

The Electric Time-Dependent Capacitated Arc Routing Problem

Jahir Llagas

Department of Economics and Business Economics - Aarhus University
Fuglesangs Allé 4, 8210 Aarhus, Denmark
jahir.llagas@econ.au.dk

Rafael Martinelli

Departamento de Engenharia Industrial - PUC-Rio
Rua Marquês de São Vicente, 225 - Gávea, Rio de Janeiro / RJ - 22.451-900
martinelli@puc-rio.br

ABSTRACT

With rising energy and environmental concerns, electric vehicles (EVs) are becoming crucial for logistics. Traffic congestion significantly affects travel times, especially in urban areas, causing EV speeds to vary across routes. Due to their limited range, models account for energy consumption based on speed and aerodynamics but are often oversimplified. Traditional models use average speeds or linear approximations, which do not accurately reflect real conditions, especially in Arc Routing Problems. We address these issues by defining the Electric Capacitated Arc Routing Problem with Time-dependent Travel Times and Speed-dependent Energy Consumption Rate. Each arc has a stepwise speed function, allowing speed changes while traveling. The goal is to serve arcs using a fleet of EVs with limited load and battery capacity, minimizing total travel time. We propose an exact energy consumption preprocessing algorithm and integrate it into an Iterated Local Search metaheuristic, comparing routes for EVs and conventional vehicles.

KEYWORDS. Electric Vehicles, Time-Dependent Travel Times, Capacitated Arc Routing.

RESUMO

Com o aumento das preocupações energéticas e ambientais, os veículos elétricos (VEs) estão se tornando cruciais para a logística. O congestionamento do tráfego afeta significativamente os tempos de viagem, especialmente em áreas urbanas, fazendo com que as velocidades dos VEs variem nas rotas. Devido ao alcance limitado dos VEs, modelos consideram o consumo de energia com base na velocidade e aerodinâmica, mas frequentemente são simplificados em excesso. Modelos tradicionais utilizam velocidades médias ou aproximações lineares, que não refletem com precisão as condições reais, especialmente em Problemas de Roteamento em Arcos. Tais questões são abordadas definindo o Problema de Roteamento Capacitado em Arcos para Veículos Elétricos com Tempos de Viagem Dependentes do Tempo e Taxa de Consumo de Energia Dependente da Velocidade. Cada arco tem uma função degrau de velocidade, permitindo mudanças de velocidade durante a viagem. O objetivo é atender os arcos usando uma frota de VEs com capacidade limitada de carga e bateria, minimizando o tempo total de viagem. Um algoritmo exato de pré-processamento de consumo de energia é proposto e integrado em uma metaheurística de Busca Local Iterada, comparando rotas para VEs e veículos convencionais.

PALAVRAS CHAVE. Veículos Elétricos, Electric Vehicles, Tempos de Viagem Dependentes do Tempo, Roteamento Capacitado em Arcos.

1. Introduction

In recent years, many countries have adopted policies to reduce greenhouse gas emissions and promote alternative energy sources. U.S. policies aim to decrease fossil fuel use, cut CO₂ emissions, reduce dependence on foreign oil, and support renewable energy, leading to regulations and incentives for alternative fuel vehicles [Sissine, 2007]. Interest in green logistics, especially in the transport sector, has grown since it accounts for 26% of global CO₂ emissions and was the second-largest contributor in 2019. EVs are promising for reducing oil consumption and emissions, but challenges like limited range, battery degradation, and long charging times persist. Research on EVs in routing problems has increased, focusing on energy consumption and factors like speed and weight. Goeke (2015) tackled the E-VRP using mixed fleets with a detailed energy model. Asamer et al. [2016] explored non-linear energy usage with a dynamic model including motor efficiency and other variables. Fernández et al. [2022] applied this to an arc routing problem (E-ARP) with dynamic recharges and optimal travel speeds. Basso et al. [2019] used a linear mass function to calculate energy usage, considering topography and speed. Despite progress, many models remain simplified due to computational limitations. In this context, time-dependent travel times offer a more accurate representation of real-world conditions. Ichoua et al. [Ichoua et al., 2003] introduced VRP with time-dependent travel times (TDVRP) using piecewise constant speed functions. This can be extended to time-dependent speeds, resulting in non-linear energy consumption. However, applying time-dependency at the network level is challenging. Vidal et al. [2021] noted that complete graph models are often inadequate for ARPs, leading to the creation of the Time-dependent Capacitated Arc Routing Problem (TDCARP) with network-level speed functions.

We propose introducing EVs to TDCARP using an energy consumption model based on speed and time. EVs will service network edges without exceeding load and battery limits. Our non-linear ECR model, based on Asamer et al. [2016]; Fernández et al. [2022], avoids oversimplifications in speed and travel time. We will solve this new CARP variant with a metaheuristic approach, assuming constant vehicle mass. This is relevant for postal services, electric road sweepers, and other applications where capacity is volume-based rather than weight-based.

2. Problem Statement

Let $G = (V, E)$ be an undirected graph where V and E represent the set of vertices and edges, respectively. The vertex $0 \in V$ represents the depot, where a fleet of EVs with a load capacity of Q and battery capacity C in kWh are available at time 0. The set $E_R \subseteq E$ denotes edges that require service, where each service $u \in E_R$ has a non-negative demand q_u . Each edge $u \in E_R$ can be served in one of its two possible modes. To formalize this, an edge $u \in E_R$ with vertices $i, j \in V$ has a set of modes M_u , where $M_u = \{(i, j), (j, i)\}$. For each edge $(i, j) \in E$, d_{ij} is the edge distance, and there is also an instantaneous time-dependent speed function $v_{ij}(t) : [0, H] \rightarrow \mathbb{R}^+$ and an instantaneous traction force time-dependent function $F_{ij}(t) : [0, H] \rightarrow \mathbb{R}^+$.

The traction force, denoted as $F_{ij}(t)$, encompasses forces opposing the vehicle's movement. To advance and maintain speed, the vehicle must overcome these forces using its energy. Therefore, we must consider the instantaneous power, $P_{ij}(t)$, which measures the energy consumption at a specific time t on a given edge $(i, j) \in E$ and is calculated as $P_{ij}(t) = F_{ij}(t)v_{ij}(t)$.

Note that the arrival time depends on the departure time. We define this operation with the function Φ_{ij} for an edge $(i, j) \in E$, where the arrival time given a departure time t_i is $\Phi_{ij}(t_i)$. Consequently, the total energy consumption $e_{ij}(t_i)$ of a vehicle traveling along edge $(i, j) \in E$ from departure time t_i until arrival time $\Phi_{ij}(t_i)$ is given by the integral of the instantaneous power over this interval, as shown in Equation (1).

$$e_{ij}(t_i) = \int_{t_i}^{\Phi_{ij}(t_i)} P_{ij}(t) dt = \int_{t_i}^{\Phi_{ij}(t_i)} F_{ij}(t) v_{ij}(t) dt \quad (1)$$

The Electric Time-Dependent Capacitated Arc Routing Problem (E-TDCARP) seeks to design routes to minimize the sum of their duration, subject to the following constraints: each vehicle leaves the depot at the start time and must return to the depot, the total demand served of each vehicle cannot exceed its load capacity Q and the total energy consumption cannot exceed its battery capacity C , and the duration of each route does not exceed a set maximum value H . Finally, the FIFO property is considered when traveling and providing service, and waiting is never profitable, so this possibility is eliminated from our problem.

Arrival Time Function based on departure time. We employ the stepwise speed function, denoted as $v_{ij}(t)$, for directed edge $(i, j) \in E$, and the service speed function, $\hat{v}_{ij}(t)$, for directed edges with service requirements $(i, j) \in E_R$. According to these definitions, Equation (2) calculates the arrival time $\Phi_{ij}(t_i)$ at j , when departing from i at time t_i , and Equation (3) calculates the departure times $\Phi_{ij}^{-1}(t_j)$ at i , from i to j , arriving at time t_j .

$$\Phi_{ij}(t_i) = \left\{ x \left| \int_{t_i}^x v_{ij}(t) dt = d_{ij} \right. \right\} \quad (2)$$

$$\Phi_{ij}^{-1}(t_j) = \left\{ x \left| \int_x^{t_j} v_{ij}(t) dt = d_{ij} \right. \right\} \quad (3)$$

Since in the literature there are algorithms to solve them in 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}) is piecewise linear, continuous, monotonic, and can be represented in closed-form. Due to all the above, we will employ these algorithms to compute the arrival time.

Longitudinal Dynamic Model. The primary vehicle longitudinal dynamic model is based on Newton's second law, where the electric motor's traction force (F) causes the vehicle's movement [Asamer et al., 2016]. The calculation of the traction force is subject to the influence of the motor efficiency (η), aerodynamic resistance, rolling resistance, and grade resistance, among others [Asamer et al., 2016; Goeke and Schneider, 2015; Fernández et al., 2022], as shown in Equation (4). This formulation includes variables such as the mass (m), acceleration (a), aerodynamic drag coefficient (C_w), air density (ρ), the frontal area of the vehicle (A_F), speed (v), gravitational constant (g), rolling friction coefficient (C_r), and gradient angle (α).

$$F = \left(ma + \frac{\rho A_F C_w v^2}{2} + C_r mg \cos(\alpha) + mg \sin(\alpha) \right) \frac{1}{\eta} \quad (4)$$

In the context of time dependency, we represent speed as a function of time, which allows us to redefine v as $v_{ij}(t)$ for an directed edge $(i, j) \in E$ in Equation (4). Since we are dealing with a stepwise speed function, we establish an acceleration equal to zero (i.e., $a = \frac{v(t)}{dt} = 0$). In addition, we will use the assumption of the inexistence of inclination on the road ($\alpha = 0$). Therefore, in Equation (5), we establish the traction force in terms of time t for each directed edge $(i, j) \in E$ called $F_{ij}(t)$.

$$F_{ij}(t) = \left(\frac{\rho A_F C_w v_{ij}(t)^2}{2} + mg \right) \frac{1}{\eta} \quad (5)$$

3. Methodology

In this section, we introduce the assumptions related to topological factors and our preprocessing methods for the energy consumption functions based on departure time. We will also explain the preprocessing method for energy consumption functions for the Quickest Path between two vertices. Additionally, we will present our solution representation based on an indirect solution and introduce the equations for calculating exact time and energy consumption. Finally, we will detail the composition of our Iterated Local Search metaheuristic.

Assumptions. For this work, we assume that the vehicle's mass remains constant or does not change significantly. Furthermore, although traction force involves multiple variables, many of them remain constant or nearly constant under numerous situations. Therefore, we will focus solely on the influence of speed over time ($v_{ij}(t)$) in the traction force. This implies that for a traction force based on speed (v), denoted as $F'_{ij}(v)$, Equation (6) must be satisfied.

$$v_{ij}(t) = v, t \in [0, H] \Rightarrow F_{ij}(t) = F'_{ij}(v) \quad (6)$$

Preprocessing Energy Consumption Function based on departure time. In this section, we will focus on preprocessing $e_{ij}(t)$, which is computed by solving Equation (1). Each edge $(i, j) \in E$ has a $v_{ij}(t)$ that allows discrete values for the vehicle's speed over time. This enables us to compute $F_{ij}(t)v_{ij}(t)$ as a stepwise function based on time t , as demonstrated in Lemma 3.1.

Lemma 3.1. *An edge $(i, j) \in E$ has a stepwise speed function $v_{ij}(t)$ based on time (t) with h_{ij} pieces. Therefore, this edge has a stepwise function $P_{ij}(t) = F_{ij}(t)v_{ij}(t)$ with h_{ij} pieces.*

Proof. For a piece $p \in \{1, \dots, h_{ij}\}$ with interval time $[t_a, t_b]$, where the speed v^* is known in such interval, we establish

$$v_{ij}(t) = v^*, \forall t \in [t_a, t_b] \Rightarrow F_{ij}(t)v_{ij}(t) = F'_{ij}(v^*)v^* \quad (7)$$

In Equation (7), since v^* is known, the traction force $F'(v^*)$ is constant. If Equation (6) holds, we can replace $F_{ij}(t)$ with $F'_{ij}(v)$, making $F_{ij}(t)v_{ij}(t)$ constant in each piece p . Thus, $F_{ij}(t)v_{ij}(t)$ is a stepwise function with h_{ij} pieces. Note that $F_{ij}(t)v_{ij}(t)$ shares the same breakpoints as $v_{ij}(t)$. $\square$

Given the speed changes along an edge, we can redefine the integral in Equation (1) as a sum of integrals for each piece bounded by t_i and $\Phi_{ij}(t_i)$ in the $P_{ij}(t)$ function. The nearest upper breakpoint to the departure time is denoted as t_k (i.e., $\text{argmin}\{t_i < t_k, \forall k \in \{1, \dots, h_{ij}\}\}$), and the nearest lower breakpoint to the arrival time is denoted as t_l (i.e., $\text{argmax}\{t_l < \Phi_{ij}(t_i), \forall l \in \{1, \dots, h_{ij}\}\}$).

$$e_{ij}(t_i) = \begin{cases} \int_{t_i}^{\Phi_{ij}(t_i)} P_{ij}(t) dt & t_k = t_l \\ \int_{t_i}^{t_k} P_{ij}(t) dt + \sum_{n=k}^{l-1} \int_{t_n}^{t_{n+1}} P_{ij}(t) dt + \int_{t_l}^{\Phi_{ij}(t_i)} P_{ij}(t) dt & \text{otherwise} \end{cases} \quad (8)$$

Furthermore, each integral into Equation (8) is a constant value due to $P_{ij}(t)$ being a stepwise function. Thus, solving each integral of Equation (8), we obtain Equation (9), where a breakpoint, denoted as t_x , results in a discontinuity (i.e., $P_{ij}(t_x)$ can take two different values). Therefore, we define these two values as $P_{ij}(t_x^-)$ and $P_{ij}(t_x^+)$. They are the values found on the left-hand and the

right-hand side of $P_{ij}(t_x)$, respectively. If t_x is not a breakpoint, both values are equal. Thus, when we are performing forward integration, we choose to use $P_{ij}(t_x^+)$.

$$e_{ij}(t_i) = \begin{cases} P_{ij}(t_i^+)(\Phi_{ij}(t_i) - t_i) & t_k > t_l \\ P_{ij}(t_i^+)(t_k - t_i) + \sum_{n=k}^{l-1} P_{ij}(t_n^+)(t_{n+1} - t_n) + P_{ij}(t_l^+)(\Phi_{ij}(t_i) - t_l) & \text{otherwise} \end{cases} \quad (9)$$

This equation can be computed in the Algorithm 1, which iteratively computes energy consumption by traversing each piece involved in Equation (9) given a departure time t_i .

Algorithm 1: $e_{ij}(t_i)$

```

 $t_j, t \leftarrow \Phi_{ij}(t_i), t_i$ 
 $k \leftarrow \arg \min\{x \mid t_x > t_i\}$ 
 $l \leftarrow \arg \max\{x \mid t_x < t_j\}$ 
 $e \leftarrow 0$ 
for  $x \in k : l$  do
   $e \leftarrow e + P_{ij}(t_x^+)(t_x - t)$ 
   $t \leftarrow t_x$ 
 $e \leftarrow e + P_{ij}(t^+)(t_j - t)$ 
return  $e$ 

```

In metaheuristics, energy consumption queries can be inefficient. Algorithm 1 is computationally intensive due to its iterative process. To reduce this overhead, we propose Algorithm 2, which preprocesses $e_{ij}(t_i)$ into a closed-form piecewise linear function based on departure time t_i . To achieve this, we establish Lemma 3.2, which demonstrates that $e_{ij}(t)$ is a piecewise linear function with the same breakpoints as the closed-form arrival time function $\Phi_{ij}(t)$.

Lemma 3.2. $e_{ij}(t_i)$ is a piecewise linear function that has the same breakpoints as the closed-form arrival time function $\Phi_{ij}(t_i)$.

Proof. Given that $P_{ij}(t)$ is a stepwise function, its integral, $e_{ij}(t)$, is a piecewise linear function. It was demonstrated in [Aliprantis et al., 1999]. Let $P'(x) = \int_0^x P_{ij}(t)dt$, where $P'(x)$ is a piecewise linear function. $P'(x)$ has the same breakpoints as $P_{ij}(t)$. In Theorem 3.1, we establish that $P_{ij}(t)$ has the same breakpoints as $v_{ij}(t)$. Consequently, we rewrite Equation (10) as follows:

$$e_{ij}(t_i) = \int_{t_i}^{\Phi_{ij}(t_i)} P_{ij}(t)dt = \int_0^{\Phi_{ij}(t_i)} P_{ij}(t)dt - \int_0^{t_i} P_{ij}(t)dt = P'(\Phi_{ij}(t_i)) - P'(t_i) \quad (10)$$

Therefore, $e_{ij}(t_i)$ is piecewise linear as it is a composition of two piecewise linear functions (Φ and P'). Breakpoints can occur whenever t_i is a breakpoint of $P'_{ij}(t_i)$ or whenever t_i is a breakpoint of $\Phi_{ij}(t_i)$. Due to the breakpoints of $\Phi_{ij}(t_i)$ contain the breakpoints of $v_{ij}(t_i)$, $e_{ij}(t_i)$ has the same breakpoints as $\Phi_{ij}(t_i)$. $\square$

Algorithm 2 starts by storing all arrival time breakpoints A_{BP} for an edge $(i, j) \in E$. The set e_{BP} has the same breakpoints as A_{BP} . We calculate e for each breakpoint using Algorithm 1 and store the pieces in e_{PIECES} . This is done once per edge $(i, j) \in E$.

Algorithm 2: Closed-form construction of $e_{ij}(t_i)$

```

 $A_{BP} \leftarrow \text{GET\_BREAKPOINTS}(\Phi_{ij}(t_i))$ 
 $e_{BP} = \emptyset$ 
 $e_{BP} \leftarrow (0, e_{ij}(0))$ 
for  $t_i \in A_{BP}$  do
   $\lfloor e_{BP} \leftarrow (t_i, e_{ij}(t_i))$ 
 $e_{PIECES} = \emptyset;$ 
for  $x \in \{1, \dots, \text{SIZE}(e_{BP}) - 1\}$  do
   $\lfloor e_{PIECES} \leftarrow (e_{BP}[x], e_{BP}[x + 1])$ 
return  $e_{PIECES}$ 
  
```

Finally, the computation of the closed-form energy consumption function of all edges must be done in a phase prior. We will use these values for preprocessing energy consumption on deadhead paths, where the vehicle travels without providing any service. This will be discussed in the next section.

Energy Consumption Function for a Deadhead Path. In arc routing problems, deadhead paths constitute segments of a vehicle's route where it travels between consecutive service locations without performing any tasks or serving customers. Consider traversing the edges $(i, j) \wedge (j, k) \in E$ in sequence. The energy consumption for the first edge depends solely on the vehicle's departure time t_i , denoted as $e_{ij}(t_i)$. For the second edge, the departure time is the vehicle's arrival time at node j , represented as $\Phi_{ij}(t_i)$. Hence, the energy consumption for the second edge is $e_{jk}(\Phi_{ij}(t_i))$. The total energy consumption for traversing both edges is the sum of these individual consumptions, as shown in Equation (11).

$$e_{(i,j) \oplus (j,k)}(t_i) = e_{ij}(t_i) + e_{jk} \circ \Phi_{ij}(t_i) \quad (11)$$

In this equation, $\circ$ represents the composition of two functions (i.e., $e_{jk} \circ \Phi_{ij}(t_i) = e_{jk}(\Phi_{ij}(t_i))$). Additionally, Equation (11) maintains a piecewise linear structure in computing the energy consumption function based on departure time t_i for a given deadhead path. Many routing problems on networks often compute these paths in an initial phase based on different objectives, such as distance or time minimization. Therefore, it's feasible to compute their energy consumption function in a preliminary phase as well. For our problem, we propose an adaptation of the Quickest Paths preprocessing method from Vidal et al. [2021], as it is necessary to incorporate our energy consumption function for deadhead paths. Algorithm 3 presents the adaptation, where $\Psi_{ij}(t_i)$ represents the closed-form arrival time function for each destination j for all departure times t from the origin i .

Algorithm 3 starts by initializing the values of the functions Ψ and e for each pair (i, j) , where “id” represents the identity function ($\Psi_{ij}(t_i) = t_i$). In each iteration, our algorithm calculates the lower envelope of the Ψ'_{ik} and $\Phi_{jk} \circ \Psi_{ij}$ functions, with the goal of minimizing total travel time. Within the framework of this objective, our primary focus is to obtain the energy consumption function for each point along the lower envelope. To accomplish this task, we introduce the function ENERGY SELECTION. This function acts as a piece selector, selecting energy consumption function based on the lower envelope. This algorithm terminates until no pair (i, j) , $\forall j \in V$ can modify the current lower envelope (i.e., $\Psi_{ij} = \Psi'_{ij}$, for all $j \in V$).

The lower envelope function will lead to a discontinuous function for energy consumption because the quickest path may change over time. Therefore, in order to obtain the energy consumption function for a departure time t_i , we must choose the quickest path with the lowest

Algorithm 3: Quickest paths from i for all starting times

```

for  $j \in V$  do
   $\Psi'_{ij} \leftarrow (id \text{ if } i = j, \text{ else } \infty)$ 
   $e'_{ij} \leftarrow (0 \text{ if } i = j, \text{ else } \infty)$ 
 $L \leftarrow \{i\}$ 
while  $L \neq \emptyset$  do
  for  $(j, k) \in E : j \in L$  do
     $\Psi'_{ik} \leftarrow \text{LOWER ENVELOPE}(\Psi'_{ik}, \Phi_{jk} \circ \Psi_{ij})$ 
     $e'_{ik} \leftarrow \text{ENERGY SELECTION}(\Psi'_{ik}, e'_{ik}, e_{(i,j)} \oplus (j,k))$ 
   $L \leftarrow \emptyset$ 
  for  $y \in V$  do
    if  $\Psi_{ik} \neq \Psi'_{ik}$  then
       $L \leftarrow L \cup y$ 
       $\Psi_{ik}, e_{ik} \leftarrow \Psi'_{ik}, e'_{ik}$ 
return  $\Psi, e$ 

```

energy consumption. To achieve this, we will use Equation (12).

$$e_{ij}(t_i) = \min \{e_{ij}(t_i^-), e_{ij}(t_i^+)\} \quad (12)$$

In Equation (12), if t_i is a breakpoint of the energy consumption, $e_{ij}(t_i)$ can take two different values. Therefore, we define these two values as $e_{ij}(t_i^-)$ and $e_{ij}(t_i^+)$. They are the values found on the left-hand and the right-hand side of $e_{ij}(t_i)$, respectively. If t_i is not a breakpoint, both values are equal.

3.1. Metaheuristic Approach

Indirect Solution and Search Space. The indirect solution representation decomposes an optimization problem into two sets of decision variables: one explicitly determined and the other completed with an optimal decoder (exact or heuristic) [Vidal et al., 2021]. This simplifies the solution process and reduces heuristic search approximations. In TDCARP, the decision classes are: assigning services to vehicles, sequencing services within routes, choosing service modes, and choosing paths between services. Each decision set grows exponentially with the number of services. To optimize mode decisions, we represent solutions as sequences of services (routes) without mode information and use a dynamic programming decoding algorithm. Bellman's algorithm is used for this propagation [Vidal et al., 2021].

Our solution representation uses time-dependent travel times and closed-form functions e and $\hat{e}$ to calculate energy consumption during the quickest path and service, respectively. Evaluations require propagating arrival times, service completion times, and energy consumption in the auxiliary graph shown in Figure 1. This graph includes Quickest Path Ψ (solid arrows) and service time functions $\hat{\Phi}$ (dotted arrows). Formally, a route σ is defined as a sequence of services starting and ending at the depot, with $\sigma(1)$ and $\sigma(|\sigma|)$, respectively.

Vidal et al. [2021] present the computation of the completion time $T_{\sigma(i)}^{\text{EXACT}}[l]$ for each service $\sigma(i)$ and each mode $l \in M_{\sigma(i)}$ in a route σ . This calculation is done using Equation (13), where $\Psi_{ij}^{kl}(t)$ represents the arrival time of the vehicle leaving the end of service i in mode k at time t towards the origin of service j in mode l , and $\hat{\Phi}_i^l(t)$ represents the completion time of service i in

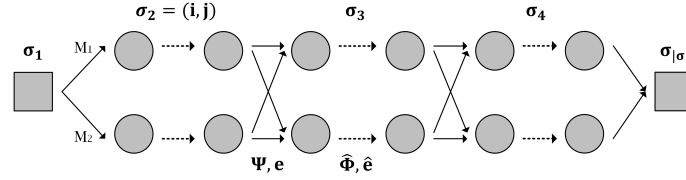


Figure 1: Indirect solution representation.

mode l starting at time t . $T_{\sigma(|\sigma|)}^{\text{EXACT}}[1]$ provides the duration of the entire route.

$$T_{\sigma(i)}^{\text{EXACT}}[l] = \begin{cases} 0 & i = 1 \\ \min_{k \in M_{\sigma(i-1)}} \left\{ \hat{\Phi}_{\sigma(i)}^l \left(\Psi_{\sigma(i-1)\sigma(i)}^{kl} \left(T_{\sigma(i-1)}^{\text{EXACT}}[k] \right) \right) \right\} & \text{otherwise} \end{cases} \quad (13)$$

Given that our objective is to minimize travel time, Equation (13) selects the best mode, denoted as $k' \in M_{\sigma(i-1)}$, which results in the quickest completion time for the current service $\sigma(i)$. Therefore, it is not necessary to find the path with the lowest consumption to calculate the exact total energy consumption $E_{\sigma(i)}^{\text{EXACT}}[l]$ until the completion of service $\sigma(i)$ in mode $l \in M_{\sigma(i)}$. Instead, we use Equation (14), where $E_{\sigma(i)}^{\text{TIME}}[k', l]$ represents the total energy consumption from the quickest completion time of the previous service $\sigma(i-1)$ (computed using the best mode $k' \in M_{\sigma(i-1)}$) to the completion time of the current service $\sigma(i)$ in mode $l \in M_{\sigma(i)}$. The total energy consumption for the entire route is given by $E_{\sigma(|\sigma|)}^{\text{EXACT}}[1]$.

$$E_{\sigma(i)}^{\text{EXACT}}[l] = \begin{cases} 0 & i = 1 \\ E_{\sigma(i-1)}^{\text{EXACT}}[k'] + E_{\sigma(i)}^{\text{TIME}}[k', l] & \text{otherwise} \end{cases} \quad (14)$$

To calculate $E_{\sigma(i)}^{\text{TIME}}$, we use equations (15). The first term represents the energy consumption for traveling from the previous service $\sigma(i-1)$ in mode k' to the current service $\sigma(i)$ in mode l , departing at time $T_{\sigma(i-1)}^{\text{EXACT}}[k']$. The second term corresponds to the energy consumption incurred during the execution of the current service $\sigma(i)$, starting at the arrival time $\Psi_{\sigma(i-1)\sigma(i)}^{k'l} \left(T_{\sigma(i-1)}^{\text{EXACT}}[k'] \right)$.

$$E_{\sigma(i)}^{\text{TIME}}[k', l] = e_{\sigma(i-1)\sigma(i)}^{k'l} (T_{\sigma(i-1)}^{\text{EXACT}}[k']) + \hat{e}_{\sigma(i)}^l \left(\Psi_{\sigma(i-1)\sigma(i)}^{k'l} \left(T_{\sigma(i-1)}^{\text{EXACT}}[k'] \right) \right) \quad (15)$$

Iterated Local Search. Due to the NP-hardness of CARP and its extensions, heuristic and metaheuristic approaches are the most effective [Corberán and Prins, 2010]. We use an Iterated Local Search (ILS) metaheuristic for its ability to address complex problems while keeping the model simple [Lourenco et al., 2003]. Our ILS employs penalties to manage infeasible solutions, allowing exploration of new search spaces. If an infeasible solution is found, we return the best one. We also implement restarts to reset penalties and acceptance criteria, diversifying the search space beyond the classic ILS approach. Lower bounds on move evaluations proposed by Vidal et al. [2021] are used to filter out a large portion of moves in the Local Search (LS).

Initial Solution. The initial solution is built using a random-constructive Constructive Heuristic. We shuffle the required edges into a list, then open an empty path and start inserting each edge. A new route is opened when the current route cannot include another edge due to load capacity. We also check if it is possible to insert an edge into any previous routes before opening a new one.

Adaptive Penalty. The cost of a feasible route includes its duration and penalties P for load capacity (L), battery capacity (B), and time limits (T). Each constraint $i \in \{\text{load, battery, time}\}$ has

a penalty coefficient p_i that adjusts based on the difference between the route's value and the limit. Penalties have base (p_i^{base}), minimum (p_i^{min}), and maximum (p_i^{max}) values, along with reduction (δ_1) and increase (δ_2) coefficients. In each ILS iteration, if the candidate, local minimum, and best solutions satisfy the constraints, penalties are minimized. If the local minimum fails but the best solution meets the constraints, penalties are reduced by δ_1 . If neither the best nor candidate solutions satisfy the constraints, penalties increase by δ_2 . Costs of stored solutions are updated with new p_i values in each iteration.

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 . To minimize travel times, we use the minimum travel time as our criterion. Each endpoint of two edges $(i, j) \wedge (k, l) \in E$ is evaluated to find the minimum travel time. We then select the Γ clients based on this criterion. The closest neighbors of (i, j) are $(k, l) \in \Gamma$, and the granularity parameter Γ limits neighborhood size. Movements are explored in random order of indexes i and j , and any improvement is applied immediately [Vidal et al., 2021].

$$\min \left\{ \min_{t \in [0, H]} \{ \Psi_{xy}(t) - t \} \mid \forall x \in [i, j], y \in [k, l] \right\} \quad (16)$$

Local Search. The LS procedure includes preprocessing, concatenation, and lower bounds techniques. During each improvement step, the time structures of the lower bounds between all service pairs (sub-sequences) of each route are updated. Our neighborhood structure consists of six moves: Swap-inter, Swap-intra, Crossover, Relocate-inter, Relocate-intra, and 2-Opt, executed in random sequence.

Acceptance Criteria. The *better* acceptance criterion accepts only solutions better than the current one, but it can get stuck in a local minimum. As a result, we employ the simulated annealing (SA) method that Pisinger and Ropke [2007] proposed, which occasionally accepts worse solutions instead of always the better.

Perturbation. Perturbation involves executing p random movements. The neighborhoods or movements used are the same as for the local search, and we also include an open route function to handle infeasible solutions.

4. Computational Results

The preprocessing and metaheuristic methods were coded in *Julia* 1.7. The tests were carried out in an Intel Core i7-8700K CPU @ 3.7GHz (12 cores) with 64 GB of RAM. All the tests carried out were run ten times with different seeds. Table 1 presents the energy parameters associated with the consumption rate based on the specifications of an Eforce EF18 electric truck and an IVECO Stralis conventional truck. For the local search, we use the following parameters, adjusted through experiments on type C instances, to achieve the shortest average travel time. We conduct 100 iterations with 20 restarts. In swap and relocate movements, we limit sizes to up to five clients ($k \leq 5$). Three local search movements ($p = 3$) are applied during perturbation without restrictions. We use only the initial ($P_i = 0.215$) and final probability ($P_f = 0.005$) parameters for the acceptance criteria with SA. A granularity of forty ($\Gamma = 40$) required arcs is set, restricting the neighborhood search to movements.

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 $\delta_1 = 0.99$ as long as a better solution is found and this minimum value is not exceeded. Otherwise, the value increases by the increment coefficient $\delta_2 = 1.5$ as long as the value does not exceed the maximum value of 10^4 .

Table 1: Energy parameters

Parameter	Unit	Symbol	Value
Auxiliary power	kW	P_0	3
Weight	kg	m	18000
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

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 create 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]. We set the number of buckets to 200.

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 Fernández et al. [2022], the Eforce EF18 can use batteries up to 630 kWh.

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 2 reports the total CPU time spent by the algorithm for each of the three instance categories, where *Eglese* 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.

The E-TDCARP was initially solved without considering battery capacity limits. However, the obtained solutions were later re-evaluated to assess their feasibility under fixed battery capacities. This experiment is crucial as it highlights the impact of battery capacity on solution viability. Initially ignoring battery capacity allows for optimal solutions that might not be feasible in practice due to energy constraints. Re-evaluating these solutions with a fixed battery capacity determines their continued viability and the percentage of feasible solutions for each battery capacity. Table 3 shows the percentage of feasible solutions for each instance after re-evaluating the optimal solutions without considering battery capacity, tested with capacities of 100, 200, and 300 kWh. For capacities above 300 kWh, the battery no longer renders TDCARP solutions infeasible.

Subsequently, the E-TDCARP was solved including battery capacity constraints. Results shown in Table 3 indicate that for battery capacities between 100 and 300 kWh, the solutions often became infeasible due to battery constraints. We found a greater number of feasible solutions within the same set of instances when considering battery constraints. However, the number of feasible solutions varied with each model, indicating that the choice of energy consumption model significantly influences the results. Notably, for a vehicle battery capacity of 100 kWh, no feasible solutions were found, underscoring the importance of analyzing vehicle type selection in the preliminary stages.

In Table 4, we present a summary of the results for the E-TDCARP with battery constraints. We observe that vehicles with 200 kWh batteries have an increased average cost. This is

Table 2: Performance of the Quickest path and energy consumption preprocessing Algorithm.

Instance	$ V $	$ E_R $	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
Avg. C			67.65	71.70	77.33
egl-e1	77	51	122.78	135.18	156.89
egl-e2	77	72	131.46	145.82	156.07
egl-e3	77	87	129.67	138.37	156.41
egl-e4	77	98	129.07	144.01	162.78
egl-s1	140	75	1041.69	1033.06	823.26
egl-s2	140	147	981.75	983.50	826.31
egl-s3	140	159	997.55	995.85	865.70
egl-s4	140	190	1100.64	976.98	888.11
Avg. egl			579.33	569.10	504.44

Table 3: Performance of the ILS on E-TDCARP instances.

Instance		Percentage of feasible solutions for each battery level (kWh)					
		Without Battery Capacity			With Battery Capacity		
Name	TD	100	200	300	100	200	300
C	Low	0	90	100	0	100	100
egl	Low	0	16.3	100	0	37.5	100
C	Medium	0	79	100	0	100	100
egl	Medium	0	3.8	100	0	37.5	100
C	High	0	69	100	0	100	100
egl	High	0	0	100	0	25	100

because the smaller battery cannot accommodate the TDCARP solution and requires routes that are longer in terms of time but more energy-efficient. It is also important to note that not all instances in each group have found a viable solution. This information highlights the crucial importance of considering battery sizes in route planning. A very small battery produces unfeasible solutions, while a very large battery incurs extra costs. Additionally, extremely large batteries imply larger vehicles, which are often unsuitable for many activities, especially last-mile delivery.

Table 4: Performance of the ILS on E-TDCARP

Instance		TDCARP	Battery Capacity (kWh)					
			200			300		
Name	TD	Avg. Best	Avg. Best	% Δ Cost	T(s)	Avg. Best	% Δ Cost	T(s)
C	Low	2138.28	2150.59	0.55%	422.74	2144.15	0.21%	378.57
egl	Low	3383.01	2351.00	1.24%	431.95	3179.55	0.95%	1002.20
C	Medium	2173.41	2179.93	0.25%	490.43	2175.84	0.09%	446.99
egl	Medium	3218.54	2440.75	3.01%	503.70	3266.83	1.06%	1081.82
C	High	2182.75	2202.14	0.79%	628.20	2186.65	0.15%	541.80
egl	High	3272.24	2687.35	3.05%	714.81	3315.37	0.95%	1185.93

5. Conclusions and Future works

This paper introduces E-TDCARP, an extension of TDCARP that captures non-linear energy consumption based on speed and travel time. We propose a closed-form energy consumption

preprocessing method and an ILS + SA metaheuristic with restarts and adaptive penalties to solve the problem. The goal is to design routes that minimize total duration while serving a finite number of arcs and edges. Experiments showed that ignoring energy consumption can lead to infeasible solutions. By implementing the E-TDCARP constraint, we demonstrated feasible route design even with battery constraints.

Future research could involve designing exact algorithms and more efficient genetic and hybrid metaheuristics for E-TDCARP to improve solution quality and reduce computational time. Another direction is addressing E-TDCARP with energy consumption rates based on speed and mass, where mass increases only during service, remaining constant on deadhead paths. Predictive models or overestimated mass increases can be used. Additionally, incorporating electricity prices into the objective function could enhance analysis and design by considering paths that save energy. Adapting the Quickest Path Preprocessing Algorithm to minimize energy consumption could help achieve this.

References

- Aliprantis, C. D., Border, K. C., Aliprantis, C. D., and Border, K. C. (1999). Integrals. *Infinite Dimensional Analysis: A Hitchhiker's Guide*, p. 395–426.
- Asamer, J., Graser, A., Heilmann, B., and Ruthmair, M. (2016). Sensitivity analysis for energy demand estimation of electric vehicles. *Transportation Research Part D: Transport and Environment*, 46:182–199.
- Basso, R., Kulcsár, B., Egardt, B., Lindroth, P., and Sanchez-Diaz, I. (2019). Energy consumption estimation integrated into the electric vehicle routing problem. *Transportation Research Part D: Transport and Environment*, 69:141–167.
- Corberán, A. and Prins, C. (2010). Recent results on arc routing problems: An annotated bibliography. *Networks*, 56(1):50–69.
- Fernández, E., Leitner, M., Ljubić, I., and Ruthmair, M. (2022). Arc routing with electric vehicles: dynamic charging and speed-dependent energy consumption. *Transportation Science*, 56(5): 1219–1237.
- Goeke, D. and Schneider, M. (2015). Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81–99. ISSN 0377-2217. URL <https://www.sciencedirect.com/science/article/pii/S0377221715000697>.
- Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European journal of operational research*, 144(2):379–396.
- Lourenco, H. R., Martin, O., and Stutzle, T. (2003). Iterated local search. handbook of metaheuristics. f. glover and g. kochenberger.
- Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435.
- Sissine, F. J. (2007). *Energy Independence and Security Act of 2007: a summary of major provisions*. Congressional Research Service Washington, DC.
- Vidal, T., Martinelli, R., Pham, T. A., and Hà, M. H. (2021). Arc routing with time-dependent travel times and paths. *Transportation Science*, 55(3):706–724.