan-Shan Zhong. Impact of air pollution on the burden of chronic respiratory diseases in china: time for urgent action. *The Lancet*, 388(10054):1939–1951, 2016.

[86] Dilek Günneç, Subramanian Raghavan, and Rui Zhang. Least-cost influence maximization on social networks. *INFORMS Journal on Computing*, 32(2):289–302, 2020.

[87] LLC Gurobi Optimization. Gurobi optimizer reference manual. `http://www.gurobi.com`, 2020. Accessed : 2020-02-06.

[88] Pierre Hansen and Nenad Mladenovic. A tutorial on variable neighborhood search. *Les Cahiers Du GERAD*, 711:2440, 2003.

[89] Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José A. Moreno Pérez. *Variable neighborhood search*. Springer, 2019.

[90] Vera C. Hemmelmayr, Karl F. Doerner, and Richard F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009.

[91] Vera C. Hemmelmayr, Jean-François Cordeau, and Teodor Gabriel Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228, 2012.

[92] Gerhard Hiermann, Jakob Puchinger, Stefan Ropke, and Richard F. Hartl. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3): 995–1018, 2016.

[93] Gerhard Hiermann, Richard F. Hartl, Jakob Puchinger, and Thibaut Vidal. Routing a mix of conventional, plug-in hybrid, and electric vehicles. *European Journal of Operational Research*, 272(1):235–248, 2019.

[94] John H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[95] Andrés Huerta-Rojo, Elizabeth Montero, and Nicolás Rojas-Morales. An ant-based approach to solve the electric vehicle routing problem with time windows and partial recharges. In *2021 40th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–8. IEEE, 2021.

[96] IBM. ILOG CPLEX optimization studio documentation. *http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer*, 2020. Accessed : 2024-02-16.

[97] International Energy Agency (IEA). CO2 Emissions in 2022. https://www.iea.org/reports/co2-emissions-in-2022, 2023. License: CC BY 4.0.

[98] International Energy Agency (IEA). Global energy sector co2 emissions reductions by measure in the sustainable development scenario relative to the stated policies scenario, 2023. URL https://www.iea.org/data-and-statistics/charts/global-energy-sector-co2-emissions-reductions-by-measure-in-the-sustainable-development-scenario-relative-to-the-stated-policies-scenario. License: CC BY 4.0.

[99] Bassem Jarboui, Houda Derbel, Saïd Hanafi, and Nenad Mladenović. Variable neighborhood search for location routing. *Computers & Operations Research*, 40(1):47–57, 2013.

[100] Mads Jepsen, Simon Spoorendonk, and Stefan Ropke. A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. *Transportation Science*, 47(1):23–37, 2013.

[101] Wanchen Jie, Jun Yang, Min Zhang, and Yongxi Huang. The two-echelon capacitated electric vehicle routing problem with battery swapping stations: Formulation and efficient methodology. *European Journal of Operational Research*, 272(3):879–904, 2019.

[102] Sašo Karakatič. Optimizing nonlinear charging times of electric vehicle routing with genetic algorithm. *Expert Systems with Applications*, 164:114039, 2021.

[103] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[104] Merve Keskin and Bülent Çatay. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65:111–127, 2016.

[105] Merve Keskin and Bülent Çatay. A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & Operations Research*, 100:172–188, 2018.

[106] Merve Keskin, Bülent Çatay, and Gilbert Laporte. A simulation-based heuristic for the electric vehicle routing problem with time windows and stochastic waiting times at recharging stations. *Computers & Operations Research*, 125:105060, 2021.

[107] Scott Kirkpatrick, C. Daniel Gelatt Jr., and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[108] Çağrı Koç, Tolga Bektaş, Ola Jabali, and Gilbert Laporte. A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. *Computers & Operations Research*, 64:11–27, 2015.

[109] Çağrı Koç, Tolga Bektaş, Ola Jabali, and Gilbert Laporte. The fleet size and mix location-routing problem with time windows: Formulations and a heuristic algorithm. *European Journal of Operational Research*, 248(1):33–51, 2016.

[110] Çağrı Koç, Ola Jabali, Jorge E Mendoza, and Gilbert Laporte. The electric vehicle routing problem with shared charging stations. *International Transactions in Operational Research*, 26(4):1211–1243, 2019.

[111] Christos Koulamas, S. R. Antony, and R. Jaen. A survey of simulated annealing applications to operations research problems. *Omega*, 22(1): 41–56, 1994.

[112] İlker Küçükoğlu, Reginald Dewil, and Dirk Cattrysse. Hybrid simulated annealing and tabu search method for the electric travelling salesman problem with time windows and mixed charging rates. *Expert Systems with Applications*, 134:279–303, 2019.

[113] Ilker Kucukoglu, Reginald Dewil, and Dirk Cattrysse. The electric vehicle routing problem and its variations: A literature review. *Computers & Industrial Engineering*, 161:107650, 2021.

[114] Gilbert Laporte, Francois Louveaux, and Hélène Mercure. The vehicle routing problem with stochastic travel times. *Transportation Science*, 26(3): 161–170, 1992.

[115] Allan Larsen and O. B. Madsen. *The dynamic vehicle routing problem*. PhD thesis, Institute of Mathematical Modelling, Technical University of Denmark, 2000.

[116] Christine Lellis. These 21 companies are switching to electric vehicle fleets, 2021. `https://www.perillon.com/blog/21-companies-switching-to-electric-vehicle-fleets`.

[117] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[118] Rhyd Lewis, Dhananjay Thiruvady, and Kerri Morgan. Finding happiness: an analysis of the maximum happy vertices problem. *Computers & Operations Research*, 103:265–276, 2019.

[119] Geng Lin, Jian Guan, and Huibin Feng. An ILP based memetic algorithm for finding minimum positive influence dominating sets in social networks. *Physica A: Statistical Mechanics and Its Applications*, 500:199–209, 2018.

[120] Geng Lin, Jinyan Luo, Haiping Xu, and Meiqin Xu. A hybrid swarm intelligence-based algorithm for finding minimum positive influence dominating sets. In Yong Liu, Lipo Wang, Liang Zhao, and Zhengtao Yu, editors, *Proceedings of ICNC-FSKD 2019 – Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery*, pages 506–511. Springer International Publishing, 2020.

[121] Qixing Liu, Peng Xu, Yuhu Wu, and Tielong Shen. A hybrid genetic algorithm for the electric vehicle routing problem with time windows. *Control Theory and Technology*, 20(2):279–286, 2022.

[122] Tian Liu, Zhixing Luo, Hu Qin, and Andrew Lim. A branch-and-cut algorithm for the two-echelon capacitated vehicle routing problem with grouping constraints. *European Journal of Operational Research*, 266(2): 487–497, 2018.

[123] Cheng Long and Raymond Chi-Wing Wong. Minimizing seed set for viral marketing. In *2011 IEEE 11th International Conference on Data Mining*, pages 427–436. IEEE press, 2011.

[124] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3: 43–58, 2016.

[125] Cathy Macharis, Sandra Melo, Johan Woxenius, and Tom Van Lier. *Sustainable logistics*. Emerald Group Publishing, 2014.

[126] Chryssi Malandraki and Mark S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, 1992.

[127] Simona Mancini. Multi-echelon distribution systems in city logistics. *European Transport/Trasporti Europei*, 54:1–24, 2013.

[128] Setyo Tri Windras Mara, Rachmadi Norcahyo, Panca Jodiawan, Luluk Lusiantoro, and Achmad Pratama Rifai. A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, 146:105903, 2022.

[129] Guillaume Marques, Ruslan Sadykov, Jean-Christophe Deschamps, and Rémy Dupas. An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Computers & Operations Research*, 114:104833, 2020.

[130] Michalis Mavrovouniotis, Georgios Ellinas, and Marios Polycarpou. Ant colony optimization for the electric vehicle routing problem. In *2018 IEEE Symposium series on computational intelligence (SSCI)*, pages 1234–1241. IEEE, 2018.

[131] Michalis Mavrovouniotis, Changhe Li, Georgios Ellinas, and Marios Polycarpou. Parallel ant colony optimization for the electric vehicle routing problem. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1660–1667. IEEE, 2019.

[132] Alan McKinnon. *Decarbonizing logistics: Distributing goods in a low carbon world*. Kogan Page Publishers, 2018.

[133] Anuj Mehrotra and Michael A. Trick. A branch-and-price approach for graph multi-coloring. *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, pages 15–29, 2007.

[134] Zbigniew Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In John R. McDonnell and Robert G. Reynolds, editors, *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming*, volume 4, pages 135–155, 1995.

[135] Gendreau Michel and Jean-Yves Potvin, editors. *Handbook of Metaheuristics*, volume 272 of *Series in Operations Research & Management Science*. Springer Nature Switzerland, 3rd edition, 2019.

[136] Nicholas L. Mills, Ken Donaldson, Paddy W. Hadoke, Nicholas A. Boon, William MacNee, Flemming R. Cassee, Thomas Sandström, Anders Blomberg, and David E. Newby. Adverse cardiovascular effects of air pollution. *Nature Clinical Practice Cardiovascular Medicine*, 6(1):36–44, 2009.

[137] John E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. volume 1, pages 65–77. Oxford, UK, 2002.

[138] Alejandro Montoya, Christelle Guéret, Jorge E. Mendoza, and Juan G. Villegas. The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, 103:87–110, 2017.

[139] Gábor Nagy, Niaz A. Wassan, M. Grazia Speranza, and Claudia Archetti. The vehicle routing problem with divisible deliveries and pickups. *Transportation Science*, 49(2):271–294, 2015.

[140] Gabriela Ochoa, Katherine M. Malan, and Christian Blum. Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. *Applied Soft Computing*, 109:107492, 2021.

[141] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2017.

[142] Ibrahim H Osman and Gilbert Laporte. Metaheuristics: A bibliography. *Annals of Operations research*, 63:511–623, 1996.

[143] Heinrich Paessens. The savings algorithm for the vehicle routing problem. *European Journal of Operational Research*, 34(3):336–344, 1988.

[144] Jiehui Pan and Tian-Ming Bu. A fast greedy algorithm for finding minimum positive influence dominating sets in social networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 360–364. IEEE, 2019.

[145] Sophie N. Parragh, Karl F. Doerner, and Richard F. Hartl. A survey on pickup and delivery problems. *Part II: Transportation Between Pickup and Delivery Locations, Journal Für Betriebswirtschaft*, 2007.

[146] Samuel Pelletier, Ola Jabali, and Gilbert Laporte. The electric vehicle routing problem with energy consumption uncertainty. *Transportation Research Part B: Methodological*, 126:225–255, 2019.

[147] Guido Perboli, Roberto Tadei, and Daniele Vigo. The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45(3):364–380, 2011.

[148] David Pisinger and Stefan Ropke. Large neighborhood search. In *Handbook of metaheuristics*, pages 399–419. Springer, 2010.

[149] David Pisinger and Stefan Ropke. *Large Neighborhood Search*, pages 99–127. Springer International Publishing, 2019.

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

[151] Amir Afrasiabi Rad and Morad Benyoucef. Towards detecting influential users in social networks. In *International Conference on E-Technologies*, pages 227–240. Springer, 2011.

[152] Hassan Raei, Nasser Yazdani, and Masoud Asadpour. A new algorithm for positive influence dominating set in social networks. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 253–257. IEEE, 2012.

[153] Günther R. Raidl. A unified view on hybrid metaheuristics. In *International workshop on hybrid metaheuristics*, pages 1–12. Springer, 2006.

[154] Mauricio G. C. Resende and Celso C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks: An International Journal*, 41(2): 104–114, 2003.

[155] Roberto Roberti and Min Wen. The electric traveling salesman problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 89:32–52, 2016.

[156] Yves Rochat and Éric D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1 (1):147–167, 1995.

[157] Dale S. Rogers and Ronald S. Tibben-Lembke. *Going backwards: reverse logistics trends and practices*. Center for Logistics Management, University of Nevada, Reno, Reverse Logistics Executive Council, 1999.

[158] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[159] Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.

[160] Mir Ehsan Hesam Sadati and Bülent Çatay. A hybrid variable neighborhood search approach for the multi-depot green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 149:102293, 2021.

[161] Saïd Salhi and Gábor Nagy. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50(10):1034–1042, 1999.

[162] Dimitrios Sariklis and Susan Powell. A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, 51: 564–573, 2000.

[163] Ons Sassi, Wahiba Ramdane Cherif-Khettaf, and Ammar Oulamara. Iterated tabu search for the mix fleet vehicle routing problem with heterogenous electric vehicles. In *Modelling, Computation and Optimization in Information Systems and Management Sciences: Proceedings of the 3rd International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences-MCO 2015-Part I*, pages 57–68. Springer, 2015.

[164] Martin Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45(6):831–841, 1997.

[165] Martin Savelsbergh and Marc Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[166] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.

[167] Michael Schneider, Andreas Stenger, and Julian Hof. An adaptive VNS algorithm for vehicle routing problems with intermediate stops. *OR Spectrum*, 37(2):353–387, 2015.

[168] Albert López Serrano, Teddy Nurcahyadi, Salim Bouamama, and Christian Blum. Negative learning ant colony optimization for the minimum positive influence dominating set problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, page 1974–1977, New York, NY, USA, 2021. Association for Computing Machinery.

[169] Yunfan Shan, Qinma Kang, Ran Xiao, Yiran Chen, and Yunfan Kang. An iterated carousel greedy algorithm for finding minimum positive influence dominating sets in social networks. *IEEE Transactions on Computational Social Systems*, 9(3):830–838, 2021.

[170] Sai Shao, Wei Guan, and Jun Bi. Electric vehicle-routing problem with charging demands and energy consumption. *IET Intelligent Transport Systems*, 12(3):202–212, 2018.

[171] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer, 1998.

[172] Natasja Sluijk, Alexandre M. Florio, Joris Kinable, Nico Dellaert, and Tom Van Woensel. Two-echelon vehicle routing problems: A literature review. *European Journal of Operational Research*, 304(3):865–886, 2023.

[173] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[174] Thomas Stützle and Holger H. Hoos. MAX–MIN ant system. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[175] Eiichi Taniguchi. City logistics. *Infrastructure Planning Review*, 18:1–16, 2001.

[176] Vasileios A. Tatsis and Konstantinos E. Parsopoulos. Dynamic parameter adaptation in metaheuristics using gradient approximation and line search. *Applied Soft Computing*, 74:368–384, 2019.

[177] Sam R. Thangiah, Ibrahim H. Osman, Rajini Vinayagamoorthy, and Tong Sun. Algorithms for the vehicle routing problems with time deadlines. *American Journal of Mathematical and Management Sciences*, 13(3-4):323–355, 1993.

[178] Paul Michael Thompson and James B. Orlin. The theory of cyclic transfers. 1989.

[179] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002.

[180] United Nations. Sustainable development goals, 2023. URL `https://sdgs.un.org/goals`. Accessed April 10, 2024.

[181] Dan Wang and Hong Zhou. A two-echelon electric vehicle routing problem with time windows and battery swapping stations. *Applied Sciences*, 11(22): 10779, 2021.

[182] Feng Wang, Erika Camacho, and Kuai Xu. Positive influence dominating set in online social networks. In *International Conference on Combinatorial Optimization and Applications*, pages 313–321. Springer, 2009.

[183] Feng Wang, Hongwei Du, Erika Camacho, Kuai Xu, Wonjun Lee, Yan Shi, and Shan Shan. On positive influence dominating sets in social networks. *Theoretical Computer Science*, 412(3):265–269, 2011.

[184] Guangyuan Wang. *Domination problems in social networks*. PhD thesis, University of Southern Queensland, 2014.

[185] Kangzhou Wang, Yeming Shao, and Weihua Zhou. Matheuristic for a two-echelon capacitated vehicle routing problem with environmental considerations in city logistics service. *Transportation Research Part D: Transport and Environment*, 57:262–276, 2017.

[186] Qing Wang, Shuai Peng, and Shuan Liu. Optimization of electric vehicle routing problem using tabu search. In *2020 Chinese control and decision conference (CCDC)*, pages 2220–2224. IEEE, 2020.

[187] Yong Wang, Kevin Assogba, Jianxin Fan, Maozeng Xu, Yong Liu, and Haizhong Wang. Multi-depot green vehicle routing problem with shared transportation resource: Integration of time-dependent speed and piecewise penalty cost. *Journal of Cleaner Production*, 232:12–29, 2019.

[188] Special Working Session WCED. World commission on environment and development. *Our Common Future*, 17(1):1–91, 1987.

[189] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[190] World Meteorological Organization. Record high greenhouse gas concentrations with no signs of peaking. `https://news.un.org/en/story/2023/11/1143607`, 2023. Accessed: 01.05.2024.

[191] Zhiguo Wu and Juliang Zhang. A branch-and-price algorithm for two-echelon electric vehicle routing problem. *Complex & Intelligent Systems*, 9(3):2475–2490, 2023.

[192] Yiyong Xiao, Yue Zhang, Ikou Kaku, Rui Kang, and Xing Pan. Electric vehicle routing problem: A systematic review and a new comprehensive model with nonlinear energy recharging and consumption. *Renewable and Sustainable Energy Reviews*, 151:111567, 2021.

[193] P.C. Yellow. A computational modification to the savings method of vehicle scheduling. *Journal of the Operational Research Society*, 21(2):281–283, 1970.

[194] Yusuf Yilmaz and Can B Kalayci. Variable neighborhood search algorithms to solve the electric vehicle routing problem with simultaneous pickup and delivery. *Mathematics*, 10(17):3108, 2022.

[195] Shuai Zhang, Yuvraj Gajpal, S.S. Appadoo, and M.M.S. Abdulkader. Electric vehicle routing problem with recharging stations for minimizing energy consumption. *International Journal of Production Economics*, 203: 404–413, 2018.

[196] Yan Wei Zhao, B. Wu, W.L. Wang, Ying Li Ma, W.A. Wang, and H. Sun. Particle swarm optimization for vehicle routing problem with time windows. In *Materials Science Forum*, volume 471, pages 801–805. Trans Tech Publ, 2004.

[197] Qing Zhu, Limin Qian, Yingchun Li, and Shanjun Zhu. An improved particle swarm optimization algorithm for vehicle routing problem with time windows. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1386–1390. IEEE, 2006.

[198] Yanfei Zhu, Kwang Y Lee, and Yonghua Wang. Adaptive elitist genetic algorithm with improved neighbor routing initialization for electric vehicle routing problems. *IEEE Access*, 9:16661–16671, 2021.

[199] Çağrı Koç and Ismail Karaoglan. The green vehicle routing problem: A heuristic based exact solution approach. *Applied Soft Computing*, 39:154–164, 2016. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2015.10.064.