Pontifícia Universidade Católica
do Rio de Janeiro

Jahir Desaily Llagas Ortega

The Electric Time-Dependent Capacitated Arc Routing Problem

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Engenharia de Produção of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia de Produção.

Advisor: Prof. Rafael Martinelli Pinto

Rio de Janeiro
October 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Jahir Desaily Llagas Ortega

The Electric Time-Dependent Capacitated Arc Routing Problem

Dissertation presented to the Programa de Pós–graduação em Engenharia de Produção of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia de Produção. Approved by the Examination Committee:

Prof. Rafael Martinelli Pinto
Advisor
Departamento de Engenharia Industrial – PUC-Rio

Prof. Sanne Wøhlk
Aarhus University

Dr. Eduardo Vieira Queiroga
Centre de Recherche Inria Bordeaux - Sud-Ouest

Rio de Janeiro, October 21st, 2022

Jahir Desaily Llagas Ortega

Graduado em engenharia industrial pela Pontificia Universidad Católica del Perú.

Bibliographic data

Llagas Ortega, Jahir Desaily

The Electric Time-Dependent Capacitated Arc Routing Problem / Jahir Desaily Llagas Ortega; advisor: Rafael Martinelli Pinto. – 2022.

63 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Industrial, 2022.

Inclui bibliografia

1.Engenharia Industrial–Teses. 2.Roteamento em Arcos Capacitados. 3.Veículos Elétricos. 4.Tempos de viagem dependentes do tempo. 5. Taxa de consumo de energia Dependente da Velocidade. I. Martinelli, Rafael. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Industrial. III. Título.

CDD: 658.5

To my parents
for giving me life and support to continue
developing in my beloved profession.

To my grandparents
for worrying about my physical
and mental health.

To my sister
for being aware of my progress
and motivating me to continue.

To my partner
for always believing in me
even when I couldn't do it myself.

To my nephew,
hoping he can rate the importance of
overcoming himself in the future
because times tend to be
more complex and competitive.

Acknowledgments

To my parents for being my fundamental pillar and for having supported me unconditionally, despite the adversities and inconveniences that arose.

To my advisor Professor Rafael Martinelli for his guidance, compression, patience, and collaboration in carrying out this work. Also, thank you for having introduced me years ago to the study of the topic of vehicle routing.

To CNPq and PUC-Rio, for the aids granted, without which this work does not could have been accomplished.

To the Faculty of Industrial Engineering professors with whom I have always received their support and who have participated in my training.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Llagas Ortega, Jahir Desaily; Martinelli, Rafael (Advisor). **The Electric Time-Dependent Capacitated Arc Routing Problem**. Rio de Janeiro, 2022. 63p. Dissertação de Mestrado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

With energy and environmental issues rising, electric vehicles (EVs) will become an essential mode of transportation in logistics distribution. A vital scenario to consider is the dependence of traffic congestion on vehicle travel times, as it is common in urban areas today. This feature means that the speed of an EV on each route may be distinct during different periods. Because EVs have a limited driving range, various works in the literature have proposed energy consumption models as a function of speed and aerodynamic factors. However, their application remains limited and oversimplified due to their dependence on speed and travel times. In the case of speed, the models in the literature work under an average speed during a given arc or introduce approximations with piece-wise linearization methods. Regarding travel times, current vehicle routing algorithms often reformulate the road network into a complete graph where each arc represents the quickest path between two locations. The results obtained by these methods differ from reality, particularly for Arc Routing Problems involving services on the arcs of a road network. For these reasons, we define the Electric Capacitated Arc Routing Problem with Time-dependent Travel times, and Speed-dependent Energy Consumption Rate (E-TDCARP). Over a planning horizon, each arc is associated with a step-wise speed function. Based on this function, a vehicle's speed can change while traveling on a given arc. The objective is to serve a set of arcs that require services through a fleet of electric vehicles with limited load and battery capacity, minimizing the total travel time. Furthermore, the energy consumption rate per unit of time traveled (ECR) is considered a non-linear function based on speed. We propose a closed-form energy consumption preprocessing algorithm without approximations. We embed it into an Iterate Local Search metaheuristic (ILS) for E-TDCARP and compare the impact on the design of routes between these alternative vehicles and conventional ones.

Keywords

Capacitated Arc Routing; Electric Vehicles; Time-Dependent Travel Times; Speed-Dependent Energy Consumption Rate.

Resumo

Llagas Ortega, Jahir Desaily; Martinelli, Rafael. **O Problema de Roteamento em Arcos Capacitados com Dependência de Tempo e Veiculos Elétricos** Rio de Janeiro, 2022.63p. Dissertação de Mestrado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

Com o aumento das questões energéticas e ambientais, os veículos elétricos (EVs) se tornarão um modo de transporte essencial na distribuição logística. Um cenário vital a ser considerado é a dependência do congestionamento do tráfego nos tempos de viagem dos veículos, como é comum nas áreas urbanas hoje. Esse recurso significa que a velocidade de um EV em cada rota pode ser distinta durante diferentes períodos. Como os EVs possuem autonomia limitada, vários trabalhos na literatura propuseram modelos de consumo de energia em função da velocidade e fatores aerodinâmicos. No entanto, sua aplicação permanece limitada e simplificada devido à sua dependência da velocidade e dos tempos de viagem. No caso da velocidade, os modelos da literatura trabalham sob uma velocidade média durante um determinado arco ou introduzem aproximações com métodos de linearização por partes. Em relação aos tempos de viagem, os atuais algoritmos de roteamento de veículos muitas vezes reformulam a rede viária em um gráfico completo onde cada arco representa o caminho mais rápido entre dois locais. Os resultados obtidos por esses métodos divergem da realidade, principalmente para problemas de roteamento de arco envolvendo serviços nos arcos de uma rede rodoviária. Por essas razões, definimos o Problema de Roteamento de Arco Capacitado Elétrico com tempos de viagem dependentes do tempo e taxa de consumo de energia dependente da velocidade. Ao longo de um horizonte de planejamento, cada arco está associado a uma função de velocidade passo a passo. O objetivo é atender um conjunto de arcos que demandam serviços por meio de uma frota de EVs com carga e capacidade de bateria limitadas, minimizando o tempo total de viagem. Além disso, a taxa de consumo de energia por unidade de tempo percorrido é considerada uma função não linear baseada na velocidade. Propomos um algoritmo de pré-processamento de consumo de energia de forma fechada sem aproximações. Nós o incorporamos em uma metaheurística Iterate Local Search e comparamos o impacto no projeto de rotas com os veículos convencionais.

Palavras-chave

Roteamento em Arcos Capacitados;VeículosElétricos;Temposdeviagem dependentes do tempo; Taxa de consumo de energia Dependente da Velocidade.

Table of contents

List of figures

List of tables

List of algorithms

List of Abreviations

EV – *Electric Vehicle*

ECR – *Energy Consumption Rate*

VRP – *Vehicle Routing Problem*

TD-VRP – *Time-Dependent Vehicle Routing Problem*

E-VRP – *Electric Vehicle Routing Problem*

ARP – *Arc Routing Problem*

CARP – *Capacitated Arc Routing Problem*

E-ARP – *Electric Vehicle Arc Routing Problem*

TDCARP – *Time-Dependent Capacitated Arc Routing Problem*

E-TDCARP – *Electric Capacitated Arc Routing Problem with Time-dependent travel times and Speed-Dependent energy consumption rate*

E-TDARP – *Electric Arc Routing Problem with Time-dependent travel times and Speed-Dependent energy consumption rate*

ILS – *Iterated Local Search*

VND – *Variable Neighborhood Descend*

ACO – *Ant Colony Optimization*

URM – *Uniform Rectilinear Motion*

LB – *Lower Bounds*

AMSL – *Above Mean Sea Level*

kWh – *Kilowatt-hours*

Abrió los ojos a las cuatro de la madrugada y
pensó: "Hoy comienzas a cambiar el mundo".

Mario Vargas Llosa, *El paraíso en la otra esquina.*

1
Introduction

In recent decades, the increase in ecological awareness has driven initiatives in many countries ranging from regulatory policies related to greenhouse gas emissions to develop and implement new alternative energy sources. For example, the US has energy policies (WATKISS; SMITH, 1993; SISSINE, 2007) to reduce the use of fossil fuels to reduce CO_2, eliminate dependence on foreign oil and support the use of renewable energy (SISSINE, 2007). These policies led to the creation of regulations, mandates, and fiscal incentives that encourage companies' use of alternative fuel vehicles.

Due to this, the interest of companies in green logistics practices is increasing mainly in the transport sector since the CO_2 emissions caused by this sector represented 26% of global CO_2 emissions (IEA, 2019). Furthermore, that same year, the transport sector was the second largest contributor to CO_2 emissions, according to the International Energy Agency (2019). Electric vehicles (EVs) seem to be a potential alternative since they reduce oil consumption and CO_2 emissions (NIEKERK; AKKER; HOOGEVEEN, 2017). However, obstacles such as the limited driving range, battery degradation, and long charging times continue to hamper their promotion. For this reason, the number of works related to implementing electric vehicles in routing problems has grown significantly. It has become a relevant research topic within modern logistics that academia and industry are exploring and solving (XIAO et al., 2021).

The EV studies focus on charging policies, electric charging models, heterogeneous stations, and energy consumption functions. Montoya et al. (2017) introduce the Electric Vehicle Routing Problem (E-VRP) considering non-linear recharge functions. Their formulation is a piecewise linear approximation considering a realistic non-linear recharge model based on the current State of Charge (SoC) level. However, the energy consumption is assumed to be linear, and the consumption rates are constant and equal for all arcs. Froger et al. (2019) proposed two efficient MIP formulations for the E-VRP of Montoya et al. (2017) and algorithms for the problem of finding the optimal charging decisions for a given route. In Froger et al. (2021), the same problem is addressed with the addition of limited recharging capacity at stations. Ceselli et al. (2021) propose an exact *branch-and-cut-and-price* algorithm for E-VRP with heterogeneous stations. They set energy consumption and recharge as a linear function. They use a non-negative coefficient in consumption and

recharge for each link and station, respectively.

With the popularization of electric vehicles, research on the EVs energy consumption model has seen rapid progress (YUAN et al., 2017). EV energy consumption studies focus on developing energy models (mechanical or real-data driving), evaluating influences on energy consumption such as speed and weight, and analyzing the feasibility and costs of introducing EVs into the transport network. Goeke and Schneider (2015) worked on the E-VRP with mixed fleets (electric and conventional vehicles) with time intervals where the vehicle must arrive to serve a customer. The authors use a realistic energy consumption model that incorporates speed, slope, and load distribution instead of routing models that assume energy consumption is a linear function. In Asamer et al. (2016), they carried out a sensitivity analysis on the non-linear energy consumption rate (ECR) based on the longitudinal dynamic model, which, unlike the Goeke and Schneider (2015) model, incorporates the efficiency of the electric motor and complementary consumption variables such as radio, lights, air conditioning, among others. Based on the work of Asamer et al. (2016), Fernández et al. (2020) implement this energy consumption model in an arc routing problem (E-ARP) where EVs serve clients along an arc or edge. Fernández et al. (2020) consider dynamic energy recharges (recharging a vehicle while driving) and, in addition, seek to determine the optimal travel speed that minimizes the travel time of each route. Abdallah and Adel (2020) faced the E-VRP with variable speed where the energy consumption is a function of the speed, and the edge traveled. In this work, they introduced piecewise linear range energy functions based on speed, and the model does not accept changes in speed during the trip (it works under an average speed). Basso et al. (2019) introduce a method to calculate the energy consumption coefficients considering topographic factors and speed. The model generates an approximation based on a linear mass function since they determined the weight during the route. Despite the considerable increase in papers related to the EVs' energy consumption, there are still *oversimplifications* related to non-linear models, such as average speeds, constant travel times, and piecewise linearization approximations. It is due to the high computational overhead generated when implementing an accurate consumption model with variable speeds and travel times influenced by some external factor such as traffic congestion, which is *time-dependent*.

Within the Optimization area, routing problems involve more and more real-life aspects, such as the case of time-dependency, where some features may change during the day. For example, travel time on a given street varies throughout the day due to traffic congestion. Ichoua et al. (2003) introduced

VRP with time-dependent travel times (TD-VRP), assigning each edge a piecewise constant travel speed function. Unlike classical VRP models that assume constant travel time, Ichoua et al. (2003) sought to eliminate this simplification since, in the real world, these times are influenced by the time of day and implicitly by traffic conditions. This work expanded the number of articles on time dependency in nodes. For example, Haghani and Jung (2005) worked on TD-VRP with a dynamic approach (adjusting vehicle routes at certain times) under the same premises as Ichoua et al. (2003). These adjustments take into account new travel time information, the current locations of the vehicles, and the most recent demands after the previous adjustment in the route plans. Spliet et al. (2018) introduce the TD-VRP with the time window assignment based on the work of Ichoua et al. (2003). This problem seeks to assign time windows to customers before their demand is known and create vehicle routes that adhere to these time windows after demand is known.

Although the application of time-dependency in travel times is a crucial factor for many routing problems, there are few works related to its application at the network level due to its difficulty and high computational overhead. The studies, as mentioned above, defined speed functions on a complete graph (each edge represents the quickest path between two nodes). Vidal et al. (2021) argue that this model is often inadequate, particularly for arc routing problems (ARP) involving services at the edges of a road network, because the quickest paths may vary. For theses reasons, the Time-dependent Capacitated Arc Routing Problem (TDCARP) was formally defined, with the speed of travel and service functions given directly at the network level.

Due to all the above, we seek to introduce Electric Vehicles within the Time-dependent Capacitated Arc Routing Problem with travel time from Vidal et al. (2021) under an Energy Consumption Rate model based on speed. In this problem, the EV will perform services in the edges of a network without violating the load and battery capacity of the vehicle within a certain period. Each EV has a non-linear energy consumption rate function (ECR) based on speed, similar to the model used in Asamer et al. (2016) and Fernández et al. (2020). Unlike the EV routing problems in the literature, we focus on eliminating the *oversimplifications* in Energy Consumption related to speed (single and constant) and travel time (constant) during the calculation of energy consumption. With this, we will seek to solve this new CARP variant through a metaheuristic approach. However, for this problem, we will assume that the vehicle's mass does not vary significantly, so our energy consumption model will not depend on the mass level. This problem has applications in the

postal service where vehicles have a capacity in units since the weight of letters and packages is not significant compared to their volumes.

1.1 Thesis Research Structure

The structure is organized into six chapters, including this introductory one. Chapters 2 presents the work's theoretical framework, where we will carry out a brief theoretical review related to electric vehicle routing models with energy consumption and time-dependent routing models. In addition, we will explain in more detail the non-linear energy consumption rate function based on the longitudinal dynamic model. Finally, We will explain the Capacitated Arc Routing Problem with Time-dependent travel times proposed by Vidal et al. (2021) and we will focus on its Quickest Paths preprocessing algorithm.

Chapters 3 will define the Electric Vehicle Capacitated Arc Routing Problem with Time-dependent Travel Times and Speed-dependent Energy Consumption Rate as an extension of TDCARP. We will explain the speed model, the energy consumption model, premises, and assumptions used to develop the problem.

Chapters 4 focuses on the development of the proposed methodology. At this point, we will introduce the stepwise energy consumption rate (ECR) function based on the time using stepwise speed function. Furthermore, based on the work of Vidal et al. (2021), we will develop an algorithm for preprocessing energy consumption of the Quickest Paths given a departure time. Then, we will focus on the representation of the solution, calculation equations of exact travel time and energy consumption, lower bounds (proposed in Vidal et al. (2021)), and the construction of our Iterated Local Search (ILS) metaheuristic.

Chapters 5 presents the computational experiments, the results obtained by our metaheuristic, and the preprocessing algorithm implemented.

Chapters 6 presents the main conclusions and addresses suggestions to improve and expand the research.

2
Theoretical framework

2.1
Capacitated Arc Routing Problem

Arc routing problems are vehicle routing problems where services are located on some arcs or edges of a network, e.g., spreading salt on the road or collecting municipal refuse in a street (CORBERÁN; PRINS, 2010).

In 1981, Golden and Wong introduced the Capacitated Arc Routing Problem (CARP), which involves servicing a set E_R of required edges (or arcs) through a homogeneous fleet of vehicles in a connected undirected graph $G = (V, E)$, with the vertex set V and the edge set E where $E_R \subseteq E$. Each edge $(i, j) \in E_R$ has a demand d_{ij} and the total demand for any route cannot exceed the vehicle capacity Q. Additionally, the size of the vehicle fleet $|K|$ can be limited or unlimited. A CARP solution must design a set of least-cost routes for vehicles to serve all required edges to the following constraints:

- Each route starts and ends at the depot.
- Each task is serviced to exactly once by a vehicle.
- The total demand collected by each vehicle cannot exceed its capacity.

Recently, most non-exact algorithms that have emerged for CARP are based on classical and hybrid metaheuristic frameworks. Among the most popular metaheuristic methods are the Tabu Searches proposed by Hertz et al. (2000), and Brandão and Eglese (2008), Variable Neighborhood Descent developed by Hertz and Mittaz (2001), and a Greedy Random Adaptive Search Procedure proposed by Usberti et al. (2013). Other classes of hybrid and genetic metaheuristics were also developed in Lacomme et al. (2004), Wang et al. (2015), and Vidal (2017).

2.2
Time-dependent Routing Models

2.2.1
Travel time and speed models

Ichoua et al. (2003) introduced a model for Time-Dependent Travel Times where they divided the planning horizon $[0, H]$ into h time-intervals. In each division, the vehicle's speed is different, and it can change speed when it

crosses a boundary between two consecutive intervals. Furthermore, the FIFO property is satisfied, where it guarantees that if a vehicle leaves node i for node j at a given time, any identical vehicle leaving node i for node j at a later time will arrive later at node j (ICHOUA et al., 2003). Figure 2.1 illustrates the travel speed function in an arc.

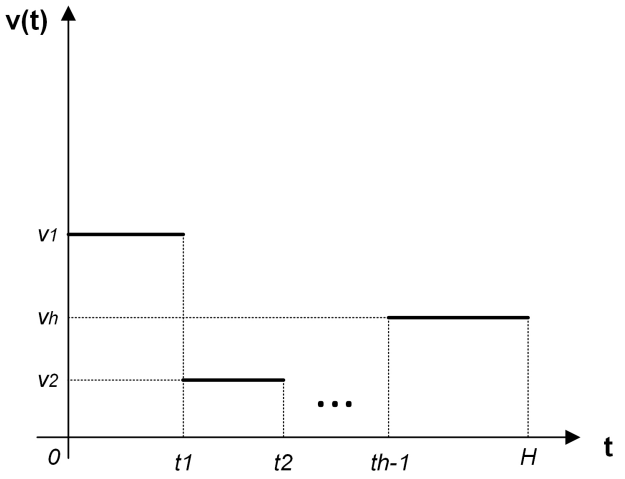


Figure 2.1: stepwise Speed function on an arc.

In addition, Algorithms 1 and 2 are presented in Ichoua et al. (2003) and Vidal et al. (2021), which allow calculating arrival time (Φ_{ij}) and departure time (Φ_{ij}^{-1}) between vertices i and j.

In Algorithm 1, the vehicle leaves node i at t_i and passes the edge (i, j) with distance d_{ij}. First, we initialize the current time t ($t = t_i$) with travel speed $v_{ij}(t^+)$ associated with piece k, where $v_{ij}(t^-) = \lim_{x \to t^+} v_{ij}(x)$. In each iteration, we seek to determine if the distance d_{ij} is completed without the arrival time (t_j) passing the limits of the current piece (i.e., $t_j > t_k$).

In Algorithm 2, the vehicle arrives at t_j with speed $v_{ij}(t^-)$. The iterative process is inverse to Algorithm 1. First, we initialize the current time t ($t = t_j$) with travel speed $v_{ij}(t^+)$ associated with piece k, where $v_{ij}(t^-) = \lim_{x \to t^-} v_{ij}(x)$. In each iteration, we seek to determine if the distance d_{ij} is completed without the departure time (t_i) passing the limits of the current piece (i.e., $t_i < t_k$).

Furthermore, the complexities of both algorithms, in the worst case, grow linearly in $O(h_{ij})$ with the number of pieces (h_{ij}) in the speed functions (VIDAL et al., 2021).

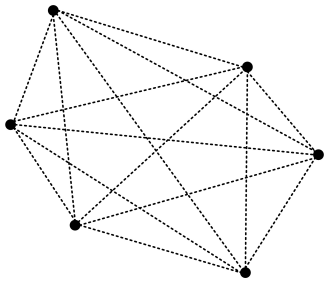
Algorithm 1: $\Phi_{ij}(t_i)$

$t \leftarrow t_i$
$d \leftarrow d_{ij}$
$k \leftarrow arg\ min\{x \mid t_x > t_i\}$
$t_j \leftarrow t + d/v_{ij}(t^+)$
while $t_j > t_k$ **do**
    $d \leftarrow d - v_{ij}(t^+) \times (t_k - t)$
    $t \leftarrow t_k$
    $k \leftarrow k + 1$
    $t_j \leftarrow t + d/v_{ij}(t^+)$
return t_j

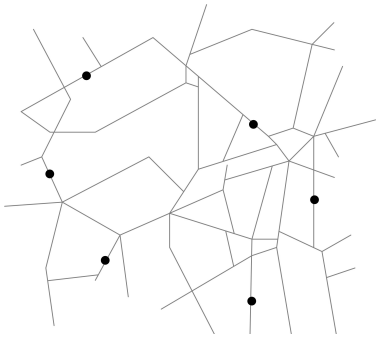
Algorithm 2: $\Phi_{ij}^{-1}(t_j)$

$t \leftarrow t_j$
$d \leftarrow d_{ij}$
$k \leftarrow arg\ max\{x \mid t_x < t_j\}$
$t_i \leftarrow t - d/v_{ij}(t^-)$
while $t_i < t_k$ **do**
    $d \leftarrow d - v_{ij}(t^-) \times (t - t_k)$
    $t \leftarrow t_k$
    $k \leftarrow k - 1$
    $t_i \leftarrow t - d/v_{ij}(t^-)$
return t_i

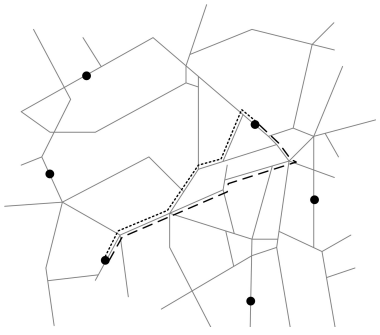
However, most TD-VRP studies use a complete graph representation of the network as shown in Figure 2.2a in which each vertex corresponds to a service or depot location (VIDAL et al., 2021). In Ichoua et al. (2003), the stepwise constant vehicle speed functions are associated with the arcs of the complete graph, assuming the existence of a unique Quickest Path between any two locations. In practice, time-dependent travel times are specific to each street in an urban network as shown in Figure 2.2b. In this representation, we eliminate the assumption of a unique Quickest Path, since in reality the Quickest Path depends on time and can be different paths as shown in figure 2.2c, so a road network level representation offers a realistic model.



(a) Complete graph



(b) Road Network level



(c) Quickest Paths

Figure 2.2: Time-Dependent Quickest Path

2.2.2 Time-Dependent Arc Routing

Tagmouti et al. (2007) worked on an arc routing problem with capacity constraints and time-dependent service costs with winter grit applications. In this problem, the service cost functions are piecewise linear, and the times to pass and serve an arc are constant. The authors used a transformation method for ARP to a VRP, which was then solved using a column generation approach. Subsequently, Tagmouti et al. (2010) proposed a Variable Neighborhood Descent heuristic to solve the same routing problem in larger instances.

Black et al. (2013) introduced a new problem called the Time-Dependent Prize Collection Arc Routing Problem (TD-PARP), which arises whenever one has to choose between several pickups and deliveries of full truckloads on a road network where the travel times change with the time of day. In addition, it performs mathematical formulations and solution methods for time-dependent travel times of single-vehicle cases, but these methods are still limited. Black et al. (2015) developed a new variant of the TD-PARP with multi vehicles (TD-MPARP). They proposed two metaheuristic algorithms, one based on Variable Neighborhood Search and another based on Tabu Search called LANTIME (Black et al. 2013, Black et al. 2015).

Jin et al. (2020) introduced a Time-dependent Penalty Cost Arc Routing Problem with practical application in garbage collection service. This problem considers minimizing the cost of service, the cost of the trip and the cost of the penalty. The penalty costs depend on the parking pattern and the period of service in each arc. When a truck must service a street, all cars parked on the side of the road must move away. Therefore, a roadside traffic sign is placed to alert drivers of the no parking period. However, the model keeps travel and service times constant on each arc.

Vidal et al. (2021) conducted extensive studies on the network-level Time-Dependent Capacitated Arc Routing Problem, where the simplification of time-dependent travel times of current models was questioned. Therefore, they derived a closed-form representation for the arrival time functions based on the definition of the travel speed function given by Ichoua et al. (2003) and proposed a dynamic programming method to compute the time-dependent Quickest Path with continuous speed without approximation or discretization. In addition, they developed a preprocessing approach for travel and service time queries. An exact Branch-Cut-and-Price Algorithm and a metaheuristic based on Hybrid Genetic Search were proposed.

2.3 Time-Dependent Capacitated Arc Routing Problem with Travel Time

Let $G = (V, E, A)$ be an indirect and incomplete graph in which V is the set of vertices, E is the set of edges, and A is the set of arcs. The node $0 \in V$ represents the depot, where a fleet with n vehicles with charging capacity Q are available at time 0. $E_R \subseteq E$ and $A_R \subseteq A$ represent edges and arcs that require service. For each service $u \in E_R \cup A_R$, a non-negative demand q_u is assigned. Each edge $u \in E_R$ can be served in one of its two possible orientations, called *modes* in Vidal (2017), unlike arcs $u \in A_R$ where its orientation is fixed. To formalize this difference, we associate to each service u a mode set M_u, where $M_u = \{1, 2\}$ if $u \in E_R$ and $M_u = \{1\}$ otherwise. Each service must be performed once and by a single vehicle, but any edge or arc of $E \cup A$ can be *deadheaded* multiple times while traveling on the network. Finally, arcs and edges are characterized by time-dependent travel and service speed functions. Also, the FIFO property is respected when traveling and servicing (i.e. starting later does not allow arriving earlier), and waiting is never profitable.

TD-CARP is defined over a planning horizon $[0, H]$. They associate a distance d_{ij} with a stepwise speed function $v_{ij} : [0, H] \rightarrow \mathbb{R}^+$ with h_{ij} pieces representing the travel speed on this link as a function of time, i.e., the distance traveled per time unit (VIDAL et al., 2021). The vehicle speed can change when it crosses the boundary between two consecutive periods. Furthermore, FIFO property is respected when traveling and servicing (i.e., starting later does not allow arriving earlier), and waiting is never profitable (VIDAL et al., 2021). Finally, TD-CARP seeks to design routes in order to minimize the sum of the route durations subject to the following properties:

- Each route starts and ends at the depot.
- Each required edge is serviced to exactly once by a vehicle.
- The total demand served by each vehicle cannot exceed its capacity Q.
- The duration of each route does not exceed a maximum value of H.

2.3.1 Continuous Travel Time Functions

Section 2.2.1 presents the arrival and departure time algorithms used in the time-dependent routing literature. Since these algorithms have a linear computational time (due to their iterative process), the queries of these values result in high computational overhead. To avoid this, Vidal et al. (2021) show that the Arrival Time function (Φ_{ij}) are piecewise linear, continuous,

monotonic and can be represented in closed-form. To do this, they propose a preprocessing method for Φ_{ij}. This procedure, described in Algorithm 3, is performed once for each arc and oriented edge $(i, j) \in E \cup A$ in a preprocessing phase prior to the Quickest Path algorithm.

First, we must define the breakpoints of the Closed-form Arrival Time function ($\Phi_{ij}(t)$). We establish that $t_1, ..., t_{h_{ij}-1} \in [0, H]$ are the breakpoints of the stepwise Speed function (v_{ij}) with h_{ij} pieces. If we determine the arrival time of these points, not all of them meet the upper planning horizon constraint ($\Phi(t) \leq H$), so $k = argmax\{x \mid t_x \leq \Phi_{ij}^{-1}(H)\}$ is defined to represent the farthest breakpoint that satisfies this condition. It defines the first set of time departure breakpoints $\{t_1, ..., t_k\}$. Furthermore, we obtain a set of additional breakpoints when we consider the v_{ij} breakpoints as arrival times. However, not all of them meet the lower planning horizon constraint ($\Phi^{-1}(t) \geq 0$), so $l = argmin\{x \mid t_x \geq \Phi_{ij}(0)\}$ is defined to represent the minimum breakpoint that satisfies this condition. Thus, the set of breakpoints $\{\Phi_{ij}^{-1}(t_l), ..., \Phi_{ij}^{-1}(t_{h_{ij}-1})\}$ is generated. Therefore, the function Φ_{ij} has up to $2 \times (h_{ij} - 1)$ breakpoints with values $t_1, ..., t_k, \Phi_{ij}^{-1}(t_l), ..., \Phi_{ij}^{-1}(t_{h_{ij}-1})$.

Algorithm 3: Closed-form construction of $\Phi_{ij}(t)$

1 $k \leftarrow arg\ max\{x \mid t_x \leq \Phi_{ij}^{-1}(H)\}$
2 $l \leftarrow arg\ min\{x \mid t_x \geq \Phi_{ij}(0)\}$
3 $A_{BP} = \varnothing$
4 $A_{BP} \leftarrow (0, \Phi_{ij}(0))$
5 **for** $x \in \{1, ..., k\}$ **do**
6 $\quad A_{BP} \leftarrow (t_x, \Phi_{ij}(t_x))$
7 **for** $x \in \{l, ..., h_{ij} - 1\}$ **do**
8 $\quad A_{BP} \leftarrow (\Phi_{ij}^{-1}(t_x), t_x)$
9 $A_{BP} \leftarrow (\Phi_{ij}^{-1}(H), H)$
10 *Sort*(A_{BP})
11 *Remove Duplicates*(A_{BP})
12 $A_{PIECES} = \varnothing$
13 **for** $x \in \{l, ..., Size(A_{BP}) - 1\}$ **do**
14 $\quad A_{PIECES} \leftarrow (A_{BP}[x], A_{BP}[x + 1])$
15 **return** A_{PIECES}

Source: Vidal et al. 2021

Algorithm 3 starts by determining the values of k and l (Lines 1-2). The first (Lines 5 - 7) and second (Lines 8 - 9) iteration steps compute the breakpoints along with their arrival times. The $\Phi_{ij}(t)$ and $\Phi_{ij}^{-1}(t)$ values within Algorithm 3 are obtained through iterative Algorithms 1 and 2 explained in Section 2.2.1. Each breakpoint and its respective arrival time is stored in tuples $(t,\ \Phi(t))$

in A_{BP}. Once we finish this process, all tuples must be sorted and duplicates removed. Finally, a linear *piece* will be represented by two tuples and stored inside A_{PIECES}.

Thus, the functions $\Phi_{ij}(t)$ are represented as piecewise linear continuous in closed-form, where one piece is composed of two continuous breakpoints as shown in Figure 2.3.

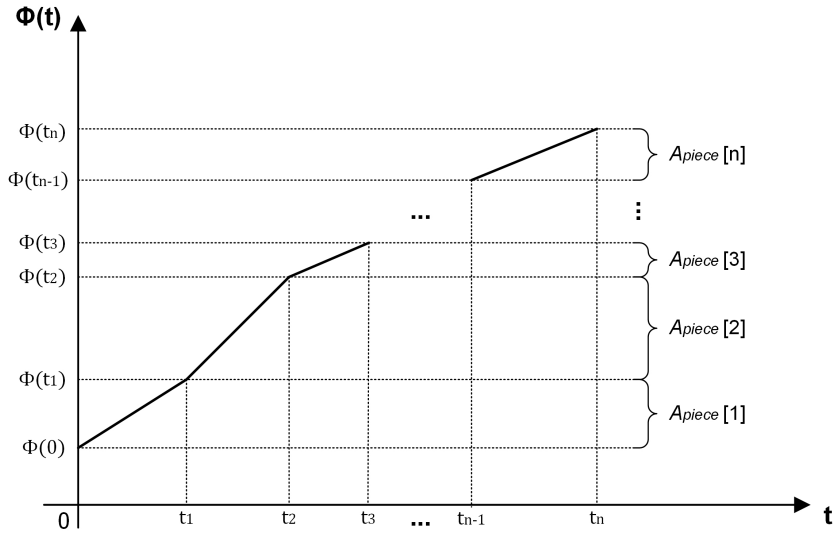


Figure 2.3: Departure Time x Arrival Time

Therefore, queries for the value of $\Phi_{ij}(t)$ are performed in computational time $O(1)$ if the index piece is known and $O(log\ h_{ij})$ otherwise by binary search (VIDAL et al., 2021).

2.3.2 Quickest Path Algorithm

Vidal et al. (2021) proposed a continuous preprocessing approach with the aim of reducing the computational overhead of approaches based on iterative travel time queries, as well as the memory overhead and imprecision of approaches based on time discretization. This approach consists of computing closed-form representations of the arrival time function $\Psi_{ij}(t)$ of the Quickest Path between each origin i to each destination j at any departure time t. Vidal et al. (2021) used a variant of the Bellman-Ford algorithm, presented in Algorithm 4, in which piecewise continuous linear functions are maintained in closed-form and updated using *compound* and *lower envelope* operations.

Algorithm 4: Quickest path algorithm from i for all starting times

for $j \in V$ **do**
  $\Psi'_{ij} = \Psi_{ij} \begin{cases} id & i = j \\ \infty & otherwise \end{cases}$
$L \leftarrow \{i\}$
while $L \neq \varnothing$ **do**
  for $(x, y) \in E : x \in L$ **do**
    $\Psi'_{iy} \leftarrow$ *LowerEnvelope*$(\Psi'_{iy}, \Phi_{xy} \circ \Psi_{ix})$
  $L \leftarrow \varnothing$
  for $y \in V$ **do**
    if $\Psi_{iy} \neq \Psi'_{iy}$ **then**
      $L \leftarrow L \cup y$
      $\Psi_{iy} \leftarrow \Psi'_{iy}$
return Ψ

Source: Vidal et al. 2021

Algorithm 4 starts by initializing the values of the Ψ functions for every pair (i, j), where "id" represents the identity function ($\Psi_{ij}(t) = t$). Due to the continuous representation of the functions $\Psi_{ij}(t)$, each usual label comparison is replaced by a lower envelope operation (HERSHBERGER, 1989; VIDAL et al., 2021). This operation is represented in Figure 2.4, where the objective is to obtain the set of lower pieces considering intersection points. In addition, the "$\circ$" operation, called compound in Vidal et al. (2021), corresponds to the composition of two closed-form PL representations functions. Equation (2-1) describes the compound on the arrival time functions. This algorithm ends until no pair (i, j), $\forall j \in V$, succeeds in modifying the current lower envelope (i.e., $\Psi_{ij} = \Psi'_{ij}$, $\forall j \in V$).

$$\Phi_{xy} \circ \Psi_{ix}(t) = \Phi_{xy}(\Psi_{ix}(t)) \tag{2-1}$$

As a result of this algorithm, we obtain the continuous PL functions Ψ_{ij} that represent the value of the Quickest paths from any vertex i to any vertex j at any departure time t. Similar to the closed-form PL functions of Φ_{ij}, the Quickest Path arrival time query has logarithmic time complexity based on the number of breakpoints. Furthermore, as shown in Figure 2.4, new breakpoints are generated when intersections occur in the lower envelope.

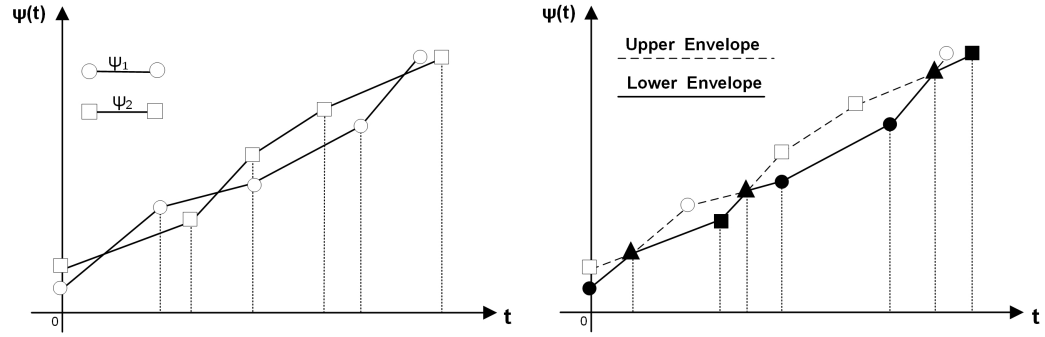


Figure 2.4: Lower Envelope representation and intersection points.

While obtaining the lower envelope, we observe that the Quickest Paths can differ whenever an intersection appears. So we can say that each piece of the lower envelope has its path that can be different from other pieces of the same PL function. Therefore, the assumption or simplification of a single Quickest path between two vertices that numerous works in the literature have used is eliminated.

2.4 Electric Vehicle with Energy Consumption on Routing Models

Electric vehicles promise to reduce transportation costs and the effects of pollution compared to fossil fuel-based engines (JING et al., 2016). However, the limited driving range due to the electric battery, the long charging times, and the limited availability of charging facilities make charging operations a more complex problem than refueling operations for conventional vehicles. In recent years, several researchers from the Computer Science, Artificial Intelligence, and Operations Research communities have developed optimization, simulation, and machine learning approaches. The aim is to generate efficient and sustainable routing plans for hybrid fleets, including electric and internal combustion engine vehicles (MARTINS et al., 2021). Figure 2.5 shows a representative scheme of the influence of electric vehicles on routing problems and their variants.

Most studies on EVs in routing problems consider electric battery capacity, energy consumption rate, recharging rate, and recharging policies under different assumptions. In this work, we will focus on Energy Consumption Rate because, in an urban environment, the routes tend to be shorter. It means the problems related to charging times and infrastructure can be avoided (PELLETIER et al., 2019). Typically, companies that use EVs to distribute goods charge them on company property overnight and rarely use public charging stations (NABEREZHNYKH et al., 2012; MORGANTI; BROWNE, 2018).

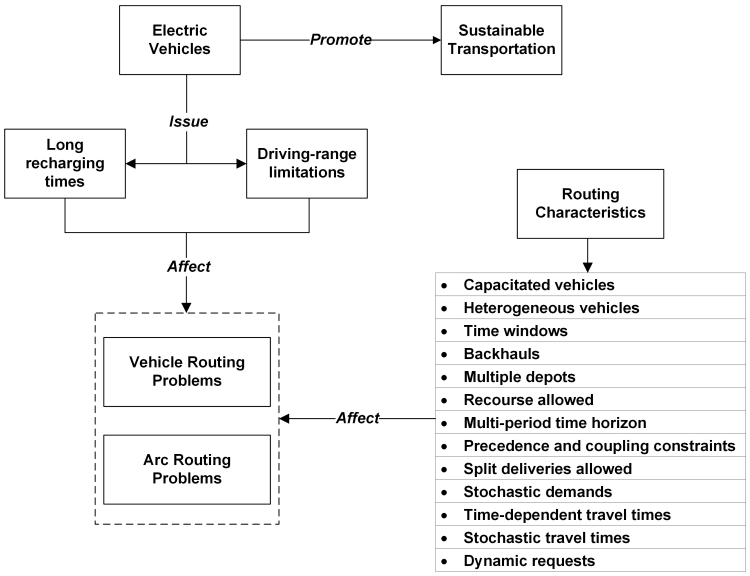


Figure 2.5: Conceptual scheme of the Routing Problems environment with EVs adapted from Martins et al. (2021)

2.4.1 Energy consumption models

The energy consumption calculation of EVs is one of the essential topics within the E-VRPs and E-ARPs, since it directly affects the route plans, the calculation times, the data required, and, most importantly, the ability to execute it in practice (KUCUKOGLU et al., 2021). In order to obtain more realistic and accurate results, the calculation of the energy consumption must take into account several aerodynamic and physical factors, such as the road conditions (roughness, inclination, impact, and others), the technical characteristics of the vehicle (front surface, battery capacity, weight, and others), vehicle load and environmental conditions (air density). However, each additional factor to consider makes energy consumption calculations more complex. In the existing studies, the energy consumption calculations can be categorized into three groups (KUCUKOGLU et al., 2021).

Linear deterministic functions. These functions determine the energy consumption using a constant consumption rate for a given metric (e.g., distance or time).

Non-linear deterministic functions: These functions are considered to obtain more realistic results for the energy consumption of electric vehicles (MURAKAMI, 2017; KUCUKOGLU et al., 2021). Goeke and Schneider (2015) introduced a comprehensive approach based on the longitudinal dynamic model to determine the energy consumption in which the factors of air

resistance force (F_a), resistance force to rolling (F_r), and gravitational force (F_g), to then be converted to mechanical power (P_M) using the Equation (2-2).

$$\begin{aligned} P_M &= \Big(F_{Air} + F_{Rolling} + F_{Gravity}\Big).v \\ P_M &= \left(m.a + \frac{\rho A_F.C_w.v^2}{2} + m.g.sin(\alpha) + C_r.m.g.cos(\alpha)\right).v \end{aligned} \tag{2-2}$$

In Equation (2-2), m denotes the weight, a the acceleration, C_w the aerodynamic drag coefficient, ρ the air density, A_F the frontal area of the EV, v the speed, g the gravitational constant, C_r the rolling friction coefficient, and α the gradient angle (KUCUKOGLU et al., 2021). Researchers mainly prefer the linear deterministic model to formulate the E-VRP as a mixed integer programming (MIP) model. On the other hand, non-linear deterministic functions are incorporated in many works with metaheuristic approaches to simulate energy consumption more realistically (KUCUKOGLU et al., 2021).

Goeke and Schneider (2015) proposed an Adaptive Large Neighborhood Search (ALNS) algorithm that uses this non-linear formulation to determine amounts of energy consumption. Similarly, S. Zhang et al. (2018) used an Ant Colony Optimization (ACO) metaheuristic approach considering the same non-linear function. More recent works, such as Wang et al. (2020), propose a Variable Neighborhood Search (VNS) algorithm for the Time-dependent Electric Vehicle Routing Problem with Time Windows, and J. Li et al. (2020) propose an Adaptive Genetic Algorithm based on scaling optimization and neighborhood search for the Battery-Swapped Electric Vehicle Routing Problem considering energy consumption. However, these works do not consider the consumption of auxiliary components such as air conditioning, radio, and lights. In the literature related to energy consumption, models such as Asamer et al. (2016) involve the consumption of auxiliary components, motor efficiency, and other external factors (which are used more in stochastic models), which allows a more realistic model of consumption.

Stochastic function: Some recent publications present variants of E-VRP that consider stochastic variables associated with more realistic scenarios. Zhang et al. (2020) propose an E-VRP with time windows and charging stations with a partial charge policy. The uncertainty is modeled by fuzzy dummy numbers for service times, battery energy consumption, and travel times. An initial scenario is created, and it generates additional random values from it. A hybrid algorithm, combining an ALNS with a variable neighborhood descent (VND), is proposed as a resolution approach. Also, Zhao et al. (2020) and Florio et al. (2021) propose E-VRP with time-varying

traffic conditions. In Zhao et al. (2020), an ACO algorithm is used to plan the routing of fresh products in the urban cold chain, which is affected by the type of road, the requirements of the client's time windows, the freshness of the products and in-route loading queues. In Florio et al. (2021), they focus on an electric vehicle routing problem with stochastic and time-dependent travel times where battery recharging is not allowed along the routes. They introduced a new method for generating network-consistent (time-correlated and space-correlated) time-dependent speed scenarios.

2.5 Non-Linear Energy Consumption Function

In this section, we will explain the non-linear energy consumption rate model employed in this work based on the models proposed in Asamer et al. (2016) and Fernandez et al. (2020).

2.5.1 Longitudinal Dynamic Model

The primary vehicle longitudinal dynamic model is based on Newton's second law, where the electric motor's traction force causes the vehicle's movement. However, more forces are involved in these dynamics, such as aerodynamic resistance, grade resistance, rolling resistance, and others. (ASAMER et al., 2016). Figure 2.6 shows the forces involved in the dynamic system where m is mass, and g is gravity.

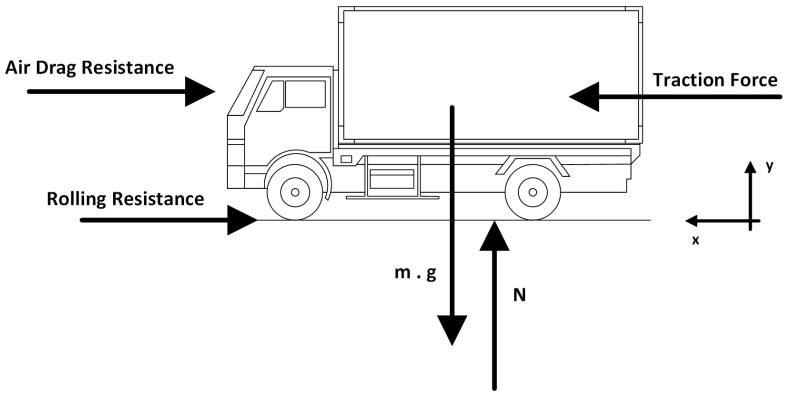


Figure 2.6: Forces acting on a moving vehicle.

In Goeke and Schneider, and Asamer et al. (2015, 2016), the gravity or grade resistance was included. We will use the assumption of the inexistence of inclination in the road (Degree α equal to zero). Therefore, we describe the composition of the resistance force in Equation (2-3).

$$F_{Resistance} = F_{Rolling} + F_{Air} \tag{2-3}$$

In Equation 2-3, Rolling Resistance ($F_{Rolling}$) is a forward displaced ground reaction force due primarily to the hysteresis of the tire materials (EHSANI et al., 2018). This force has a magnitude of $F_{Rolling} = C_r.N$ where N is the normal acting weight ($N = m.g$ applying newton's second law on the y-axis), and C_r is called the rolling resistance coefficient. Aerodynamic Drag (F_{Air}) occurs when a vehicle traveling at a particular speed in air encounters a force that resists its motion, known as aerodynamic drag (EHSANI et al., 2018). The Aerodynamic Drag Force has a magnitude of $F_{Air} = \frac{\rho.A_F.C_w.v^2}{2}$. It is based on the vehicle speed v, the vehicle frontal area A_F, the air density ρ, and the coefficient of Aerodynamic Drag C_w that characterizes the shape of the vehicle body (EHSANI et al., 2018). We obtained the Equation (2-4) by applying Newton's second law on the x-axis.

$$F_{Traction} - F_{Resistance} = m.\frac{\partial v}{\partial t} \tag{2-4}$$

If we consider the constant speed, it establishes that the derivative of the speed concerning time is equal to zero ($\frac{\partial v}{\partial t} = 0$). Therefore, the Traction Force is equal to the Resistance Force, as shown in Equation (2-5).

$$\begin{aligned} F_{Traction} &= F_{Resistance} \\ F_{Traction} &= F_{Rolling} + F_{Air} \end{aligned} \tag{2-5}$$

2.5.2 Non-linear Energy Consumption Rate

The battery's energy capacity is generally measured in kilowatt-hour (kWh) (EHSANI et al., 2018). Instantaneous Energy Consumption is an integration of power output at the battery terminals (EHSANI et al., 2018). This mechanical power needed to drive is the traction force multiplied by the vehicle's speed, as shown in Equation (2-6). Furthermore, the power losses in transmission, drive, and motor conversion power can be represented by an efficiency η (ASAMER et al., 2016). Therefore, we express the battery's instantaneous power output in Equation (2-6).

$$P_{out} = \frac{F_{Traction}}{\eta}.v + P_0 \tag{2-6}$$

The auxiliary components of the car also generate additional energy demand, which depends mainly on heating, air conditioning, lights, and radio. In Asamer et al. (2016), this value is denoted by P_0. Replacing the Traction Force by the Resistance Forces as indicated in Equation (2-5), we obtain Equation (2-7).

$$\begin{aligned} P_{out} &= (F_{Rolling} + F_{Air}).\frac{v}{\eta} + P_0 \\ P_{out} &= \left(\frac{C_r.m.g}{\eta}\right).v + \left(\frac{\rho.A_F.C_w}{2.\eta}\right).v^3 + P_0 \end{aligned} \tag{2-7}$$

This equation is an adaptation proposed by Fernandez et al. (2020) from the work of Goeke and Schneider (2015) and Asamer et al.(2016). The Energy Consumption Rate (ECR) is the instantaneous power of the vehicle (P_{out}). We call the speed-based Energy Consumption Rate function $ECR(v)$ when ECR is solely a function of the speed and set the other parameters to constants. Therefore, we define $ECR(v)$ in the equation (2-8). Figure 2.7 shows the graph of this function.

$$ECR(v) = P_{out} = \left(\frac{C_r.m.g}{\eta}\right).v + \left(\frac{\rho.A_F.C_w}{2.\eta}\right).v^3 + P_0 \tag{2-8}$$

If we maintain $ECR(v)$ for a travel time T in hours, we obtain the Energy Consumption Function based on the speed $e(v,T)$ expressed in kWh of said travel time as shown in Equation (2-9).

$$e(v,T) = ECR(v).T \tag{2-9}$$

We can use this function as long as the speed (v) and travel time (T) are known and the speed does not change during the travel time. In future sections, we will discuss how we can apply this function to the problem under study.

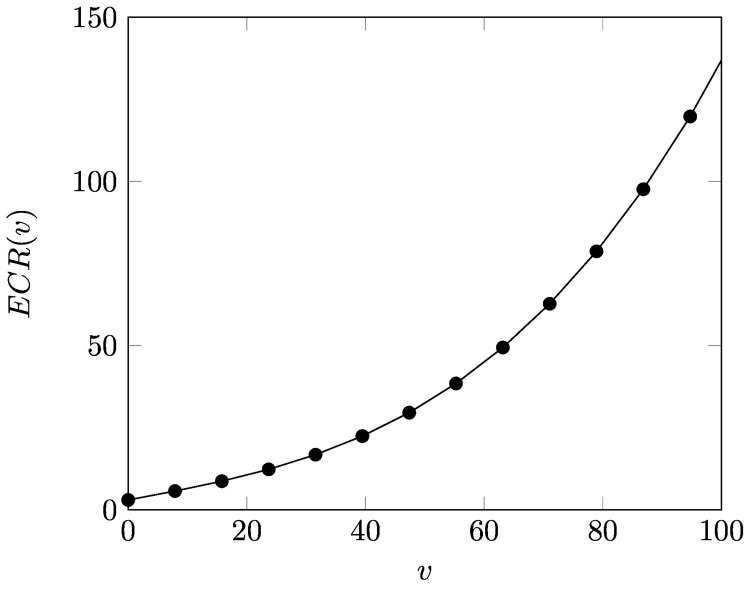


Figure 2.7: Energy Consumption Rate based on speed

3
Problem Definition

Let $G = (V, E)$ be an undirected graph in which V is the set of vertices and E is the set of edges. The vertex $0 \in V$ represents the depot, where a fleet with electric vehicles with load capacity Q in *units* and battery capacity C in kWh are available at time 0. $E_R \subseteq E$ represent edges that require service. For each service $u \in E_R$, a non-negative demand q_u is assigned. Each edge $u \in E_R$ can be served in one of its two possible orientations, called *modes* in Vidal (2017, 2021). To formalize this difference, we associate to each service u a mode set M_u, where $M_u = \{1, 2\}$. Each service must be performed once and by a single vehicle, but any edge E can be *deadheaded* multiple times while traveling on the network. Finally, edges are characterized by time-dependent travel and service speed functions based on the model introduced by Vidal et al. (2021). As in Vidal et al. (2021), we will maintain the FIFO property when traveling and providing service, and waiting will never be profitable, eliminating this possibility within our problem. Finally, E-TDCARP seeks to design routes in order to minimize the sum of the route durations subject to the following properties:

- Each vehicle leaves the depot at the start time and must return to the depot.
- The total demand served by each vehicle cannot exceed its load capacity Q.
- The total energy consumption by each vehicle cannot exceed its battery capacity C.
- The duration of each route does not exceed a set maximum value H.

3.1
Speed model

We will use the definition of Vidal et al. (2021) on network-level time-dependent travel times explained in Section 2.3. Given a planning horizon $[0, H]$, we assign each edge $(i, j) \in E$ a distance d_{ij} along with a time-dependent piecewise constant speed function $v_{ij}(t)$ for $t \in [0, H]$ with h_{ij} parts. Furthermore, each edge $(i, j) \in E_R$ is assigned a time-dependent piecewise constant serving speed function $\hat{v}_{ij}(t)$ for $t \in [0, H]$ with $\hat{h}_{ij}$ parts. Asymmetric edge speeds are allowed, so $v_{ij}(t) \neq v_{ji}(t)$ and $\hat{v}_{ij}(t) \neq \hat{v}_{ij}(t)$ are possible. Therefore, an edge $(i, j) \in E$ can have different speed functions and

breakpoints for each mode. Also, speeds can change when crossing breakpoints while the vehicle travels on an edge. Under these conditions, the FIFO property still holds.

3.2 Energy Consumption model

We will use a homogeneous EV fleet with weight m, frontal area A_F, and energy efficiency η. In Equations (2-8) and (2-9), we describe the non-linear Speed-dependent Energy Consumption Rate (ECR) in kilowatt (kW) and the Energy Consumption in kilowatt-hours (kWh) respectively. They are obtained from the longitudinal dynamic model described in Section 2.5. We will use this energy consumption rate model, maintaining the following premises and assumptions:

- Inexistence of inclination ($\alpha = 0$) in the edges (due to lack of this information in the network).
- Mechanical coefficients (C_w and C_r) and air density (ρ) are constants during travel on a given edge.

Since the speed model depends on the time in each edge and mode (i, j), we will rewrite Equation (2-9) based on the speed $v_{ij}(t_i)$ obtained from the stepwise Speed function v_{ij} at departure time t_i. Furthermore, the travel time (T) can be expressed in terms of the departure time (t_i) and the arrival time ($\Phi_{ij}(t_i)$) for each edge (i, j) (i.e., $T = \Phi_{ij}(t_i) - t_i$). Finally, we define the Energy Consumption $e(v_{ij}(t), T)$ under the symbol $e_{ij}(t_i)$ since it is now *time-dependent* for each edge (i, j). Thus we get the Equation (3-1).

$$e_{ij}(t_i) = ECR(v_{ij}(t_i)).(\Phi_{ij}(t_i) - t_i) \tag{3-1}$$

Due to the premise that *speed can change during the trip* on a given edge (i, j), it is not possible to use Equation (3-1) directly. Therefore, the energy consumption values in traveling $e_{ij}(t_i)$ and serving $\hat{e}_{ij}(t_i)$ in j when leaving i in t_i are calculated according to the Equations (3-2) and (3-3).

$$e_{ij}(t_i) = \int_{t_i}^{\Phi_{ij}(t_i)} ECR(v_{ij}(t)).dt \tag{3-2}$$

$$\hat{e}_{ij}(t_i) = \int_{t_i}^{\hat{\Phi}_{ij}(t_i)} ECR(\hat{v}_{ij}(t)).dt \tag{3-3}$$

In the next section, we will propose an iterative method for calculating energy consumption with speed changes using Equation (3-1) and a preprocessing method for the energy consumption function.

4
Methodology

In this section, we will introduce the stepwise energy consumption rate as a function of the time and their similarities with the stepwise speed function. Then, we will introduce our preprocessing method for the Energy Consumption functions of any edge based on the departure time. Also, we will explain our preprocessing method for the Energy Consumption functions corresponding to the Quickest Path between two vertices. Furthermore, we will explain the representation of our solution based on an indirect solution. In addition, the equations for calculating exact time and energy consumption will be introduced together with the procedure for obtaining lower bounds proposed in Vidal et al. (2021). It will be used in our proposed metaheuristic. Finally, we will explain the composition and design of our Iterated Local Search metaheuristic to produce high quality solutions to the studied problem.

4.1
Energy Consumption in Quickest Path

4.1.1
stepwise Energy Consumption Rate function

We define $ECR_{ij}(v_{ij}(t))$ as the speed-based Energy Consumption Rate function of the edge (i, j), where $v_{ij}(t)$ is its stepwise Speed function with h_{ij} number of pieces. A piece $x \in [1,\ h_{ij}]$ of $v_{ij}(t)$ contains a specified speed v_x. Since each piece has a unique speed, we can obtain the Energy Consumption Rate for each piece in $v_{ij}(t)$. Therefore, we can generate stepwise Energy Consumption Rate function based on the stepwise Speed function $v_{ij}(t)$ for any edge $(i, j) \in E$ as shown in Figure 4.1.

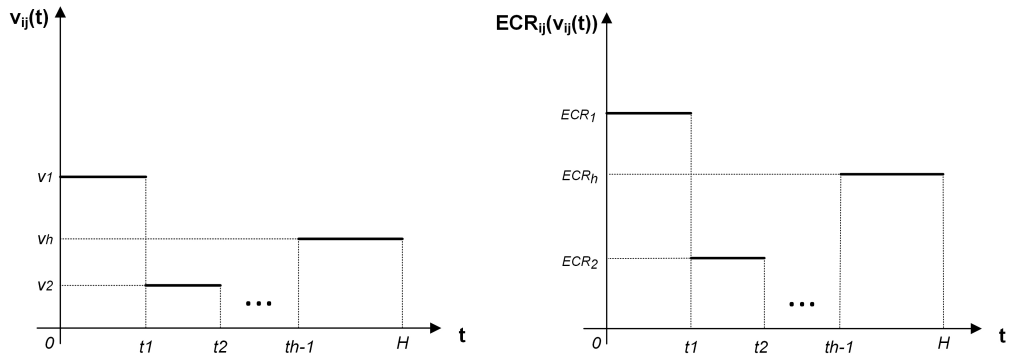


Figure 4.1: Comparison between speed and energy consumption rate profiles of the edge (i, j).

Finally, we are going to define the stepwise Energy Consumption Rate function $ECR_{ij}(v_{ij}(t))$ under the symbol $ECR_{ij}(t)$, due to now it will be just *time-dependent*. Thus, we can reformulate Equation (3-2) as shown in Equation (4-1).

$$e_{ij}(t_i) = \int_{t_i}^{\Phi_{ij}(t_i)} ECR_{ij}(t)\ dt \tag{4-1}$$

The calculation of the stepwise Energy Consumption Rate functions of all edges must be done in a phase prior to our Energy Consumption function preprocessing method, which we will explain in the following sections.

4.1.2 Iterative Algorithm for Energy Consumption Queries

Equation (3-1) was given in terms of the departure time t_i. We will use this equation by replacing $ECR_{ij}(v_{ij}(t_i))$ with $ECR_{ij}(t_i)$ as shown in Equation 4-2, since it now depends only on departure time (See Section 4.1.1).

$$e_{ij}(t_i) = ECR_{ij}(t_i)\ (\Phi_{ij}(t_i) - t_i) \tag{4-2}$$

We introduce Algorithm 5 to obtain the Energy Consumption (e_{ij}), where the input data is the departure time (t), the arrival time $\Phi_{ij}(t)$, and the function $ECR_{ij}(t)$.

Algorithm 5 starts by locating the departure time (piece k) and the arrival time (piece l) within the energy consumption rate function ECR_{ij}. We use Equation (4-2) to calculate the Energy consumption. Since the speed can change during the trajectory, we must determine the energy consumption for each piece involved (Lines 5 - 9).

Algorithm 5: $e_{ij}(t, \Phi_{ij}(t))$

```
1 t_d ← t
2 k ← arg min{x | t_x > t}
3 l ← arg min{x | t_x > Φ_ij(t)}
4 energy ← 0.0
5 for x ∈ k : l − 1 do
6     energy ← energy + ECR_ij(t_d^+) × (t_x − t_d)
7     t_d ← t_x
8 energy ← energy + ECR_ij(t_d^+) × (Φ_ij(t) − t_d)
9 return energy
```

Since energy consumption queries are as frequent as arrival time queries in metaheuristic approaches, the implementation of iterative algorithms is inefficient in terms of computational time. In the next section, we introduce an effective algorithm for preprocessing energy consumption e_{ij} in a closed-form piecewise linear function to avoid computational time overhead.

4.1.3 Continuous Energy Consumption Function

In this section, we will explain our preprocessing method for e_{ij} function as a closed-form piecewise linear function to avoid computational time overload. We first establish two essential properties of energy consumption functions and then describe our approach.

Property 1 *Functions e_{ij} are piecewise linear and continuous.*

Proof. This follows directly from Equation (4-1) given that functions ECR_{ij} are piecewise constant, positive and bounded. ■

Property 2 *Let $t_1, ..., t_{hij-1}$ be the breakpoints of function ECR_{ij} with h_{ij} number of pieces. Function e_{ij} has the same breakpoints as Φ_{ij} with values $t_1, ..., t_k, \Phi^{-1}(t_l), ..., \Phi^{-1}(t_{h_{ij}-1})$ where $k = argmax\{x \mid t_x \leq \Phi_{ij}^{-1}(H)\}$ and $l = argmin\{x \mid t_x \geq \Phi_{ij}(0)\}$.*

Proof. Define $E_{ij}(x) = \int_0^x ECR_{ij}(t)\ dt$. E_{ij} has breakpoints $t_1, ..., t_{h_{ij}-1}$. Based on Equation (4-1), we obtain Equation (4-3).

$$e_{ij}(t) = \int_0^{\Phi_{ij}(t)} ECR_{ij}(t)\ dt - \int_0^t ECR_{ij}(t)\ dt = E_{ij}(\Phi_{ij}(t)) - E_{ij}(t) \quad (4\text{-}3)$$

Function e_{ij} is PL as a difference between two PL functions. The first term $E_{ij}(\Phi_{ij}(t))$ is PL as a composition of two PL functions. The breakpoints can occur whenever t is a breakpoint of $\Phi_{ij}(t)$ (i.e., $t \in \{t_1, ..., t_k, \Phi_{ij}^{-1}(t_l), ..., \Phi_{ij}^{-1}(t_{h_{ij}-1})\}$) and whenever $\Phi_{ij}(t)$ is a breakpoint of E_{ij}. In the latter case, there exists s such that $\Phi_{ij}(t) = t_s$ (i.e., $t = \Phi_{ij}^{-1}(t_s),\ \forall t_s \in \{t_1, ..., t_{h_{ij}-1}\}$). However, not all t_s satisfy that $\Phi_{ij}^{-1}(t_s) \geq 0$. To deal with this, we define $l = argmin\{x \mid t_x \geq \Phi_{ij}(0)\}$, then $t = \Phi^{-1}(t_s),\ \forall t_s \in \{t_l, ..., t_{h_{ij}-1}\}$. Therefore, $t \in \{\Phi^{-1}(t_l), ..., \Phi^{-1}(t_{h_{ij}-1})\}$. This second group is included in the first group, so the breakpoints for $E(\Phi(t))$ is $\{t_1, ..., t_k, \Phi^{-1}(t_l), ..., \Phi^{-1}(t_{h_{ij}-1})\}$.

For the second term, we have $E_{ij}(t)$ with breakpoints $t_1, ..., t_{h_{ij}-1}$ and domain $[0, H] \to \mathbb{R}^+$. Since the function $E_{ij}(t)$ is the difference between two PL functions, the resulting domain will be the intersection of the domains of both functions ($E_{ij}(\Phi_{ij}(t))$ and $E_{ij}(t)$). Since the domain of $E_{ij}(t)$ is the entire planning horizon, the resulting domain will be the domain of the function $E_{ij}(\Phi_{ij}(t))$. Therefore, the breakpoints of the function $E_{ij}(t)$ are bounded to $\{t_1, ..., t_k\}$, since the function $E_{ij}(\Phi_{ij}(t))$ does not allows points that violate the condition that the arrival time exceeds the planning horizon limit ($\Phi(t) \leq H$), so there exists $k = argmax\{x \mid t_x \leq \Phi_{ij}^{-1}(H)\}$ (See breakpoints of $\Phi(t)$ in Section 2.3.1).

Finally, the breakpoints of e_{ij} is the product of the union between the breakpoints of both functions PL. It means that $t \in \left\{\{t_1, ..., t_k\} \cup \{t_1, ..., t_k, \Phi_{ij}^{-1}(t_l), ..., \Phi_{ij}^{-1}(t_{h_{ij}-1})\}\right\}$. Thus, e_{ij} breakpoints are $t \in t_1, ..., t_k, \Phi_{ij}^{-1}(t_l), ..., \Phi_{ij}^{-1}(t_{h_{ij}-1})$ identically to $\Phi(t)$. ■

To avoid the computational time overhead related to travel time queries, we will use the closed-form arrival time in traveling $\Phi_{ij}(t)$ and serving $\hat{\Phi}_{ij}(t)$ described in Section 2.3.1.

Based on Property 1, Property 2 and closed-form $\Phi(t)$, we propose Algorithm 6 to obtain the closed-form Energy Consumption function ($e_{ij}(t)$). We will store all the breakpoints of the energy consumption function within E_{BP} using the arrival time breakpoints A_{BP} (See Section 2.3.1). E_{BP} has the same number of breakpoints as A_{BP} as Property 2 shows. We calculate the energy consumption of the breakpoints, in Line 3, using Algorithm 5. Finally, we store the energy consumption pieces within E_{PIECES}. This procedure is done once for each edge $(i, j) \in E$ along with the closed-form $\Phi_{ij}(t)$. This process will also be performed for the service Energy Consumption $\hat{e}$.

Algorithm 6: Closed-form construction of $e_{ij}(t)$

```
1 E_BP = ∅
2 for (t_i, t_j) ∈ A_BP do
3     E_BP ← (t_i, e_ij(t_i, t_j))
4 E_PIECES = ∅
5 for x ∈ {l, ..., SIZE(E_BP) − 1} do
6     E_PIECES ← (E_BP[x], E_BP[x + 1])
7 return E_PIECES
```

Unlike the closed-form representation of $\Phi_{ij}(t)$ shown in Figure 4.2, the closed-form representation of $e_{ij}(t)$ is not strictly increasing because the FIFO property does not influence energy consumption as seen in Figure 4.3. It indicates that if the vehicle leaves earlier, it does not mean strictly less energy consumption.

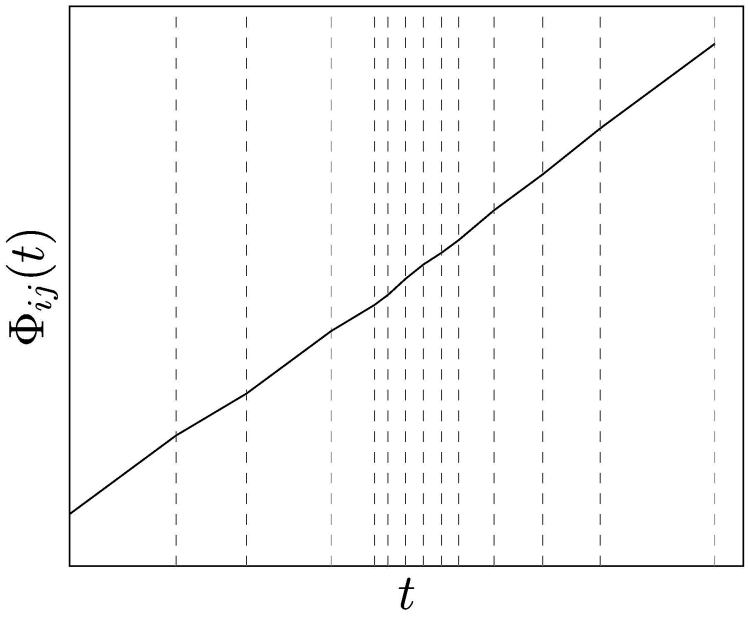


Figure 4.2: Closed-form construction of arrival time $\Phi_{ij}(t)$

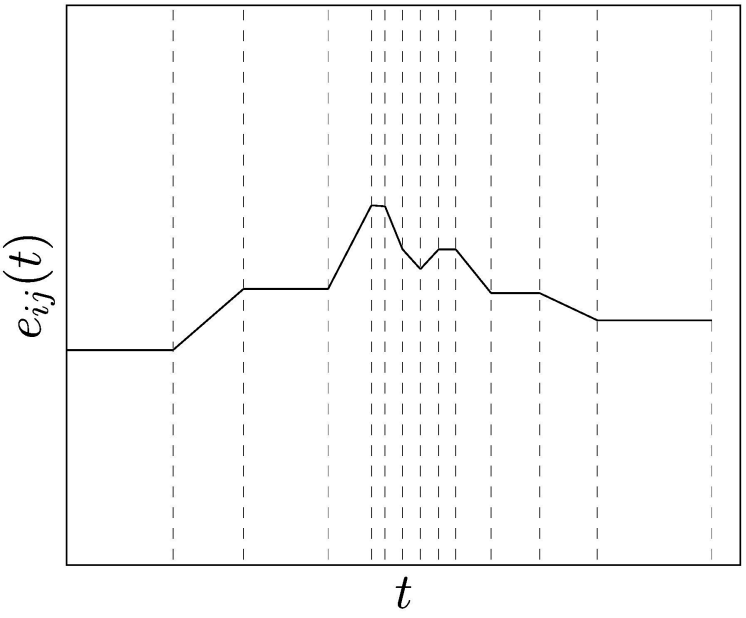


Figure 4.3: Closed-form construction of energy consumption $e_{ij}(t)$

4.1.4 Quickest Path Energy Consumption

Vidal et al. (2021) presented the Quickest Path method described in Section 2.3.2. We will use their proposed Algorithm 4 by adding a compound function for the composition of two piecewise linear functions of energy consumption. To do this, we apply Equation (4-4).

$$\begin{aligned} e_{ix} \circ e_{xy}(t) &= e_{ix}(t) + e_{xy}(\Phi_{ix}(t)) \\ e_{iy}(t) &= e_{ix}(t) + e_{xy} \circ \Phi_{ix}(t) \end{aligned} \tag{4-4}$$

Unlike the Arrival Time function, the first edge Energy Consumption function (i, x) does not generate the new departure time of the second (x, y). Therefore, the compound needs the Arrival Time function of the first edge, called $\Phi_{ix}(t)$, to establish it as the departure time of the second. Finally, the sum of the Energy Consumption of both edges will be the compound function. In the case of the Lower Envelope function, we will keep the criterion of *minimizing the arrival time* since we do not seek to minimize energy consumption. The piecewise linear functions of Energy Consumption will store according to the previous criterion. It means we will not save the Lower Envelope of the Energy Consumption function.

In Figure 4.4, we present the energy consumption function given a departure time t of the Quickest Path from i to j. The breakpoints of this function are not completely marked, unlike in Figures 4.2 and 4.3. The breakpoints marked by dotted lines represent *discontinuous breakpoints* resulting from the choice of the Quickest Path. Since the quickest path can vary in the time due to the lower envelope function, this will generate a discontinuous function for the energy consumption on the quickest path.

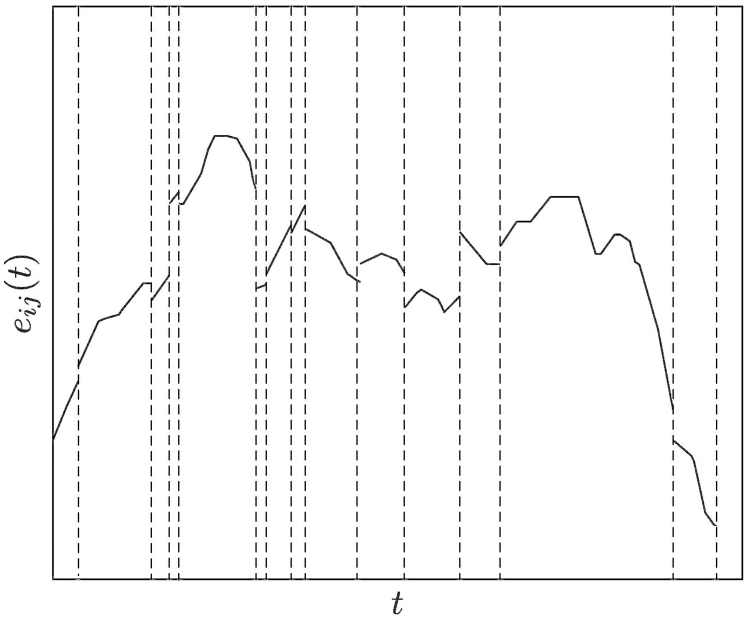


Figure 4.4: Exact Energy consumption $e_{ij}(t)$ of the Quickest Path from i to j given a departure time t.

Therefore, for an energy consumption query in a breakpoint t, it will be necessary choose for the quickest path with the lowest energy consumption. For this, we will use Equation (4-5).

$$e_{ij}(t) = \min\Big\{ \lim_{x \to t^-} e_{ij}(x),\ \lim_{x \to t^+} e_{ij}(x) \Big\} \tag{4-5}$$

4.2 Indirect Solution and Search Space

An indirect solution representation seeks to decompose the optimization problem into two subsets of decision variables. The first subset of decisions is made precisely, while the second subset will be made using an optimal decoder (exact or heuristic method) to complete the solution. This representation simplifies the solution approach by making it more structured and dramatically reduces the approximations inherent in heuristic search. The TD-CARP combines four classes of decisions (VIDAL et al., 2021):

1. ASSIGNMENT of services to vehicles
2. SEQUENCING of services within each route
3. MODE choices for the services
4. PATH choices between the services

Each of these decision sets contains several options that grow exponentially with the number of services. Now, when some of these decision sets are known (for example, Assignment and Sequence), the optimal options for the other sets can be derived through dynamic programming (VIDAL, 2017). We will use the same solution representation of Vidal et al. (2021) to optimize mode decisions efficiently and we will add Energy Consumption queries. They represent a solution as sequences of services (routes) without their *mode* information and systematically use a dynamic programming decoding algorithm to complete the solutions.

In our representation, the functions e and $\hat{e}$ illustrate the calculation of the energy consumption during the fastest route and the service, respectively. Due to time-dependent travel times and energy consumption, solution evaluations require propagation of arrival times, service completion times, and energy consumption in the auxiliary graph illustrated in Figure 4.5. In the indirect representation, this propagation in Vidal et al. (2021) is performed using *Bellman's algorithm* in topological order. We will use the same strategy incorporating the Quickest Path information Ψ (solid arrows in Figure 4.5), the service time functions $\hat{\Phi}$ (dotted arrows), thus as the energy consumption e and $\hat{e}$ for both types of arrows, respectively. We obtain these four pieces of information from a previous preprocessing step.

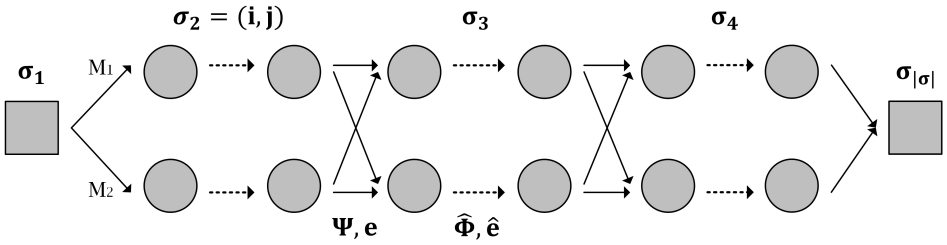


Figure 4.5: Indirect solution representation (only Assignment and Sequences).

In Vidal et al. (2021), for a route $\sigma = (\sigma(1), ..., \sigma(|\sigma|))$ described as a sequence of services starting and ending at the depot (such that $\sigma(1)$ and $\sigma(|\sigma|)$), the completion time $T^{\text{EXACT}}_{\sigma(i)}[l]$ of each service $\sigma(i)$ for each mode $l \in M_{\sigma(i)}$ can be calculated using Equation (4-6).

$$T^{\text{EXACT}}_{\sigma(i)}[l] = \begin{cases} 0 & i = 1 \\ \min\limits_{k \in M_{\sigma(i-1)}} \left\{ \hat{\Phi}^{l}_{\sigma(i)} \left(\Psi^{kl}_{\sigma(i-1)\sigma(i)} \left(T^{\text{EXACT}}_{\sigma(i-1)}[k] \right) \right) \right\} & \text{otherwise} \end{cases} \tag{4-6}$$

In Equation (4-6), $\Psi^{kl}_{ij}(t)$ represents the arrival time when the vehicle leaving the end of service i in mode k at time t towards the origin of service j in mode l, and $\hat{\Phi}^{l}_{i}(t)$ the completion time of service i in mode l starting at time t. The route duration is given by $T^{\text{EXACT}}_{\sigma(|\sigma|)}[1]$.

For the calculation of the exact total energy consumption $E^{\text{EXACT}}_{\sigma(i)}[l]$ until the time of termination of each service $\sigma(i)$ for each mode $l \in M_{\sigma(i)}$, we will use Equation (4-7).

$$E^{\text{EXACT}}_{\sigma(i)}[l] = \begin{cases} 0 & i = 1 \\ \min\limits_{k \in M_{\sigma(i-1)}} \left\{ E^{\text{EXACT}}_{\sigma(i-1)}[k] + e^{kl}_{\sigma(i-1)\sigma(i)}(T^{\text{EXACT}}_{\sigma(i-1)}[k]) + \hat{e}^{l}_{\sigma(i)}(\Psi^{kl}_{\sigma(i-1)\sigma(i)}(T^{\text{EXACT}}_{\sigma(i-1)}[k])) \right\} & \text{otherwise} \end{cases} \tag{4-7}$$

In Equation (4-7), $e^{kl}_{ij}(t)$ represents the energy consumption when the vehicle leaving the end of service i in mode k at time t towards the origin of service j in mode l, and $\hat{e}^{l}_{i}(t)$ the energy consumption of the completion time of service i in mode l starting at time t. The total energy consumption of the route is given by $E^{\text{EXACT}}_{\sigma(|\sigma|)}[1]$.

In the following sections, we will explain some methods to further mitigate the computational effort of route evaluations with motion filters based on route-cost lower bounds proposed by Vidal et al. (2021) and evaluations through preprocessing and concatenation (VIDAL, 2017).

4.2.1 Lower Bounds on move evaluations

In this section, we will explain the lower bounds proposed by Vidal et al. (2021) that will be implemented in our metaheuristic. A sequence of consecutive services σ is characterized by a lower bound $T^{LB}(\sigma)[k,l]$ on travel and service time over σ, starting the first service in mode k and finishing the last service in mode l, for each configuration k,l for $k \in M_{\sigma(1)}, l \in M_{\sigma(|\sigma|)}$, Vidal et al. (2021) state that the T^{LB} of a sequence σ containing a single service i is defined in Equation (4-8).

$$T^{LB}(\sigma)[k,l] = \begin{cases} \min\limits_{t\in[0,H]} \{\hat{\Phi}_i^k(t) - t\} & k = l \\ \infty & \text{otherwise} \end{cases} \tag{4-8}$$

We can collect these scalar values in the closed-form arrival time function on the services at an initial phase at our metaheuristic, so the query time for these values is constant. To evaluate more extensive sequences of services $\sigma_1 \oplus \sigma_2$ that arise from the concatenation ($\oplus$) of any two sequences σ_1 and σ_2, Vidal et al. (2021) propose Equation (4-9). This equation provides a lower bound on the shortest time needed to perform σ_1 followed by σ_2, starting and ending in modes k and l, respectively.

$$T^{LB}(\sigma_1 \oplus \sigma_2)[k,l] = \min_{x,y} \left\{ T^{LB}(\sigma_1)[k,x] + \min_{t\in[0,H]} \left\{ \Psi^{xy}_{\sigma_1(|\sigma_1|)\sigma_2(1)}(t) - t \right\} + T^{LB}(\sigma_2)[y,l] \right\} \tag{4-9}$$

It is important to note that the expression $\min\limits_{t\in[0,H]} \left\{ \Psi^{xy}_{\sigma_1(|\sigma_1|)\sigma_2(1)}(t) - t \right\}$ used in this equation represents the value of the quickest path at *the best starting time* t between the end extremity of service i in mode x and the starting extremity of service j in mode y. This value is obtained from the quickest path continuous functions Ψ, and ,in the same way as the function $\hat{\Phi}$, it is possible to collect these values to minimize query times. Furthermore, the values of T^{LB} between each pair of edges in a route can be calculated in a previous phase to the evaluation to reduce query times.

Finally, Vidal et al. (2021) strengthen the limit by pointing out that, in a path obtained by the concatenation of S sequences, the exact values of the arrival time (without any approximation) on the first sequence are known from the current established solution s. The resulting lower bound is Equation (4-10), where for a route composed of S sequences, the computation of this lower bound requires four evaluations of the travel time functions Ψ and a constant number of sums proportional to S. Therefore, the evaluation of the lower bound is an order of magnitude faster than the evaluation of the exact movement.

$$T^{LB+}(\sigma_1 \oplus ... \oplus \sigma_S) = \min_{x,y} \left\{ \Psi^{xy}_{\sigma_1(|\sigma_1|)\sigma_2(1)}(T^{\text{EXACT}}_{\sigma_1(|\sigma_1|)}[x]) + T^{LB}(\sigma_2 \oplus ... \oplus \sigma_S)[y,1] \right\} \quad (4\text{-}10)$$

Figure 4.6 presents an example of the use of T^{LB+}. Once the movements within the route have been made (step 2), we group the consecutive structures as in step 3. Then, we consult the exact time of the last service of the sub-sequence σ_1 and calculate the time to reach the first service of the sub-sequence σ_2 applying the quickest path function Ψ. We call the result of this operation $\Psi(T^{\text{EXACT}})$. The T^{LB} values of the sub-sequences σ_2, σ_3, and σ_S are calculated before the evaluations. The equation 4-9 will be used to calculate the Lower Bound concatenation operations ($\oplus_{LB} = T^{LB}(\sigma_i \oplus \sigma_j)$) in order (starting from the left side). Once the concatenation operations are done, we sum the result together with $\Psi(T^{\text{EXACT}})$. This procedure must be done for all possible configurations of $x, y \in \{M_1, M_2\}$. Finally, the min value of all configurations is determined.

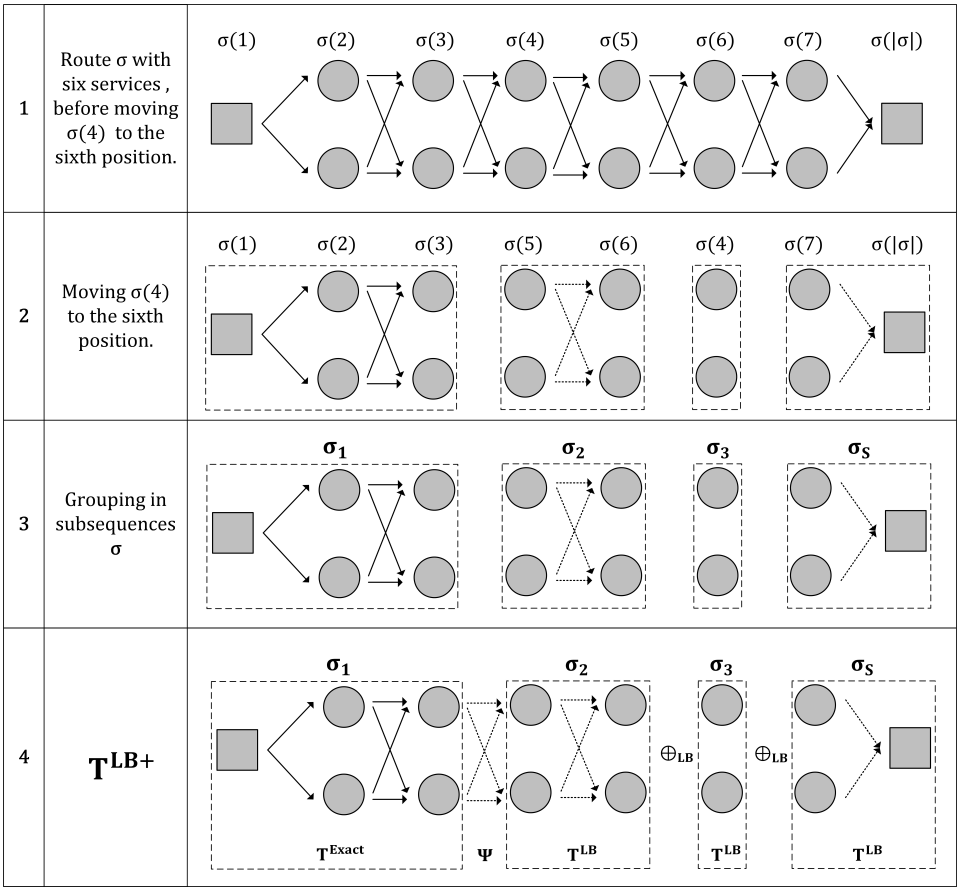


Figure 4.6: T^{LB+} example of execution.

4.3 Iterated Local Search Procedure

Since the CARP and all its extensions are $NP-hard$, the most effective methods for solving the problems are based on heuristic and metaheuristic approaches (CORBERÁN; PRINS, 2010).

For this work, we use an Iterated Local Search (ILS) metaheuristic because it provides an easy way to approach a complex problem without the risk of losing the conceptual and practical simplicity of a model design (LOURENÇO et al., 2003). In Algorithm 7, we present the structure of the proposed metaheuristic based on Iterative Local Search. The procedure begins with an *initial solution* obtained from a constructive heuristic, and the result is inserted into the *local search*, obtaining the local minimum S^*. The algorithm store the solution obtained from the local search in the variable *best*. We start the iterative process by performing *perturbations* on the current solution S^*. Then, it is again inserted in the local search obtaining the local minimum. To finish the iterative process, we execute the *acceptance criterion* to determine if the solution obtained from the local search is the best and if it is the candidate solution to carry out the following iterative process. Also, the use of *penalties* is allowed in our model, which allows infeasible solutions at an additional cost and helps explore new search spaces. If we find an infeasible solution in the end process, we return the best infeasible solution. Unlike the classic ILS, we propose a certain number of *restarts* of the iterative process that allows restoring the values of the penalties used and the parameters used for our acceptance criteria. It allows diversifying the exploration of the search space.

Algorithm 7: Iterated Local Search

```
1  s_0 ← Initial Solution()
2  s* ← Local Search(s_0)
3  Best ← s*
4  r = 1
5  while r ≤ Restarts do
6      n = 1
7      Restart Penalties()
8      Restart Acceptance Criteria Parameters()
9      while n ≤ Iterations do
10         s' ← Perturbation(s*)
11         s*' ← Local Search(s')
12         s*, Best ← Acceptance Criteria (s*, s*', Best)
13         Update Penalties()
14         n = n + 1
15     r = r + 1
16 return Best
```

Initial Solution: We built the initial solution using a Random Constructive Heuristic that starts by shuffling all the required edges into a list. An empty path is then opened, and start inserting each required edge. A new route is opened when the route cannot include another edge required by its load capacity. Also, on each include, it is checked whether it is possible to insert an edge on any previous routes before the current open route.

Adaptive Penalty: The cost of a feasible route is defined as the sum of the duration of the route plus the sum of penalties P based on load capacity, battery capacity, and planning horizon limits (time). It is shown in Equation 4-11.

$$C(\sigma) = T^{\text{EXACT}}_{\sigma_{|\sigma|}}[1] + \left(P_{Time}(T^{\text{EXACT}}_{\sigma_{|\sigma|}}[1]) + P_{Load}(Q[\sigma_{|\sigma|}]) + P_{Battery}(E^{\text{EXACT}}_{\sigma_{|\sigma|}}[1])\right) \tag{4-11}$$

Each penalty type $i \in \{\text{load, battery, time}\}$ has a penalty coefficient called p_i that multiplies the difference between the original value i of the route and the restriction or limit of i. It is shown in Equation 4-12.

$$P_i\ (Value) = max\Big\{Value - Limit_i,\ 0\Big\} \times p_i \tag{4-12}$$

These adaptive penalties need five parameters which are p_i^{base}, p_i^{min} and p_i^{max}, which are the base, minimum and maximum value of the penalty i, respectively, and δ_1 and δ_2, which are the reduction coefficient and the increase coefficient, respectively. The penalties start with their base value. In each iteration of the ILS, we verify if the solutions obtained and stored are viable

for a given constraint i. If all solutions (candidate, local minimum, and best) are viable, It sets the penalty to its minimum value. In case the local minimum does not find a feasible solution, and the best solution is feasible, the penalty is reduced under the coefficient δ_1 as long as the value of the penalty is above the minimum. Suppose the best solution and the candidate solution are not feasible solutions. In that case, the penalty increases under δ_2 as long as it does not exceed its maximum value p_i^{max}. Finally, we update the costs of the stored solutions with the new values of p_i.

Granularity: Neighborhoods are limited to moves involving pairs of geographically close nodes (i, j) such that j belongs to the closest Γ clients of i (VIDAL, 2022). We will use another criterion for our problem because our goal is to minimize travel times and not the total distance traveled. To do this, since time is not a constant value, we will use the minimum travel time as a criterion. Each endpoint of two edges $(i, j) \wedge (k, l) \in E$ is evaluated to obtain the minimum travel time as shown in Expression 4-13. We finally reserve the Γ clients (required edges) based on the minimum travel time from (i, j) to (k, l) for any combination of *modes*. Now the closest neighbors of (i, j) are $(k, l) \in \Gamma$. Therefore, the granularity parameter Γ limits the size of the neighborhoods. The exploration of the movements is organized in random order of the indexes i and j, and any improvement movement is applied immediately (VIDAL, 2022).

$$\min\Big\{\min_{t\in[0,H]}\{\Psi_{xy}(t) - t\}\ \forall x \in [i, j],\ y \in [k, l]\Big\} \tag{4-13}$$

Local Search: The local search used in our computational experiments is summarized in Algorithm 8 using preprocessing, concatenations, and lower bounds. In each improvement within the Local Search, it will be necessary to update the time structures of the lower bounds T^{LB} between all service pairs (sub-sequences) of each route.

In addition, It is made up of 7 moves that are executed randomly.

1. *Swap-inter k*: Exchange two sub-strings of k consecutive edges from different routes without altering the order of both sub-strings.

2. *Swap-intra k*: Exchange two sub-strings non-overlapping of k consecutive edges from a route without altering the order of both sub-strings.

3. *Swap21-inter*: Exchange 2 consecutive edges of one route with 1 edge of another.

4. *Crossover*: Exchange two sub-strings of any size from different routes that are connected to the depot.

5. *Relocate-inter k*: Remove a sub-string of k consecutive edges from one route and insert them into another without altering the order of the sub-string.

6. *Relocate-intra k*: Remove a sub-string of k consecutive edges from a route and insert them at another position within the same route without altering the order of the sub-string.

7. *2-Opt*: Remove 2 connections within a route and then reconnect the resulting segments by other connections.

Algorithm 8: Local Search

```
1  while improve = true do
2      improve = false
3      Update the lower bounds T^LB structures of the solution S'.
4      for Each move φ in random order do
5          The move φ can modify two routes of S'. Let z_CURRENT be the
            sum of the time of these two routes in S' and and let
            (σ_1 ⊕ ... ⊕ σ_K) and (σ'_1 ⊕ ... ⊕ σ'_L) be the sequences of
            services which form the two new routes.
6          if φ does not involves neighbors (Granularity Γ) then
7              continue Next move evaluation
8          Evaluate a lower bound on the time of the new routes:
9          z_LB = T^LB+(σ_1 ⊕ ... ⊕ σ_K) + T^LB+(σ'_1 ⊕ ... ⊕ σ'_L)
10         if z_LB ≥ z_CURRENT then
11             continue Next move evaluation
12         Let C_CURRENT be the cost of these two routes in S'
13         Evaluate the cost of the new routes with optimal mode
            choices, using dynamic programming (Bellman's algorithm)
            and the known auxiliary data structures T^EXACT and E^EXACT:
14         C_AFTER = C(σ_1 ⊕ ... ⊕ σ_K) + C(σ'_1 ⊕ ... ⊕ σ'_L)
15         if C_AFTER ≥ C_CURRENT then
16             continue Next move evaluation
17         UpdateRoutes(New routes)
18         improve = true
19         break
20 return A local minimum S*'
```

Acceptance Criteria: The *better* acceptance criterion is a classic strategy for metaheuristic descent where only solutions that are better than the

current one are accepted. However, these criteria tend to get stuck in a local minimum, so it seems sensible sometimes to accept solutions that are worse than the current (PISINGER; ROPKE, 2007) solution. Therefore, the simulated annealing proposed by Pisinger and Ropke (2007) will be used in our metaheuristic, where a solution $S^{*'}$ is accepted given the current solution S^{*} with probability equal to Equation 4-14.

$$P_{acceptance} = e^{-\frac{f(S^{*'}) - f(S^{*})}{T}} \tag{4-14}$$

$T > 0$ is the temperature and f represent the objective function. The method consists of setting the initial and final temperature based on the initial and final probabilities, the only parameters controlling the simulated annealing. To define the initial and final temperature, we use Equations 4-15 and 4-16 respectively.

$$T_i = \frac{P_i \ f(S_0)}{-log(P_i)} \tag{4-15}$$

$$T_f = \frac{P_f \ f(S_0)}{-log(P_f)} \tag{4-16}$$

S_0 represents the initial solution, P_i and P_f the initial and final acceptance probability. Finally, the cooling rate (c) is calculated in Equation 4-17, considering the number of iterations n.

$$c = \left(\frac{T_i}{T_f}\right)^{1/n} \tag{4-17}$$

The temperature starts at T_i and decreases each iteration using the expression $T = T \ c$, where $0 < c < 1$ until reaching the temperature T_f at iteration n.

Perturbation: The Perturbation used consists of the execution of p movements at random. The neighborhoods or movements used are the same as for the local search, and we also include *OpenRoute* function, which opens a route if a solution is infeasible. In Perturbation, there is no evaluation of the cost of the movement before execution.

5
Computational Experiments

We code this work's preprocessing and metaheuristic methods in *Julia* 1.7. The tests were carried out in an Intel Core i7-8700K CPU @ 3.7GHz (12 cores), with 64GB of RAM. All the tests carried out were run ten times with different seeds.

5.1
ILS and Energy Parameters

We compute the energy consumption rate given a speed v as in Fernandez et al. (2020) using Equation 2-8. We obtain Table 5.1 of the energy parameters associated with the consumption rate based on the specifications of an Eforce EF18 electric truck and an IVECO Stralis conventional truck.

Table 5.1: Energy parameters

Parameter	Unit	Symbol	Value
Auxiliary power	kW	P_0	3
Weight	kg	m	15000
Gravity	m/s^2	g	9.81
Air density (200m AMSL, 20°C)	kg/m^3	ρ	1.165
Frontal Area	m^2	A_F	9.69
Engine efficiency		η	0.97
Rolling resistance		C_r	0.008
Drag coefficient		C_w	0.8

On the local search side, we work under the parameters of Table 5.2. These parameters were adjusted through various experiments in type C instances. They obtained the shortest average travel time got in the experiments. We will use 100 iterations with 20 restarts (i.e., 2000 total iterations). In the swap and relocate movements, we will use sizes of up to five clients ($k \leq 5$). We will apply three local search movements in the perturbation without time, battery, or load restrictions.

Also, we only need the initial and final probability parameters for the acceptance criteria with Annealing Simulating. Finally, we will use a granularity of forty clients or required arcs. Our neighborhood will restrict the search for solutions when the movement does not involve the closest neighbors concerning the first Γ clients with minimum travel time.

Table 5.2: Local Search parameters

Parameter	Symbol	Value
Restart	r	20
Iteration	n	100
Perturbations	p	3
Initial probability	P_i	0.215
Final probability	P_f	0.005
Length Swap moves	k_S	5
Length Relocate moves	k_R	5
Granularity	Γ	40
Buckets	B	200

To further reduce the complexity of querying arrival times and energy consumption, we will divide the planning horizon into B *buckets* as in Vidal et al. (2021) and creates an auxiliary list whose i^{th} element points to the part of the function that contains the time $(i-1)H/B$. Any travel time query from time t is resolved by comparing the buckets at indices $\lfloor t/B \rfloor$ and $\lceil t/B \rceil$: if both buckets point to the same piece, then this piece is returned at time $O(1)$ otherwise, a binary search between the pieces is started and completed in $O(log\ h_{ij})$ time (VIDAL et al., 2021).

For the penalties, the parameters used were obtained empirically. We established a base surcharge of 100 proportional to the excess violation for the load, battery, and time. This cost overrun decreases to a minimum value equal to 10 through the reduction coefficient δ_1 as long as a better solution is found and this minimum value is not exceeded. Otherwise, the value increases by the increment coefficient δ_2 as long as the value does not exceed the maximum value.

Table 5.3: Penalty parameters

Parameter	Symbol	Value
Penalty time base	P_T^{base}	10^2
Penalty battery base	P_B^{base}	10^2
Penalty load base	P_L^{base}	10^2
Penalty time min	P_T^{min}	10
Penalty battery min	P_B^{min}	10
Penalty load min	P_L^{min}	10
Penalty time max	P_T^{max}	10^4
Penalty battery max	P_B^{max}	10^4
Penalty load max	P_L^{max}	10^4
Reduction coefficient	δ_1	0.99
Increment coefficient	δ_2	1.5

5.2 Benchmark Instances

We will use the TD-CARP benchmark instances generated from Vidal et al. (2021) for our analysis. As in Vidal et al. (2021), we will assume that the travel speed during service represents 70% of the deadhead travel speed, i.e., $\hat{v}_{ij}(t) = 0.7 \times v_{ij}(t)\ \forall (i,j) \in E_R$. In addition, we consider six different battery sizes, i.e. $C \in [100, 200, 300, 400, 500, 600]$ in kWh. According to Fernandez et al. (2020), the Eforce EF18 can use batteries up to 630 kWh.

5.2.1 Performance of the Time-dependent Quickest Path and Energy Consumption Algorithm

We evaluate the performance of the quickest path algorithm together with its energy consumption function. Since this algorithm is only executed during preprocessing, its results can be preserved for successive routing solutions (for example, for different client sets) (VIDAL et al., 2021) as long as the speed estimates and assumptions are unchanged. Table 5.4 reports the total CPU time spent by the algorithm for each of the three instance categories where "$Eglese - s$" instances are the largest tested. These have between 140 vertices that our algorithm took on average 960 seconds to preprocess. On the C instances side, our algorithm was able to preprocess the quickest paths and their energy consumption in less than 180 seconds.

Table 5.4: Performance of the quickest path and energy consumption preprocessing Algorithm.

Instance	|V|	|ER|	CPU Time (s)		
			Low	Medium	High
C01	69	79	101.34	111.60	125.89
C02	48	53	39.69	40.72	42.48
C03	46	51	36.94	39.50	43.04
C04	60	72	76.50	81.44	85.82
C05	56	65	57.95	59.24	63.83
C06	38	51	29.24	31.80	32.63
C07	54	52	47.16	47.36	51.60
C08	66	63	81.61	87.03	91.45
C09	76	97	141.03	151.23	165.25
C10	60	55	65.06	67.12	71.34
egl-e1-B	77	51	122.78	135.18	156.89
egl-e2-B	77	72	131.46	145.82	156.07
egl-e3-B	77	87	129.67	138.37	156.41
egl-e4-B	77	98	129.07	144.01	162.78
egl-s1-B	140	75	1041.69	1033.06	823.26
egl-s2-B	140	147	981.75	983.50	826.31
egl-s3-B	140	159	997.55	995.85	865.70
egl-s4-B	140	190	1100.64	976.98	888.11

5.3 E-TDCARP results

We run our ILS metaheuristic ten times with different seeds on all TDCARP instances with a battery capacity level equal to 300 kWh. The tables 5.5 to 5.7 report the results of this experiment. Since these are the first results obtained in E-TDCARP, it is impossible to compare them with some results from the existing literature.

Table 5.5: Performance of the ILS on low E-TDCARP instances

Instance	$\|E_R\|$	LB	UB	Avg	T(s)
C01	79	2390.99	2401.36	2397.02	1403.44
C02	53	1874.24	1898.94	1884.66	371.49
C03	51	1602.26	1604.52	1602.72	541.26
C04	72	1959.58	1965.99	1960.30	974.20
C05	65	2516.68	2530.86	2526.13	426.90
C06	51	1572.59	1572.77	1572.65	518.00
C07	52	2070.49	2088.60	2080.41	358.46
C08	63	1902.67	1910.42	1907.44	523.36
C09	97	3387.17	3407.01	3397.57	1737.27
C10	55	2163.26	2188.58	2176.71	385.17
egl-e1-B	51	1627.97	1641.80	1633.24	500.98
egl-e2-B	72	2344.83	2363.72	2351.82	1062.02
egl-e3-B	87	2907.85	2952.60	2934.28	1389.50
egl-e4-B	98	3367.07	3404.58	3374.14	1655.92
egl-s1-B	75	2312.66	2332.97	2321.51	2201.81
egl-s2-B	147	4454.35	4538.83	4510.87	4318.81
egl-s3-B	159	4727.76	4771.01	4750.40	4969.70
egl-s4-B	190	5679.66	5749.17	5702.42	5519.88

Table 5.6: Performance of the ILS on medium E-TDCARP instances

Instance	$\|E_R\|$	LB	UB	Avg	T(s)
C01	79	2431.20	2451.58	2443.84	1610.44
C02	53	1896.32	1923.70	1904.36	431.25
C03	51	1637.80	1638.96	1638.26	616.80
C04	72	1996.80	2004.88	1998.03	1124.06
C05	65	2518.55	2561.83	2545.82	450.19
C06	51	1582.08	1584.60	1582.35	602.44
C07	52	2118.61	2133.92	2124.79	389.90
C08	63	1945.69	1988.28	1959.61	589.86
C09	97	3424.18	3469.75	3453.86	2020.80
C10	55	2207.63	2226.57	2220.91	430.77
egl-e1-B	51	1653.19	1662.47	1658.95	520.95
egl-e2-B	72	2398.10	2409.53	2401.76	1136.10
egl-e3-B	87	2958.46	3008.54	2994.59	1412.85
egl-e4-B	98	3481.56	3547.15	3494.35	1679.47
egl-s1-B	75	2392.14	2417.69	2404.83	2199.99
egl-s2-B	147	4649.29	4703.96	4675.77	4136.65
egl-s3-B	159	4869.67	4990.51	4923.94	5199.01
egl-s4-B	190	5833.11	5953.13	5870.52	5605.23

Table 5.7: Performance of the ILS on high E-TDCARP instances

Instance	$\|E_R\|$	LB	UB	Avg	T(s)
C01	79	2450.71	2478.51	2465.64	1871.43
C02	53	1889.78	1915.99	1902.34	515.35
C03	51	1647.38	1647.38	1647.38	733.25
C04	72	2013.53	2024.61	2015.17	1358.18
C05	65	2514.72	2547.38	2526.75	505.67
C06	51	1566.24	1567.27	1566.48	695.69
C07	52	2155.70	2168.58	2162.53	454.47
C08	63	1959.71	2005.76	1974.76	687.45
C09	97	3435.39	3465.98	3451.06	2387.67
C10	55	2228.00	2228.00	2228.00	486.83
egl-e1-B	51	1672.81	1682.74	1679.28	580.46
egl-e2-B	72	2444.33	2466.26	2457.89	1265.79
egl-e3-B	87	3010.81	3039.14	3022.92	1458.76
egl-e4-B	98	3584.15	3626.20	3603.65	1784.13
egl-s1-B	75	2436.65	2465.53	2449.76	2108.10
egl-s2-B	147	4687.45	4833.62	4780.20	4170.21
egl-s3-B	159	4971.78	5040.76	5009.41	5330.08
egl-s4-B	190	5953.03	6049.46	5997.43	5333.49

5.4 TDCARP solutions without considering Battery Capacity

We extend our ILS metaheuristic to TDCARP to compare our solutions with those obtained in Vidal et al. (2021) since E-TDCARP is a new extension of TDCARP, and there are no solutions in the literature. The results are found in Tables 5.8 to 5.10, where we obtained an average Gap of 0.50% in an average time of 643 seconds in the group of instances C. For instances *Eglese*, we obtained an average Gap of 1.73% in an average time of 2088 seconds. With these results, we can establish that our metaheuristic approach produces quality solutions very close to the existing literature. In addition, we perform a reassessment of the solutions based on energy consumption to determine if the TDCARP solutions continue to be feasible when considering this characteristic.

This analysis allows us to visualize that when introducing the EVs in the TDCARP of Vidal et al. (2021) without considering the battery capacity as a constraint can generate unfeasible solutions when considering working with low battery levels ($< 300\ kWh$). Although considering working with larger capacity batteries ($> 400\ kWh$) can eliminate the dependency on energy consumption in the tested instances, this does not mean taking full advantage of the vehicle's resources and, therefore, generating cost overruns for the EVs implementation.

Table 5.8: Performance of the ILS on low TDCARP instances.

Instance		Infeasible battery level (kWh)						HGS	ILS			
Name	$\|E_R\|$	100	200	300	400	500	600	Best	Best	Avg	Gap	T(s)
C01	79	**x**	-	-	-	-	-	2381.01	2394.34	2398.24	0.724%	1266.20
C02	53	**x**	-	-	-	-	-	1874.24	1877.44	1888.60	0.766%	318.97
C03	51	**x**	-	-	-	-	-	1602.26	**1602.26**	1602.49	0.014%	480.72
C04	72	**x**	-	-	-	-	-	1959.58	**1959.58**	1960.12	0.028%	860.02
C05	65	**x**	**x**	-	-	-	-	2509.43	2527.52	2530.49	0.839%	464.70
C06	51	**x**	-	-	-	-	-	1572.59	**1572.59**	1572.65	0.004%	470.26
C07	52	**x**	**x**	-	-	-	-	2070.49	**2070.49**	2080.66	0.491%	301.35
C08	63	**x**	-	-	-	-	-	1899.67	**1899.67**	1906.94	0.383%	426.17
C09	97	**x**	**x**	-	-	-	-	3350.24	3367.64	3393.62	1.295%	1528.71
C10	55	**x**	-	-	-	-	-	2163.26	**2163.26**	2181.83	0.858%	316.56
Avg C											0.540%	643.37
egl-e1-B	51	**x**	**x**	-	-	-	-	1627.97	**1627.97**	1630.69	0.167%	361.39
egl-e2-B	72	**x**	**x**	-	-	-	-	2342.83	2344.22	2349.31	0.277%	899.51
egl-e3-B	87	**x**	**x**	-	-	-	-	2904.36	2915.19	2940.28	1.237%	1195.75
egl-e4-B	98	**x**	**x**	-	-	-	-	3341.41	3365.86	3371.62	0.904%	1437.13
egl-s1-B	75	**x**	**x**	-	-	-	-	2294.85	2308.50	2322.09	1.187%	1079.35
egl-s2-B	147	**x**	**x**	-	-	-	-	4382.64	4504.52	4524.50	3.237%	3127.91
egl-s3-B	159	**x**	**x**	-	-	-	-	4633.7	4720.43	4752.29	2.559%	3904.06
egl-s4-B	190	**x**	**x**	-	-	-	-	5524.07	5653.59	5707.07	3.313%	4428.81
Avg egl											1.610%	2054.24

Table 5.9: Performance of the ILS on medium TDCARP instances.

Instance		Infeasible battery level (kWh)						HGS	ILS			
Name	$\|E_R\|$	100	200	300	400	500	600	Best	Best	Avg	Gap	T(s)
C01	79	**x**	-	-	-	-	-	2423.80	2431.03	2444.10	0.838%	1426.24
C02	53	**x**	-	-	-	-	-	1896.14	**1896.14**	1905.03	0.469%	369.67
C03	51	**x**	-	-	-	-	-	1637.80	**1637.80**	1638.03	0.014%	550.79
C04	72	**x**	**x**	-	-	-	-	1996.75	1996.80	1999.21	0.123%	1003.57
C05	65	**x**	**x**	-	-	-	-	2518.55	2528.29	2545.02	1.051%	336.14
C06	51	**x**	-	-	-	-	-	1582.08	**1582.08**	1582.08	0.000%	541.24
C07	52	**x**	-	-	-	-	-	2118.61	**2118.61**	2124.28	0.268%	329.08
C08	63	**x**	-	-	-	-	-	1945.23	1949.43	1958.23	0.668%	481.20
C09	97	**x**	**x**	-	-	-	-	3407.54	3420.30	3447.43	1.171%	1783.91
C10	55	**x**	-	-	-	-	-	2207.63	**2207.63**	2214.46	0.309%	350.69
Avg C											0.491%	717.25
egl-e1-B	51	**x**	**x**	-	-	-	-	1653.19	**1653.19**	1658.81	0.340%	369.38
egl-e2-B	72	**x**	**x**	-	-	-	-	2392.91	2395.44	2401.14	0.344%	949.54
egl-e3-B	87	**x**	**x**	-	-	-	-	2956.03	2986.64	2998.77	1.446%	1215.77
egl-e4-B	98	**x**	**x**	-	-	-	-	3451.16	3479.76	3486.20	1.015%	1443.39
egl-s1-B	75	**x**	**x**	-	-	-	-	2381.80	2386.77	2399.60	0.747%	1107.68
egl-s2-B	147	**x**	**x**	-	-	-	-	4514.46	4640.34	4665.93	3.355%	3132.46
egl-s3-B	159	**x**	**x**	-	-	-	-	4774.16	4862.28	4903.47	2.709%	3923.42
egl-s4-B	190	**x**	**x**	-	-	-	-	5663.44	5799.40	5862.04	3.507%	4230.39
Avg egl											1.683%	2046.50

Table 5.10: Performance of the ILS on high TDCARP instances.

Instance		Infeasible battery level (kWh)						HGS	ILS			
Name	$\|E_R\|$	100	200	300	400	500	600	Best	Best	Avg	Gap	T(s)
C01	79	**x**	**x**	-	-	-	-	2444.24	2450.51	2463.25	0.778%	1647.81
C02	53	**x**	-	-	-	-	-	1889.78	1891.04	1901.52	0.621%	443.90
C03	51	**x**	-	-	-	-	-	1647.38	**1647.38**	1647.38	0.000%	650.90
C04	72	**x**	**x**	-	-	-	-	2012.42	**2012.42**	2014.45	0.101%	1222.50
C05	65	**x**	**x**	-	-	-	-	2505.37	2511.86	2522.70	0.692%	423.77
C06	51	**x**	-	-	-	-	-	1566.24	**1566.24**	1566.54	0.019%	631.54
C07	52	**x**	-	-	-	-	-	2153.38	2159.50	2165.21	0.549%	388.18
C08	63	**x**	-	-	-	-	-	1959.71	**1959.71**	1974.10	0.734%	562.77
C09	97	**x**	**x**	-	-	-	-	3420.94	3435.39	3457.48	1.068%	2115.53
C10	55	**x**	-	-	-	-	-	2228.00	**2228.00**	2228.00	0.000%	399.38
Avg C											0.456%	848.63
egl-e1-B	51	**x**	**x**	-	-	-	-	1671.31	1672.81	1678.50	0.430%	403.79
egl-e2-B	72	**x**	**x**	-	-	-	-	2430.55	**2430.55**	2446.56	0.659%	1050.61
egl-e3-B	87	**x**	**x**	-	-	-	-	2978.79	2992.96	3027.46	1.634%	1250.69
egl-e4-B	98	**x**	**x**	-	-	-	-	3555.56	3577.59	3600.32	1.259%	1529.86
egl-s1-B	75	**x**	**x**	-	-	-	-	2423.32	2432.35	2445.61	0.920%	1187.08
egl-s2-B	147	**x**	**x**	-	-	-	-	4640.3	4746.61	4781.41	3.041%	3198.68
egl-s3-B	159	**x**	**x**	-	-	-	-	4840.15	4958.80	5004.68	3.399%	4385.41
egl-s4-B	190	**x**	**x**	-	-	-	-	5766.76	5934.82	5982.04	3.733%	4314.36
Avg egl											1.884%	2165.06

Although analyzing the results of TDCARP under battery levels is a viable option in route design, this does not mean making a decision based solely on these values. New results may exist when this constraint is introduced directly to the problem to search for feasible and optimal solutions under this condition (which the original TDCARP does not show), as we will see in the next section.

5.5 Best E-TDCARP Results under different levels of Battery Capacity

We ran our metaheuristic ILS ten times with different seeds in all instances TDCARP for each battery level described in the previous section. Tables 5.11 to 5.13 report the best results of this experiment. Since they are the first results obtained in E-TDCARP, it is impossible to compare them with some existing literature results. The development of this problem would allow not only to evaluate the route design but also to introduce a pre-existing fleet of EVs with a certain battery level where the original TDCARP results are not feasible.

Table 5.11: Best Results of the ILS on low time-dependency E-TDCARP.

Instance	Battery level (kWh)					
	100	200	300	400	500	600
C01	-	2393.71	2390.99	2393.08	2388.60	2393.10
C02	-	1874.24	1874.24	1874.24	1877.44	1877.61
C03	-	1602.26	1602.26	1602.26	1602.26	1602.26
C04	-	1959.58	1959.58	1959.58	1959.58	1959.58
C05	-	**2509.98**	2516.68	2519.40	2509.43	2509.43
C06	-	1572.59	1572.59	1572.59	1572.59	1572.59
C07	-	**2070.49**	2070.49	2070.49	2070.49	2070.49
C08	-	1899.67	1902.67	1903.87	1903.87	1903.87
C09	-	**3367.56**	3387.17	3382.13	3378.41	3383.68
C10	-	2163.26	2163.26	2163.26	2163.26	2163.26
egl-e1-B	-	**1636.60**	1627.97	1627.97	1627.97	1627.97
egl-e2-B	-	**2346.23**	2344.83	2343.60	2343.11	2343.11
egl-e3-B	-	**2920.09**	2907.85	2934.64	2927.80	2927.80
egl-e4-B	-	**3368.32**	3367.07	3352.63	3364.92	3364.92
egl-s1-B	-	-	2312.66	2308.53	2313.40	2313.40
egl-s2-B	-	**4756.17**	4454.35	4489.85	4494.40	4486.50
egl-s3-B	-	-	4727.76	4689.64	4715.73	4721.27
egl-s4-B	-	-	5679.66	5660.12	5658.58	5675.00

Table 5.12: Best Results of the ILS on medium time-dependency E-TDCARP.

Instance	Battery level (kWh)					
	100	200	300	400	500	600
C01	-	2434.42	2431.20	2434.66	2432.22	2424.44
C02	-	1896.32	1896.32	1896.14	1896.14	1896.14
C03	-	1637.80	1637.80	1637.80	1637.80	1637.80
C04	-	**1999.06**	1996.80	1996.80	1996.80	1996.80
C05	-	**2528.29**	2518.55	2531.73	2528.29	2522.74
C06	-	1582.08	1582.08	1582.08	1582.08	1582.08
C07	-	2118.61	2118.61	2118.61	2118.61	2118.61
C08	-	1947.79	1945.69	1948.25	1945.69	1947.79
C09	-	**3443.05**	3424.18	3424.71	3419.07	3419.92
C10	-	2207.63	2207.63	2207.63	2207.63	2207.63
egl-e1-B	-	**1669.47**	1653.19	1653.19	1653.19	1653.19
egl-e2-B	-	**2406.15**	2398.10	2393.60	2395.44	2392.91
egl-e3-B	-	**2978.40**	2958.46	2987.68	2989.39	2979.30
egl-e4-B	-	**3479.54**	3481.56	3480.84	3477.92	3484.25
egl-s1-B	-	**2722.30**	2392.14	2391.60	2386.96	2395.57
egl-s2-B	-	**4890.05**	4649.29	4643.53	4607.38	4650.25
egl-s3-B	-	-	4869.67	4864.98	4875.82	4895.10
egl-s4-B	-	-	5833.11	5789.81	5821.85	5836.41

Table 5.13: Best Results of the ILS on high time-dependency E-TDCARP.

Instance	Battery level (kWh)					
	100	200	300	400	500	600
C01	-	**2455.18**	2450.71	2450.54	2448.18	2451.55
C02	-	1889.78	1889.78	1889.78	1889.78	1889.78
C03	-	1647.38	1647.38	1647.38	1647.38	1647.38
C04	-	**2014.96**	2013.53	2012.42	2012.42	2013.53
C05	-	**2520.93**	2514.72	2517.65	2521.46	2517.81
C06	-	1566.24	1566.24	1566.24	1566.24	1566.24
C07	-	2155.70	2155.70	2155.70	2155.70	2153.38
C08	-	1959.71	1959.71	1959.71	1962.25	1962.40
C09	-	**3437.98**	3435.39	3445.61	3442.56	3440.25
C10	-	2228.00	2228.00	2228.00	2228.00	2228.00
egl-e1-B	-	**1719.53**	1672.81	1671.31	1679.68	1679.68
egl-e2-B	-	**2508.39**	2444.33	2433.45	2439.27	2439.27
egl-e3-B	-	**3053.47**	3010.81	3006.85	3013.53	3012.99
egl-e4-B	-	**3598.27**	3584.15	3589.54	3590.81	3579.09
egl-s1-B	-	-	2436.65	2423.32	2430.16	2433.41
egl-s2-B	-	-	4687.45	4747.31	4743.05	4750.80
egl-s3-B	-	-	4971.78	4948.28	4961.07	4948.89
egl-s4-B	-	-	5953.03	5896.02	5952.61	5920.69

However, we can highlight the finding of feasible solutions obtained with batteries with a capacity of 200 kWh since some TDCARP solutions are unfeasible under this battery capacity (results in bold). In the case of 100 kWh capacity, the metaheuristic failed to obtain feasible solutions (even expanding the number of routes).

In summary, we present the graph 5.1 showing the average times of the groups of instances type C and $Eglese$ for each battery capacity. This graph shows us that when the battery capacity is lower or more restrictive, the computation times tend to increase because it is more difficult to find a viable solution for low battery levels. On the other hand, when the battery capacity is greater, the times tend to stabilize since the battery restriction begins to be insignificant (i.e., other restrictions such as load or time predominate).

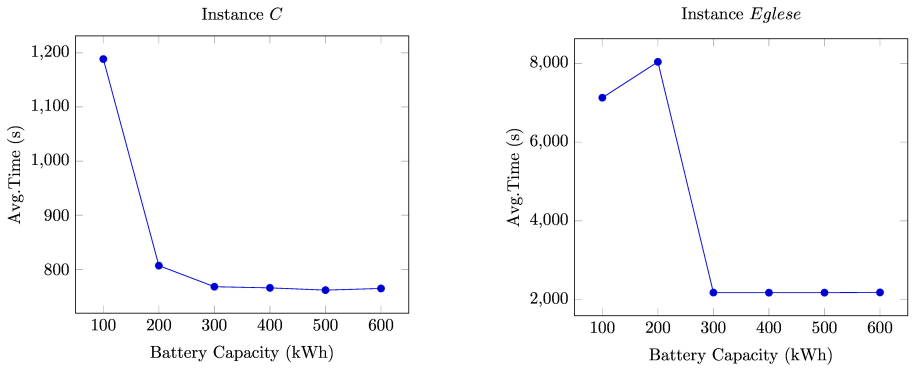


Figure 5.1: Representation of average computational times of E-TDCARP results.

6
Conclusion and Future works

This paper introduces a new extension of TDCARP that captures the non-linear energy consumption process based on speed and travel time without approximations: The Electric Capacitated Arc Routing Problem with Time-dependent travel times and Speed-dependent Energy Consumption Rate (E-TDCARP). We proposed a closed-form energy consumption and arrival time preprocessing method based on departure time and an ILS + SA metaheuristic with restarts and adaptive penalty to solve the problem. This problem consists of designing routes to serve a finite number of arcs and edges while minimizing the total duration of each route.

We perform an experiment comparing the results of our metaheuristic under different levels of battery capacity in kWh. Additionally, we adapt our ILS metaheuristic to solve the TDCARP proposed in Vidal et al. (2021) and its feasibility when EVs are used. The results suggest that our ILS can offer reasonable quality solutions for the TDCARP compared to the results obtained in Vidal et al.(2021). Furthermore, our results on introducing electric vehicles show that ignoring energy consumption can lead to infeasible solutions for the TDCARP. Also, our results in implementing the energy consumption constraint (E-TDCARP) show that it is possible to design feasible routes for those TDCARP instances where the energy consumption is violated when the vehicles' battery capacity is already defined.

6.1
Future works

Interesting research directions include the design of exact algorithms such as *branch-and-cut-and-price* or implementation within solvers such as the *VRP Solver* (PESSOA et al., 2020) since, thanks to our preprocessing method, it is possible to work with the non-linear energy consumption rate. In addition, more efficient genetic and hybrid metaheuristic methods for E-TDCARP and EV-TDARP can be developed to provide better or equal solutions than those presented in this work in less computational time.

Another exciting possibility is approaching the E-TDCARP with an energy consumption rate based on speed and mass. If the additional mass is significant, the energy consumption is affected. In this case, it is possible to work with the model considering non-fixed mass, where the value of the ECR is given by the mass and the speed (that is, $ECR(v) \rightarrow ECR(m, v)$).

Furthermore, it is essential to establish that the mass only increases when a service is executed, so the energy consumption function maintains the mass in deadhead paths. In the case of services, we can establish different possibilities, such as working with a predictive model of mass growth based on real-data edge information or working under overestimates where the mass increase is applied completely and instantly at the beginning of the service.

Lastly, it would be interesting to incorporate the price of electricity into the objective function. It will influence the problem's analysis and design because not only the quickest paths must be taken into consideration, but also paths with more energy savings and intermediate value paths (balance between time and consumption based on an objective function). These values can be obtained by adapting the Quickest Path Preprocessing Algorithm (i.e., changing the selection criterion in Lower Envelope to minimize energy consumption). In addition, we should consider whether waiting between services is profitable. Although the FIFO property defines that waiting is not profitable, this does not mean that starting later can reduce energy consumption and cost.

.

7
Bibliography

ABDALLAH, K. S.; ADEL, Y. Electric vehicles routing problem with variable speed and time windows. In: **International Conference on Industry, Engineering & Management Systems**. [S.l.: s.n.], 2020. p. 55–65.

ASAMER, J. et al. Sensitivity analysis for energy demand estimation of electric vehicles. **Transportation Research Part D: Transport and Environment**, Elsevier, v. 46, p. 182–199, 2016.

BASSO, R. et al. Energy consumption estimation integrated into the electric vehicle routing problem. **Transportation Research Part D: Transport and Environment**, Elsevier, v. 69, p. 141–167, 2019.

BLACK, D.; EGLESE, R.; WØHLK, S. The time-dependent prize-collecting arc routing problem. **Computers & Operations Research**, Elsevier, v. 40, n. 2, p. 526–535, 2013.

BLACK, D.; EGLESE, R.; WØHLK, S. The time-dependent multiple-vehicle prize-collecting arc routing problem. 2015.

BRANDÃO, J.; EGLESE, R. A deterministic tabu search algorithm for the capacitated arc routing problem. **Computers & Operations Research**, Elsevier, v. 35, n. 4, p. 1112–1126, 2008.

CESELLI, A. et al. A branch-and-cut-and-price algorithm for the electric vehicle routing problem with multiple technologies. In: SPRINGER. **Operations Research Forum**. [S.l.], 2021. v. 2, n. 1, p. 1–33.

CORBERÁN, A.; PRINS, C. Recent results on arc routing problems: An annotated bibliography. **Networks**, Wiley Online Library, v. 56, n. 1, p. 50–69, 2010.

EHSANI, M. et al. **Modern electric, hybrid electric, and fuel cell vehicles**. [S.l.]: CRC press, 2018.

FERNÁNDEZ, E. et al. Arc routing with electric vehicles: dynamic charging and speed-dependent energy consumption. 2020.

FLORIO, A. M.; ABSI, N.; FEILLET, D. Routing electric vehicles on congested street networks. **Transportation Science**, INFORMS, v. 55, n. 1, p. 238–256, 2021.

FROGER, A. et al. The electric vehicle routing problem with capacitated charging stations. **Transportation Science**, INFORMS, 2021.

FROGER, A. et al. Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. **Computers & Operations Research**, Elsevier, v. 104, p. 256–294, 2019.

GOEKE, D.; SCHNEIDER, M. Routing a mixed fleet of electric and conventional vehicles. **European Journal of Operational Research**, v. 245, n. 1, p. 81–99, 2015. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221715000697>.

GOLDEN, B. L.; WONG, R. T. Capacitated arc routing problems. **Networks**, Wiley Online Library, v. 11, n. 3, p. 305–315, 1981.

HAGHANI, A.; JUNG, S. A dynamic vehicle routing problem with time-dependent travel times. **Computers & operations research**, Elsevier, v. 32, n. 11, p. 2959–2986, 2005.

HERSHBERGER, J. Finding the upper envelope of n line segments in o (n log n) time. **Information Processing Letters**, Elsevier, v. 33, n. 4, p. 169–174, 1989.

HERTZ, A.; LAPORTE, G.; MITTAZ, M. A tabu search heuristic for the capacitated arc routing problem. **Operations research**, INFORMS, v. 48, n. 1, p. 129–135, 2000.

HERTZ, A.; MITTAZ, M. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. **Transportation science**, INFORMS, v. 35, n. 4, p. 425–434, 2001.

ICHOUA, S.; GENDREAU, M.; POTVIN, J.-Y. Vehicle dispatching with time-dependent travel times. **European journal of operational research**, Elsevier, v. 144, n. 2, p. 379–396, 2003.

IEA, I. e. a. Co2 emissions from fuel combustion 2019—highlights. In: **CO2 emissions from fuel combustion 2019—highlights**. [S.l.]: IEA, 2019.

JIN, X. et al. Planning of garbage collection service: an arc-routing problem with time-dependent penalty cost. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 22, n. 5, p. 2692–2705, 2020.

JING, W. et al. Electric vehicles: A review of network modelling and future research needs. **Advances in Mechanical Engineering**, SAGE Publications Sage UK: London, England, v. 8, n. 1, p. 1687814015627981, 2016.

KUCUKOGLU, I.; DEWIL, R.; CATTRYSSE, D. The electric vehicle routing problem and its variations: A literature review. **Computers & Industrial Engineering**, Elsevier, v. 161, p. 107650, 2021.

LACOMME, P.; PRINS, C.; RAMDANE-CHERIF, W. Competitive memetic algorithms for arc routing problems. **Annals of Operations Research**, Springer, v. 131, n. 1, p. 159–185, 2004.

LOURENCO, H. R.; MARTIN, O.; STUTZLE, T. **Iterated local search. Handbook of Metaheuristics. F. Glover and G. Kochenberger**. [S.l.]: Springer-Verlag, 2003.

MARTINS, L. d. C. et al. Electric vehicle routing, arc routing, and team orienteering problems in sustainable transportation. **Energies**, MDPI, v. 14, n. 16, p. 5131, 2021.

MONTOYA, A. et al. The electric vehicle routing problem with nonlinear charging function. **Transportation Research Part B: Methodological**, Elsevier, v. 103, p. 87–110, 2017.

MORGANTI, E.; BROWNE, M. Technical and operational obstacles to the adoption of electric vans in france and the uk: An operator perspective. **Transport Policy**, Elsevier, v. 63, p. 90–97, 2018.

MURAKAMI, K. A new model and approach to electric and diesel-powered vehicle routing. **Transportation Research Part E: Logistics and Transportation Review**, Elsevier, v. 107, p. 23–37, 2017.

NABEREZHNYKH, D. et al. Electric vehicle charging points for freight vehicles in central london (version-draft 0.7). **Prepared for Central London FQP by Transport & Travel Research Ltd, in partnership with TRL and Zero Carbon Futures. URL http://www. centrallondonfqp. org/app/download/12240926/CLFQP_EVCP_strategy+ report_Final+ v1+ 0. pdf. Last accessed**, v. 9, n. 6, p. 2014, 2012.

NIEKERK, M. E. van K.; AKKER, J. Van den; HOOGEVEEN, J. Scheduling electric vehicles. **Public Transport**, Springer, v. 9, n. 1, p. 155–176, 2017.

PELLETIER, S.; JABALI, O.; LAPORTE, G. The electric vehicle routing problem with energy consumption uncertainty. **Transportation Research Part B: Methodological**, Elsevier, v. 126, p. 225–255, 2019.

PESSOA, A. et al. A generic exact solver for vehicle routing and related problems. **Mathematical Programming**, Springer, v. 183, n. 1, p. 483–523, 2020.

PISINGER, D.; ROPKE, S. A general heuristic for vehicle routing problems. **Computers & operations research**, Elsevier, v. 34, n. 8, p. 2403–2435, 2007.

SISSINE, F. Energy independence and security act of 2007: a summary of major provisions. In: LIBRARY OF CONGRESS WASHINGTON DC CONGRESSIONAL RESEARCH SERVICE. [S.l.], 2007.

SPLIET, R.; DABIA, S.; WOENSEL, T. V. The time window assignment vehicle routing problem with time-dependent travel times. **Transportation Science**, INFORMS, v. 52, n. 2, p. 261–276, 2018.

TAGMOUTI, M.; GENDREAU, M.; POTVIN, J.-Y. Arc routing problems with time-dependent service costs. **European Journal of Operational Research**, Elsevier, v. 181, n. 1, p. 30–39, 2007.

TAGMOUTI, M.; GENDREAU, M.; POTVIN, J.-Y. A variable neighborhood descent heuristic for arc routing problems with time-dependent service costs. **Computers & Industrial Engineering**, Elsevier, v. 59, n. 4, p. 954–963, 2010.

USBERTI, F. L.; FRANÇA, P. M.; FRANÇA, A. L. M. Grasp with evolutionary path-relinking for the capacitated arc routing problem. **Computers & Operations Research**, Elsevier, v. 40, n. 12, p. 3206–3217, 2013.

VIDAL, T. Node, edge, arc routing and turn penalties: Multiple problems—one neighborhood extension. **Operations Research**, INFORMS, v. 65, n. 4, p. 992–1010, 2017.

VIDAL, T. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. **Computers & Operations Research**, Elsevier, v. 140, p. 105643, 2022.

VIDAL, T. et al. Arc routing with time-dependent travel times and paths. **Transportation Science**, INFORMS, v. 55, n. 3, p. 706–724, 2021.

WANG, L. et al. Time-dependent electric vehicle routing problem with time windows and path flexibility. **Journal of Advanced Transportation**, Hindawi, v. 2020, 2020.

WANG, Z.; JIN, H.; TIAN, M. Rank-based memetic algorithm for capacitated arc routing problems. **Applied soft computing**, Elsevier, v. 37, p. 572–584, 2015.

WATKISS, J. D.; SMITH, D. W. The energy policy act of 1992-a watershed for competition in the wholesale power market. **Yale J. on Reg.**, HeinOnline, v. 10, p. 447, 1993.

XIAO, Y. et al. Electric vehicle routing problem: A systematic review and a new comprehensive model with nonlinear energy recharging and consumption. **Renewable and Sustainable Energy Reviews**, Elsevier, v. 151, p. 111567, 2021.

YUAN, X. et al. Method for evaluating the real-world driving energy consumptions of electric vehicles. **Energy**, Elsevier, v. 141, p. 1955–1968, 2017.

ZHANG, S. et al. Fuzzy optimization model for electric vehicle routing problem with time windows and recharging stations. **Expert systems with applications**, Elsevier, v. 145, p. 113123, 2020.

ZHANG, S. et al. Electric vehicle routing problem with recharging stations for minimizing energy consumption. **International Journal of Production Economics**, Elsevier, v. 203, p. 404–413, 2018.

ZHAO, Z.; LI, X.; ZHOU, X. Distribution route optimization for electric vehicles in urban cold chain logistics for fresh products under time-varying traffic conditions. **Mathematical Problems in Engineering**, Hindawi, v. 2020, 2020.