Computers & Industrial Engineering 176 (2023) 108987

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Finding an energy efficient path for plug-in electric vehicles with speed optimization and travel time restrictions ☆

Bilgenur Erdoğan [a,*], Mustafa Kemal Tural [a], Arsham Atashi Khoei [b]

[a] Department of Industrial Engineering, Middle East Technical University, Ankara, Turkey
[b] School of Management, University of Bath, Bath, United Kingdom

ARTICLE INFO

Keywords:
Plug-in electric vehicle
Minimum cost path problem
Second order cone programming
Matheuristic
Iterated local search
Speed optimization

ABSTRACT

Transportation is one of the main factors when global total energy consumption is considered and is a significant contributor to emissions of harmful gases including carbon dioxide (CO_2). Due to their lower tailpipe CO_2 emissions compared to the vehicles with internal combustion engines, electric vehicles provide an opportunity to reduce environmental impacts of transportation. In this direction, a problem for plug-in electric vehicles (PEVs) is studied where the aim is to find an energy efficient path. Given an origin–destination pair over a directed network, this problem involves determining a path joining origin and destination, the speed of the PEV on each road segment, i.e., arc, along the path, the charging stations the PEV will stop by, and how much to recharge at each stop so as to minimize the total amount energy consumption. There are speed limits on each road segment, and PEV has to arrive at the destination on or before a given total time limit. For this problem, firstly, a mixed-integer second order cone programming formulation (MISOCP) is proposed. Secondly, to be able to solve larger size instances, a matheuristic is developed. Lastly, an iterated local search (ILS) algorithm is designed for this problem. Solution quality and computation times of the heuristics and the exact algorithm are compared on different instances. Differently from the literature, the speed values of the PEV on the arcs are considered as continuous decision variables in all proposed solution approaches. Moreover, consideration of the speed limits which can be legal limits or limits imposed by congestion makes our problem more realistic. The analysis of the results of the computational experiments gives the user an insight to select the proper solution approach based on the instance settings. MISOCP formulation becomes inadequate for larger instances. On the other hand, the heuristic solution approaches can solve such instances within reasonable computational times and therefore they have the potential to be integrated in some software to dynamically find energy efficient paths.

1. Introduction

In recent years, due to increased environmental and social awareness, sustainability related studies have gained popularity. Due to their adverse effects to the environment, greenhouse gas emission reduction has been a major concern in several areas of studies. A recent study shows that transportation is responsible for 24% of global total energy consumption (IEA, UNSD, et al., 2019). Hence, governments start to take preventive actions to reduce transportation originated harmful gases which cause global warming and extinction of living creatures in the long term. For example, Energy Union put some obligations and targets on emissions for new cars produced in European countries between 2025 and 2030, and it is estimated that these limitations will provide 15% to 37.5% reductions in emissions (Regulation, 2018). In this direction, the use of electric vehicles has a great potential due to their lower tailpipe carbon dioxide emissions compared to vehicles with internal combustion engines. By using fewer vehicles or adopting advanced engine technologies, e.g., electric engines, transportation related energy consumption and emissions sourced by transportation can be reduced.

There are three types of electric vehicles that are currently in use worldwide. The first one is the plug-in electric vehicle (PEV) which uses electricity as its only energy resource to recharge its limited-capacity batteries. The second one is the hybrid electric vehicle (HEV) which uses two types of energy resources, electricity and fuel, and two types of engines, electric and internal combustion. In HEVs, the mechanical energy, when available, is converted to electric energy to recharge the batteries. The third one is the plug-in hybrid electric vehicle (PHEV). Similar to HEVs, PHEVs also include both internal combustion and

☆ The first author, Bilgenur Erdoğan, was partially supported by TÜBİTAK 2210/A National Graduate Scholarship Program.
* Corresponding author.
E-mail address: bilgenur.guru@metu.edu.tr (B. Erdoğan).

https://doi.org/10.1016/j.cie.2023.108987
Received 27 July 2022; Received in revised form 23 November 2022; Accepted 5 January 2023
Available online 7 January 2023

Table 1
Electric vehicle types.

Vehicle type	Engine type	Energy outsource
Plug-in Electric Vehicle (PEV)	Electric engine	Electricity
Hybrid Electric Vehicle (HEV)	Electric engine Internal combustion engine	Fuel
Plug-in Hybrid Electric Vehicle (PHEV)	Electric engine Internal combustion engine	Electricity & Fuel

electric engines. Differently, PHEVs can be recharged by plugging in like PEVs. The classification described in this paragraph is summarized in Table 1.

Electric vehicles have several economic, environmental, and social benefits including reduced air and noise pollution. However, they have also some limitations for users. PEV users need to have an understanding of driving ranges, driving costs, and tax-related issues, e.g., governmental incentives. With their limited battery capacities, PEVs cannot make long distance trips without recharging. Thereby, the PEV users need extensive information about the recharging stations' locations to plan their trips. On the other hand, HEVs provide longer driving ranges as they can recharge themselves while driving. HEV users visit only the fuel refilling stations when needed. PHEVs can have even more driving ranges since they can be plugged in to recharge the batteries when stopped at a charging station. Thereby, PHEV users may use both electric recharging stations and fuel refilling stations.

PEVs have low life cycle energy consumption and GHG emissions even if the charging and battery production are taken into account and do not have any exhaust pipe (Tiwari, Aditjandra, & Dissanayake, 2020). Another interesting fact is that when all types of electric vehicles were commercially available in the U.S. for the first time in 2011, 18,000 electric vehicles were sold among which more than 50% were PEVs (Krause, Carley, Lane, & Graham, 2013). Their usage still has a great potential to grow in the near future.

In this study, we focus specifically on PEVs as path planning is more critical in PEVs than other electric vehicles. As far as our knowledge, there is a gap in the literature in terms of energy efficient path construction and continuous speed optimization for electric vehicles. Thereby, we introduce a problem that seeks the most energy efficient path of a PEV which is to travel from a predetermined origin node to a destination node in a directed network within a given total time limit. There are lower and upper speed limits on each arc limiting the vehicle speed and recharging stations at some of the nodes of the network which can be visited by the PEV to get recharged. In addition to contributing to the literature by considering an energy efficient path problem for PEVs; namely, the PEVEEP, we also make contributions in terms of solution approaches. We propose exact and heuristic solution approaches which involve novel aspects including the use of mixed-integer second order cone programming. The PEVEEP includes the following decisions: finding the most energy efficient path joining the origin and destination such that the given total time limit is not exceeded; determining the speed of the PEV on each arc along the chosen path; and determining where to stop to recharge the battery and how much to recharge. Here, the total time limit refers to the restriction that limits the amount of time between the departure from the origin and arrival to the destination, i.e., the sum of the total travel time and the time spent for recharging the battery.

A mixed-integer second order cone programming (MISOCP) formulation is provided to solve the PEVEEP. As this formulation is inadequate in solving large size instances to optimality within reasonable times, a matheuristic and an iterated local search (ILS) algorithm are developed for the problem. The solution methods are compared in terms of solution quality and computational time.

The rest of the paper is organized as follows: we first review the related literature in Section 2. Then, we provide the description of the PEVEEP and the procedure for the energy calculation of an electric vehicle in Section 3. In Section 4, we formulate the PEVEEP as an MISOCP problem. The matheuristic and ILS algorithm proposed for the PEVEEP are described in Section 5. Computational experiments are discussed in Section 6. Finally, conclusions and future directions are given in Section 7.

2. Literature review

In this section, we provide a literature review of studies on minimum cost path problems with electric vehicles, the electric vehicle routing problems, the behavior of electric vehicle batteries considered in these problems, and some related classical vehicle routing problems.

First, studies including minimum cost path problems with electric vehicles (EVs) are reviewed. These problems have been studied widely in recent years where the aim is to find the most desirable path (with respect to some objective such as energy consumption, cost, or number of stops) of an EV that joins the origin and destination nodes. Sachenbacher, Leucker, Artmeier, and Haselmayr (2011) introduce an extension of the classical shortest path problem which looks for the energy-optimal route of an EV, joining the origin and destination nodes. The problem is formulated as a shortest path problem on an energy network for HEVs with energy recuperation options on the arcs. The cost of each arc on this network is defined as the energy consumption along the arc. The authors provide a heuristic which is a modified version of the Dijkstra's Algorithm (Dijkstra et al., 1959) and implement it on a real-life instance. To the best of our knowledge, this is the first study that aims to construct an energy efficient route for electric vehicles. Sweda and Klabjan (2012) study a minimum cost path problem for EVs which is derived from the refueling needs of fuel-powered vehicles. In this problem, the EV may stop and recharge its battery at the nodes of the network to arrive at the destination. Cost terms are traveling cost, recharging cost, and a function of the charge level of the battery at each node of the network. Backward recursion and approximate dynamic programming are proposed as the solution methods whose performances are not evaluated with computational experiments. Arslan, Yıldız, and Karaşan (2015) study a minimum cost path problem where the total cost of a PHEV on its path from the origin node to the destination is composed of the electricity cost, gasoline cost, battery degradation cost, vehicle depreciation cost, and the stopping cost. A mixed-integer quadratically constrained programming (MIQCP) formulation, a discrete approximation dynamic programming heuristic, and a shortest path heuristic are proposed for the problem. The shortest path heuristic is composed of two steps. In the first step, the shortest path from the origin to the destination is found. In the second step, the MIQCP formulation is solved for the found shortest path to decide on where to stop along the shortest path to recharge and how much to recharge. Strehler, Merting, and Schwan (2017) study the problem of finding a minimum-cost route between the origin and destination nodes where the vehicle is allowed to go over an arc of the network more than once. Recharging options are available at the nodes and the arcs of the network for PEVs and HEVs, respectively. The objective is to minimize the total travel time including recharging times for PEVs, and total fossil fuel consumption over the network for HEVs. In all studies covered in this paragraph speed decisions are not incorporated, and energy consumption of EVs are not explicitly considered in the objective functions.

Kucukoglu, Dewil, and Cattrysse (2021) provide a comprehensive literature review of the electric vehicle routing problems (EVRPs).

Different objectives have been considered in EVRPs such as total travel distance, total travel time, total number of stations used, total recharging cost, and total energy consumption. In most of the studies, energy consumption of an EV is computed as a function of travel distance only; whereas, in real-life, it is a function of not only travel distance but also some other parameters such as vehicle load, vehicle speed, vehicle characteristics, and road conditions. The most commonly used exact solution methods for the EVRPs are mathematical programming formulations, dynamic programming, column generation, and branch and bound approaches. On the other hand, the (adaptive) large neighborhood search (LNS) and iterated local search (ILS) algorithms are the most commonly used heuristics for EVRPs. Recently, Liu et al. (2021) provide a survey of energy management strategies for HEVs and PHEVs. Factors affecting the energy consumption of these vehicles including travel distance, travel time, average speed, and average acceleration are discussed in addition to the urban and rural road conditions. The survey also includes a list of solution methodologies for the energy management problem of HEVs and underline that artificial intelligence plays a significant role in the development of energy systems of HEVs.

An electric vehicle routing problem (EVRPTW) is defined for the first time by Schneider, Stenger, and Goeke (2014). The authors study a single PEV routing problem with time windows where the vehicle can recharge its electric battery at some stations on its route to be able to complete its tour. A mixed-integer linear programming (MILP) formulation is developed to minimize the total distance traveled by the PEV and to determine the stations at which the PEV stops to recharge. Moreover, a heuristic combining two metaheuristics, that are variable neighborhood search (VNS) and tabu search (TS), is proposed. A time-dependent electric vehicle routing problem with a fleet of PEVs is studied by Lu, Chen, Hao, and He (2020). The authors consider time-dependent traffic congestion on the arcs which affects the speeds of the vehicles and provide an MILP formulation for the problem whose objective function consists of three parts: energy consumption cost, drivers' wages, and acquisition cost of the vehicles. In this formulation, the possible speed values belong to a discrete set. In addition to the exact formulation, an iterated variable neighborhood search (IVNS), a variable neighborhood descent method (VND), and a speed optimization method are proposed. Bruglieri, Pezzella, Pisacane, and Suraci (2015) study the EV routing problem with time windows. In this problem, the objective minimizes a function of the number of vehicles used and the total travel times of the vehicles. The authors provide an MILP formulation and a variable neighborhood search branching matheuristic for the problem. Montoya, Guéret, Mendoza, and Villegas (2017) propose an EV routing problem with nonlinear recharging functions for the first time which is approximated by piecewise linear functions to develop an MILP formulation for the problem. Bac and Erdem (2021) study an extension of the EVRPTW with a heterogeneous EV fleet where partial recharging is allowed. A mixed integer linear programming formulation is proposed in which the objective is the minimization of total charging time, travel time, and penalty cost of unscheduled jobs, overtime, and the violation of the time windows. Two heuristic approaches; namely, the VNS and VND, are proposed for the problem which perform well even on large size instances. Recently, an extension of the EVRPTW is also studied by Zhou and Zhao (2022). The problem considers battery swap decisions under time window restrictions. The provided mathematical formulation aims to minimize total costs and maximize the average utilization of the batteries. Differently from the literature, PEVs visit the battery swap stations for power replenishment. A whale optimization algorithm, a nature-inspired metaheuristic, is proposed for this multi-objective problem.

Abousleiman and Rawashdeh (2014) study an energy efficient routing problem for EVs and propose particle swarm optimization for its solution. The authors argue that the traditional shortest path algorithms (those that find paths with the least travel time or distance) fail to find energy efficient routes. Abousleiman, Rawashdeh, and Boimer (2017) propose an ant colony optimization metaheuristic for an energy efficient routing problem of EVs. They use the routes suggested by Google Maps or MapQuest, and compare them with the routes constructed by the proposed metaheuristic. It is shown that in several cases, the solutions of the proposed metaheuristic yields significant savings in the energy consumption of EVs. The objective function in both of these papers is the minimization of total net energy consumption which is calculated by taking the difference between the total energy consumption and the energy gained with the regenerative braking on each arc.

A simulated annealing algorithm for a green vehicle routing problem with time windows for PHEVs is proposed by Vincent, Redi, Hidayat, and Wibowo (2017). This problem minimizes the electric energy and fuel consumption cost considering the limited availability of electric charging and fuel stations on the network. In a recent study, Ma, Hu, Chen, Wang, and Wu (2021) consider a vehicle routing problem for shared autonomous electric vehicles which can swap their batteries at certain stations. The objective minimizes a function of travel distance, total time, and energy consumption of all vehicles under given tour time limitations. An MILP formulation where speed is a discrete decision variable, and an adaptive large neighborhood search heuristic combined with a speed optimization algorithm are proposed. This is one of the few studies in the literature that controls environmental impacts of EVs via speed optimization.

There are different models in the literature for energy consumption estimation of PEVs. In the study by De Cauwer, Van Mierlo, and Coosemans (2015), an estimation function is suggested based on the real world data. The proposed energy consumption model considers energy loss due to heating and air-conditioning systems as well as energy requirement at the wheels of a PEV which includes five parts: the rolling resistance, potential energy, aerodynamic losses, kinetic energy, and the energy needed for the acceleration of rotational parts. Wu, He, Yu, Harmandayan and Wang (2015) solve an analytical model to determine time-dependent optimal speed profile for an EV such that the electricity usage along the path considering route characteristics and traffic conditions is minimized. It is observed that the energy consumption is lower with smoother acceleration and deceleration values. The authors use the parameters provided by Wu, Freese, Cabrera and Kitch (2015) who establish a data collection system, which is installed in a test EV, in order to specify the values of vehicle weight, resistance of motor, radii of tires etc. Finally, in a recent study, Li et al. (2017) provide an estimation of the energy consumption behavior of the EV battery which is the one that is used in our study. In this model, the energy consumption of an EV is a function of several parameters including the speed of the vehicle and its mass. The details of this model is discussed in Section 3.

The classical vehicle routing problem (VRP) is one of the most widely studied problems in the literature. We refer the reader to survey papers (Erdelić & Carić, 2019; Kumar & Panneerselvam, 2012; Lin, Choy, Ho, Chung, & Lam, 2014) for applications and variations of classical VRP and solution approaches. The most related extension of the classical VRP (with time windows) to the problem studied in our paper is the Pollution-Routing Problem (PRP) (Bektaş & Laporte, 2011) which aims to control cost and emission. In PRP, speed is a decision variable affecting the fuel consumption, and the objective is the minimization of the drivers' wages and costs of fuel-emission. The authors formulate the PRP as an MILP problem using discretized speed values. After the PRP is introduced, several follow-up studies considered this problem and its extensions. For example, Demir, Bektaş, and Laporte (2012) and Kramer, Subramanian, Vidal, and Lucídio dos Anjos (2015) propose an adaptive large neighborhood search (ALNS) algorithm and a matheuristic approach for the PRP, respectively. Demir, Bektaş, and Laporte (2014) introduce the bi-objective PRP, where the minimization of drivers' wages and fuel-emission costs are considered as two conflicting objectives. Franceschetti, Honhon, Van Woensel, Bektaş, and Laporte (2013) propose time-dependent PRP as another extension of the PRP and develop a tabu search procedure to solve it.

Our Contribution. In this study, a minimum cost path problem is studied for a PEV which aims to minimize the energy consumption of a PEV unlike most of the studies in the literature. In this problem, given origin and destination nodes, locations of charging stations in the network, a limit on the total travel time, and speed limits on each arc of the network; the PEV aims to find a path between origin and destination nodes, decide at which charging stations to stop by and how much to recharge at each, and determine the vehicle speed on each arc on this path so as to minimize the total energy consumption while making sure that the destination is reached within the given total time limit. This is one of the few studies where the speed is a continuous decision variable. For the considered problem, an exact solution approach is proposed which is an MISOCP formulation that is able to solve medium-size instances to optimality. To be able to solve larger instances, a matheuristic approach and an ILS algorithm are developed. It is experimentally shown that these heuristics provide good quality solutions in reasonable computational times.

3. Problem description

In this section, we define the energy efficient path problem for a PEV (PEVEEP), which aims to find the most energy efficient path of a PEV joining predetermined origin and destination nodes in a network. In this network, there are electric recharging stations at some nodes and speed limits on each arc. PEV starts its travel at the origin and has to arrive at the destination within some given total time limit by following a path. In PEVEEP, we determine which path to follow, at what speed to drive on each arc, at which nodes of this path to stop for recharging, and how much to recharge at each stop so as to minimize the total energy consumption of the PEV while making sure that the destination is reached within the total time limit. The total time limit restricts the sum of total driving time spent on the arcs and the total time spent for recharging the vehicle at the recharging stations.

In order to estimate the energy consumption of a PEV traveling at some constant speed, we use the formulation provided by Li et al. (2017). According to this formulation, the instantaneous power loss, P, of an EV is calculated as

$$P = (AV^2 + f_r Mg + BV/R_t)V + rR_t^2/K^2(AV^2 + f_r Mg + BV/R_t)^2 + P_a, \quad (1)$$

where V is the speed of the vehicle, r is the resistance of the conductor, M is the mass of the vehicle, g is the gravitational acceleration, A is the aerodynamic constant, f_r is the rolling resistant constant, B is bearings' damping coefficient, K is armature constant, R_t is tire radius, and P_a is the ancillary loss including the loss sourcing from air condition, external lights, and audio.

The energy consumption, E, of an EV traveling at constant speed V for a distance of D units, is calculated by multiplying the right hand side of Eq. (1) by $t = \frac{D}{V}$, where t denotes the driving time. After simplifications, E is obtained as

$$E = D\,(a\,V^3 + b\,V^2 + c\,V + d + e/V), \quad (2)$$

where $a = A^2(r\frac{R_t^2}{K^2})$, $b = A + (\frac{2ABrR_t}{K^2})$, $c = \frac{B}{R_t} + \frac{(2MgAf_r + B^2)r}{K^2}$, $d = f_r Mg(1 + \frac{2BrR_t}{K^2})$, $e = (\frac{f_r^2(Mg)^2 rR_t^2}{K^2} + P_a)$. This function is convex on V and its global minimum is denoted as V_{opt}.

As an example, we consider a specific EV which has the parameter values given in Table 2 taken from Li et al. (2017). For this vehicle, the values of a, b, c, d, and e are given in Table 3. Fig. 1 shows how the energy consumption rate per unit travel distance (in Joule/m) of this EV changes with respect to speed (in m/s), according to Eq. (2). As it can be seen in Fig. 1, E is a convex function of V and its global minimum V_{opt} is equal to 13.04 m/s (46.93 km/h).

Table 2
Energy consumption function parameter values (Li et al., 2017).

Parameter	Value	Unit
M	1500	kg
f_r	0.015	–
r	0.110	Ω
A	1.800	m^2
g	9.810	m/s^2
R_t	0.300	m
P_a	2.000	kW

Table 3
The values of coefficients in energy consumption model.

Coefficient	Value	Unit
a	1.023×10^{-5}	kg s/m
b	0.324	kg
c	3.348	kg m/s
d	220.868	kg m^2/s^2
e	2004.747	kg m^3/s^3

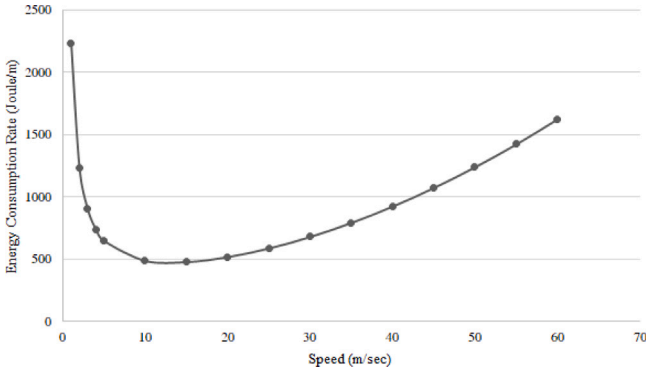

Fig. 1. Energy consumption of a PEV per unit travel distance as a function of speed.

4. Mathematical formulations for the PEVEEP

We now provide a mixed-integer nonlinear programming (MINLP) formulation for the PEVEEP. We use the notation, parameters, and the energy calculation descriptions given in Section 3. Moreover, $G = (\mathcal{N}, \mathcal{A})$ represents the directed network where $\mathcal{N} = \{1, \ldots, N\}$ denotes the set of nodes and $\mathcal{A}$ the set of arcs. We assume that 1 is the origin and N is the destination. We denote the subset of the nodes with recharging stations by $S, S \subseteq \mathcal{N}$. If the PEV stops at a recharging station and gets ec units of energy (Joule), then this takes $A + ec\,B$ seconds, where A is the fixed time spent in seconds to recharge and B is the time spent in seconds per unit energy recharged. The other parameters used to formulate the PEVEEP are given below:

U_{ij}: Upper speed limit on arc $(i, j) \in \mathcal{A}$ (m/s)
L_{ij}: Lower speed limit on arc $(i, j) \in \mathcal{A}$ (m/s)
C: Capacity of the battery (Joule)
m: Minimum charge level of the battery (Joule)
I: Initial charge level of the battery at the origin node (Joule)
D_{ij}: Length of arc $(i, j) \in \mathcal{A}$ (m)
T: Total time limit to travel from the origin node to the destination (s)

The variables used to formulate the PEVEEP are as follows.

E_{ij}: Electric energy consumption of the EV on arc $(i, j) \in \mathcal{A}$ (Joule)
V_{ij}: Speed of the EV on arc $(i, j) \in \mathcal{A}$ (m/s)
ec_i: Recharging amount of the EV's battery at node $i \in S$ (Joule)
ea_i: Charge level of the EV's battery when it arrives at node $i \in \mathcal{N}$ (Joule)
ed_i: Charge level of the EV's battery when it departs from node $i \in \mathcal{N}$ (Joule)
t_{ij}: Driving time of the EV on arc $(i, j) \in \mathcal{A}$ (s)

ct_i: Amount of time spent for recharging the EV's battery at node $i \in S$ (s)

$$x_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{A} \text{ is included in the path,} \\ 0, & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{if PEV stops at node } i \in S \text{ to recharge,} \\ 0, & \text{otherwise} \end{cases}$$

4.1. An MINLP formulation for the PEVEEP

Using the parameters, variables and the energy consumption functions described so far, we provide a mixed-integer nonlinear programming (MINLP) formulation for the PEVEEP below.

(PEVEEP-MINLP)

$$\text{Minimize} \sum_{(i,j)\in\mathcal{A}} E_{ij} \tag{3}$$

subject to

$$E_{ij} \geq D_{ij}\,(a\,V_{ij}^3 + b\,V_{ij}^2 + c\,V_{ij} + d\,x_{ij} + e/V_{ij}) \qquad \forall (i,j) \in \mathcal{A} \tag{4}$$

$$\sum_{i:(1,i)\in\mathcal{A}} x_{1i} = 1 \tag{5}$$

$$\sum_{i:(i,N)\in\mathcal{A}} x_{iN} = -1 \tag{6}$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij} - \sum_{i:(j,i)\in\mathcal{A}} x_{ji} = 0 \qquad \forall j \in \mathcal{N} \setminus \{1, N\} \tag{7}$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij} \leq 1 \qquad \forall j \in \mathcal{N} \tag{8}$$

$$M\,(1 - x_{ij}) \geq ea_j - ed_i + E_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{9}$$

$$M\,(x_{ij} - 1) \leq ea_j - ed_i + E_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{10}$$

$$V_{ij} \leq U_{ij}\,x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{11}$$

$$V_{ij} \geq L_{ij}\,x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{12}$$

$$x_{ij}\,D_{ij} = V_{ij}\,t_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{13}$$

$$t_{ij} \leq x_{ij}\,D_{ij}/L_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{14}$$

$$y_j \leq \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \qquad \forall j \in S \tag{15}$$

$$ed_i = ea_i + ec_i \qquad \forall i \in S \tag{16}$$

$$ed_i = ea_i \qquad \forall i \in \mathcal{N} \setminus S \tag{17}$$

$$ed_j \leq C \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \qquad \forall j \in \mathcal{N} \setminus \{1, N\} \tag{18}$$

$$ea_j \geq m \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \qquad \forall j \in \mathcal{N} \setminus \{1\} \tag{19}$$

$$ec_i \leq y_i\,(C - m) \qquad \forall i \in S \tag{20}$$

$$ea_1 = I \tag{21}$$

$$ec_N = 0 \tag{22}$$

$$ct_i = A\,y_i + B\,ec_i \qquad \forall i \in S \tag{23}$$

$$\sum_{(i,j)\in\mathcal{A}} t_{ij} + \sum_{i\in S} ct_i \leq T \tag{24}$$

$$V_{ij}, t_{ij}, E_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{25}$$

$$ed_i, ea_i, \geq 0 \qquad \forall i \in \mathcal{N} \tag{26}$$

$$ec_i, ct_i \geq 0 \qquad \forall i \in S \tag{27}$$

$$y_i \in \{0, 1\} \qquad \forall i \in S \tag{28}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in \mathcal{A} \tag{29}$$

The objective function of (PEVEEP-MINLP) minimizes the total amount of electric energy consumed by the PEV on its path from the origin to destination. Constraints (4) are used to calculate the amount of energy consumed by the EV on each arc. The objective function enforces that they hold at equality at an optimal solution. Constraints (5), (6), (7), and (8) are the classical shortest path constraints that are used to construct a path over the network while making sure that no subtours appear with the help of the objective function. Constraints (9) and (10) make sure that if arc (i,j) is on the constructed path, then the energy level upon the arrival at node j is equal to the sum of the energy level when departing from node i and the energy consumed on arc (i,j). The speed of the PEV on each arc satisfies the speed limits by constraints (11) and (12). By constraints (13), we establish the relation between distance, speed, and driving time on each arc if the arc is used. Constraints (11) and (14) make sure that V_{ij} and t_{ij} are both equal to zero if $x_{ij} = 0$ for every arc (i,j). Note that, it can be assumed that $L_{ij} > 0$ as in any optimal solution that uses arc (i,j) the speed of the EV will be at least $min\{U_{ij}, V_{opt}\}$. Hence, if L_{ij} is less than $min\{U_{ij}, V_{opt}\}$, then it can be updated to this value without affecting the optimal solution. Constraints (15) ensure that the EV can recharge its battery at node $j \in S$ if j is on the constructed path. Constraints (16) are used to make the energy level of the EV when departing from a node equal to the sum of the energy level upon its arrival to the node and the recharging amount at that node. Constraints (17) guarantee that the energy level when departing from a non-station node is equal to its energy level at arrival to that node. By constraints (18) and (19), the energy levels of the EV when arriving at a node and departing from it should be between the minimum and maximum allowed charge levels if the node is on the constructed path, respectively. If the EV does not stop at a station node, then the constraints (20) make sure that the recharging amount is equal to zero at that node. The predetermined initial charge level of the EV's battery at the origin node and the final charge level at the destination node are enforced by the constraints (21) and (22), respectively. Constraints (23) are used to compute the amount of time spent at a recharging station which consists of a fixed time and a variable time depending on the recharging amount. Constraints (24) make sure that the EV arrives at the destination before the total time limit T. Constraints (25), (26), and (27) are the non-negativity constraints and constraints (28) and (29) enforce binary restrictions on the other decision variables.

The PEVEEP aims to find a path with minimum energy consumption between the origin and destination nodes while satisfying time and energy constraints. The weight constrained shortest path problem (WCSPP) is an NP-hard problem (Arslan et al., 2015). In this problem, each arc in the network has an associated weight in addition to distance. The WCSPP aims to find a shortest path between the origin and destination nodes in a directed network such that the total weight of the path is less than some predetermined value. Now, assuming that the EV can travel at any speed, we show that the PEVEEP is NP-hard even when the speed values of the EV are fixed on each arc by demonstrating that it is a generalization of the WCSPP. To observe this, consider an instance of the WCSPP, where each arc has a positive length and weight. The energy consumption rate of an EV per unit time can be any value greater than or equal to some constant δ. We scale the distances of arcs in the WCSPP such that the distance over weight is at least δ for every arc. Now the weights in WCSPP become the driving times in PEVEEP and distances become energy consumption values, where the speed is fixed in each arc to make energy consumption rate per unit time equal to the distance of the arc divided by its weight. Furthermore, we assume that the initial charge amount of the EV's battery is large enough so that there is no need to stop at a station to recharge. Thus, we have converted an instance of the WCSPP to an instance of PEVEEP, proving that the PEVEEP is NP-hard as well even when the speed values are fixed in every arc.

The formulation (PEVEEP-MINLP) in its current form has nonlinear constraints which are (4) and (13). In the next section, we formulate the PEVEEP as a mixed-integer second order cone programming (MISOCP) problem by rewriting the nonlinear constraints as second order cone constraints.

4.2. An MISOCP formulation for the PEVEEP

We now provide an MISOCP formulation of the PEVEEP. A second order cone programming (SOCP) problem is a convex optimization problem where a linear objective function is minimized over second order cone constraints which are of the from $\|Ax + b\| \leq r^T x + d$. Note that, linear constraints are second order cone constraints. Linear programs, convex quadratic programs, and quadratically constrained convex quadratic programs are some of the problems which can be recast as SOCP problems (Alizadeh & Goldfarb, 2003). An MISOCP problem is an SOCP problem where a subset of the variables is restricted to take on integer values.

Recall the nonlinear constraints (4) and (13) in (PEVEEP-MINLP). Constraints (4) are nonlinear because of the terms V_{ij}^3, V_{ij}^2, and V_{ij}^{-1}; and the constraints (13) are so because of the term $V_{ij}t_{ij}$. We can rewrite these constraints as second order cone constraints as follows. First, we introduce new variables, $f_{ij} \geq 0$ and $l_{ij} \geq 0$, to substitute the terms V_{ij}^3 and V_{ij}^2, in (4), respectively. Second, we substitute the term V_{ij}^{-1} in (4) by D_{ij}/t_{ij}. After these substitutions, (4) becomes $E_{ij} \geq D_{ij}\ (a\ f_{ij} + b\ l_{ij} + c\ V_{ij} + d\ x_{ij}) + e\ t_{ij}$ and we have the following new constraints to be added to the formulation: $l_{ij} \geq V_{ij}^2$ and $f_{ij}V_{ij} \geq l_{ij}^2$. The newly added constraints can be rewritten in the form of second order cone constraints as given in (30)–(32).

$$\left\| \begin{pmatrix} V_{ij} \\ (l_{ij} - 1)/2 \end{pmatrix} \right\| \leq (l_{ij} + 1)/2 \qquad \forall (i,j) \in \mathcal{A} \tag{30}$$

$$\left\| \begin{pmatrix} l_{ij} \\ (f_{ij} - V_{ij})/2 \end{pmatrix} \right\| \leq (f_{ij} + V_{ij})/2 \qquad \forall (i,j) \in \mathcal{A} \tag{31}$$

$$l_{ij}, f_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{32}$$

Using the fact that $x_{ij} = x_{ij}^2$, the constraints (13) can be rewritten as $x_{ij}^2 D_{ij} \leq V_{ij}t_{ij}, \forall (i,j) \in \mathcal{A}$. To see this, assume that in an optimal solution $x_{ij}^2 D_{ij} < V_{ij}t_{ij}$ holds true. In this case, one can obtain a solution with a better objective function value by decreasing t_{ij}. The constraints $x_{ij}^2 D_{ij} \leq V_{ij}t_{ij}, \forall (i,j) \in \mathcal{A}$ can now be rewritten as the following second order cone constraints.

$$\left\| \begin{pmatrix} x_{ij}\sqrt{D_{ij}} \\ (t_{ij} - V_{ij})/2 \end{pmatrix} \right\| \leq (t_{ij} + V_{ij})/2 \qquad \forall (i,j) \in \mathcal{A} \tag{33}$$

After all these modifications, an MISOCP formulation of the PEVEEP is given in (PEVEEP-MISOCP).

(PEVEEP-MISOCP)

$$\text{Minimize} \sum_{(i,j)\in\mathcal{A}} E_{ij} \tag{3}$$

subject to

$$E_{ij} \geq D_{ij}\ (a\ f_{ij} + b\ l_{ij} + c\ V_{ij} + x_{ij}\ d) + e\ t_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{34}$$

(5)–(12), (14)–(33)

The objective function in (PEVEEP-MISOCP) is linear and all its constraints are second order cone constraints (including the linear constraints). Therefore, this formulation is an MISOCP formulation. There are several off-the-shelf solvers that can solve MISOCP problems. Our computational experiments show that we can solve medium-size instances of the PEVEEP to optimality in reasonable times using the formulation (PEVEEP-MISOCP) and the CPLEX solver (IBM, 2021). For larger instances this formulation becomes inefficient. For this reason, we propose some heuristic solution algorithms in the next section to be able to solve large-size instances.

5. Heuristic approaches

In this section, we present two heuristic solution approaches for the PEVEEP. The first solution approach, which is a matheuristic, starts with an initial path and aims to improve it by trying alternative paths which are evaluated using the (PEVEEP-MISOCP) with a fixed subset of the variables. The second solution approach is an iterated local search (ILS) algorithm in which an initial path is constructed and three different neighborhood definitions are used to obtain better solutions (paths).

5.1. A matheuristic algorithm

We propose a matheuristic for the PEVEEP which uses different mathematical programming formulations and neighborhood search methods. These formulations are based on the mathematical programming formulations of the PEVEEP with different groups of fixed variables and different objective functions. (PEVEEP-MINLP) becomes an MILP when the speed decision variables are fixed in advance and some constraints are accordingly updated. Moreover, based on our computational experiments, the (PEVEEP-MISOCP) formulation finds the optimal solution much faster when the path is determined in advance, i.e., when the binary variables determining the path are fixed. With these observations, a matheuristic approach is proposed to solve the PEVEEP. Before the matheuristic is explained in more detail, we need to make two definitions. From now on, when we talk about a solution, we mean that we are given a path joining origin and destination nodes and all the values of the decision variables of the formulation (PEVEEP-MINLP).

Definition 1. A solution is said to be energy feasible if it satisfies all the constraints of (PEVEEP-MINLP) except possibly the total time limit constraint (inequality (24)). Otherwise, the solution is said to be energy infeasible.

Definition 2. An energy feasible solution is said to be time feasible if it satisfies the total time limit constraint (inequality (24)) as well. Otherwise, the solution is said to be time infeasible.

Next, we discuss two different initialization methods that we use within the matheuristic to find an initial path. Then, we explain how this initial path is improved by the improvement algorithms. In both initialization methods and improvement algorithms, some mathematical programming formulations are utilized.

5.1.1. Initial path construction

In the first initialization method, we fix all the speed values in advance, and aim to find an energy feasible solution by minimizing total travel time of the PEV. The speed of the PEV on arc (i,j) is fixed to the speed value which results in the lowest energy consumption, i.e., $V_{ij} = min\{max\{V_{opt}, L_{ij}\}, U_{ij}\}$. The resulting problem which aims to find a **T**ime **E**fficient **P**ath is an MILP problem which is given in the formulation (TEP-MILP). If (TEP-MILP) is infeasible, then the PEVEEP is infeasible as well. Otherwise, it returns an energy feasible solution which is taken as an initial solution by our matheuristic. Note that this initial solution may or may not be time feasible.

(TEP-MILP)

$$\text{Minimize} \sum_{(i,j)\in\mathcal{A}} t_{ij} + \sum_{i\in S} ct_i \tag{35}$$

subject to

$$E_{ij} \geq D_{ij}\ x_{ij}\ (a\ V_{ij}^3 + b\ V_{ij}^2 + c\ V_{ij} + d + e/V_{ij}) \qquad \forall (i,j) \in \mathcal{A} \tag{36}$$

(5)–(10), (13)–(29)

In the second initialization method, we enumerate the first k shortest paths joining origin and destination nodes, and select one of them uniformly at random. For the selected path, total travel time of the PEV is minimized by solving the formulation (TEFP-MISOCP) which aims to find a **T**ime **E**fficient solution on the selected **F**ixed **P**ath. This formulation decides on the speed of the PEV on each arc on the path,

the station(s) at which to stop by, and how much to recharge at each stop. If (TEFP-MISOCP) is feasible, then the selected path is energy feasible and is taken as the initial path of the matheuristic. On the other hand, if (TEFP-MISOCP) is infeasible, then the selected path is energy infeasible. In this case, another path from the first k shortest paths is selected uniformly at random, and the same steps are repeated until an energy feasible path is found.

(TEFP-MISOCP)

$$\text{Minimize} \sum_{(i,j)\in\mathcal{A}} t_{ij} + \sum_{i\in S} ct_i \tag{35}$$

subject to

(9)–(12), (14)–(23), (25)–(28), (30)–(34)

5.1.2. Improvement algorithms

After an energy feasible initial solution is constructed by one of the initialization methods, the matheuristic aims to improve it. If the initial solution is time infeasible, the matheuristic calls the *T-Improvement* procedure which aims to obtain a time feasible solution. On the other hand, if the initial solution is time feasible or if *T-Improvement* procedure finds a time feasible solution, then the matheuristic calls the *E-Improvement* procedure to obtain an energy improved path.

The pseudocode of the proposed matheuristic is given in Algorithm 1. In Step 3 of Algorithm 1, one of the initialization methods is called. If MIP Initialization method is called, and (TEP–MILP) turns out to be infeasible, then the PEVEEP is infeasible and the matheuristic terminates (see Steps 4 and 5). On the other hand, if the Short Path Initialization method is called, and none of the k shortest paths is energy feasible, then the matheuristic terminates, yet the PEVEEP may or may not be infeasible (see Steps 4 and 5). Once an energy feasible initial solution is constructed, the matheuristic checks whether it is time feasible or not at Step 7. If the initial solution is time infeasible, the *T-Improvement* procedure is called (see Step 8). If the *T-Improvement* procedure returns a time feasible solution, then the *E-Improvement* procedure is called to further decrease the energy consumption of the PEV if possible (see Step 12). On the other hand, if the *T-Improvement* procedure cannot find a time feasible solution, then the matheuristic terminates with no feasible solution at hand (see Steps 9 and 10). If the initial solution obtained at Step 3 is time feasible, then the matheuristic directly calls the *E-Improvement* procedure at Step 15. Note that the *E-Improvement* procedure is called at most once by the matheuristic whose output is returned as the output of the matheuristic (see Step 18).

We now provide the details of two improvement procedures used within the proposed matheuristic; namely, the *T-Improvement* and *E-Improvement* procedures.

A pseudocode of the *T-Improvement* procedure is provided in Algorithm 2. The inputs of the procedure are a time infeasible path, all parameters of the PEVEEP, and two positive integers k and K. The procedure outputs a time feasible path if it can find one. At Step 3 of the procedure, the current path is taken as the initial time infeasible path. Then (TEFP-MISOCP) formulation is solved for the current path at Step 4. If the optimal solution is time feasible, then the procedure returns this solution and terminates (see Steps 5 and 6). Otherwise, the procedure selects two nodes i and j uniformly at random from the nodes that are on the current path, and enumerates k shortest paths between i and j using the Yen's algorithm (Yen, 1971), at Steps 10 and 11, respectively. The current path is modified using the enumerated shortest paths one by one starting with the shortest one and if any of the modified paths is energy feasible and has a shorter total travel time, then the current path is updated with the modified one (see Steps 13–16). In this case, if the current path becomes time feasible, then it is returned by the procedure and the procedure terminates (see Steps 17 and 22). Otherwise, the procedure selects another pair of nodes on the current path and continues looking for modified paths with less total travel time. In the *T-Improvement* procedure, at most K pairs of nodes are selected (see Step 8) and for each selected pair, k shortest paths are enumerated (see Step 11).

Algorithm 1 Matheuristic for PEV

1: **Input:** All parameters in the (PEVEEP-MINLP) formulation, V_{opt}
2: **Output:** An energy efficient and time feasible path, the values of all decision variables of (PEVEEP-MINLP) including speed of the PEV on each arc, stations the PEV stops at, recharge amounts, total energy consumption, and total travel time.
3: Call one of the initialization methods (MIP or Short Path Initialization).
4: **if** the initialization method cannot find an energy feasible solution **then**
5: Terminate the Matheuristic.
6: **else**
7: **if** the initial solution (path) is time infeasible **then**
8: Call $T-Improvement$ procedure for the initial path.
9: **if** $T-Improvement$ procedure returns empty **then**
10: Terminate the Matheuristic.
11: **else**
12: Call $E-Improvement$ procedure for the solution (path) returned by $T-Improvement$ procedure.
13: **end if**
14: **else**
15: Call $E-Improvement$ procedure for the initial path.
16: **end if**
17: **end if**
18: **Return** output of $E-Improvement$ procedure.

The *E-Improvement* procedure, summarized in Algorithm 3, gets a time feasible path, all parameters of the PEVEEP, and two positive integers k and K, as the inputs. The procedure aims to reduce the energy consumption of the PEV by evaluating neighboring paths and hopefully outputs an energy efficient time feasible path. At Step 3, the inputted path is taken as the current path. The (EEFP-MISOCP) formulation is solved for the current path at Step 4 with the aim of obtaining the lowest possible energy consumption of the PEV over the current path while making sure that the total time limit constraint is not violated. Then, the procedure selects a pair of nodes i and j uniformly at random from the nodes that are on the current path, and enumerates k shortest paths between i and j, at Steps 7 and 8, respectively. The current path is modified using the enumerated shortest paths one by one starting with the shortest one and if any of the modified paths is energy feasible and has a smaller total energy consumption value for the PEV, then the current path is updated with the modified one (see Steps 10–13). In this case, the remaining modified paths are not evaluated and the procedure continues by selecting a pair of nodes from the newly updated current path. On the other hand, if none of the modified paths decreases the energy consumption of the PEV, then the procedure selects a new pair of nodes on the current path to look for another set of neighboring paths with less energy consumption for the PEV. Overall, in the *E-Improvement* procedure, K pairs of nodes are selected (see Step 5) and for each selected pair, k shortest paths are enumerated (see Step 8). Note that the procedure is a descent method as the current solution is updated whenever an improvement is observed in terms of the energy consumption of the PEV and finally the best solution is returned at Step 20.

(EEFP-MISOCP)

$$\text{Minimize} \sum_{(i,j)\in\mathcal{A}} E_{ij} \tag{3}$$

subject to

(9)–(34)

Table 4
Comparison of (PEVEEP-MISOCP) and the mathematical programming formulations used within the matheuristic.

Formulation	Objective function	Fixed path or network	Time limit applied?	Speed on each arc
PEVEEP-MISOCP	Energy minimization	Network	Yes	DV
TEP-MILP	Total time minimization	Network	No	Parameter
TEFP-MISOCP	Total time minimization	Fixed path	No	DV
EEFP-MISOCP	Energy minimization	Fixed path	Yes	DV

Algorithm 2 T-Improvement Procedure

Input: A time infeasible path, all parameters of the PEVEEP, k, and K.
Output: A time feasible path, the values of all decision variables of (PEVEEP-MINLP) including speed of the PEV on each arc, stations the PEV stops at, recharge amounts, total energy consumption, and total travel time.
Let the current path be the initial path at hand, $a \leftarrow 0$.
Solve the (TEFP-MISOCP) formulation for the current path and let T_{BEST} be the optimal objective function value.
if $T_{BEST} \leq T$ **then**
  Return current path and related output.
else
  while $a < K$ **do**
    $b \leftarrow 0$.
    Choose two nodes i and j uniformly at random from the nodes that are on the current path.
    List k shortest paths from i to j and sort them from the shortest one to the longest.
    while $b < k$ **do**
      Replace the path between i and j in the current path with the b^{th} shortest path and call the resulting path as P'.
      Solve (TEFP-MISOCP) formulation for P' and denote its optimal objective function value by T^b.
      if $T^b < T_{BEST}$ **then**
        Current path $\leftarrow P'$ and $T_{BEST} \leftarrow T^b$.
        break
      end if
      $b = b + 1$.
    end while
    if $T_{BEST} \leq T$ **then**
      Return current path and related output.
    end if
    $a = a + 1$.
  end while
  Return $\emptyset$
end if

Algorithm 3 E-Improvement Procedure

Input: A time feasible path, all parameters of the PEVEEP, k, and K.
Output: An energy efficient time feasible path, the values of all decision variables of (PEVEEP-MINLP) including speed of the PEV on each arc, stations the PEV stops at, recharge amounts, total energy consumption, and total travel time.
Let the current path be the inputted path, $a \leftarrow 0$.
Solve the (EEFP-MISOCP) formulation for the current path and let E_{BEST} be the optimal objective function value.
while $a < K$ **do**
  $b \leftarrow 0$.
  Choose two nodes i and j uniformly at random from the nodes that are on the current path.
  List k shortest paths from i to j and sort them from the shortest one to the longest.
  while $b < k$ **do**
    Replace the path between i and j in the current path with the b^{th} shortest path and call the resulting path as P'.
    Solve (EEFP-MISOCP) formulation for P' and denote its optimal objective function value by E^b.
    if $E^b < E_{BEST}$ **then**
      Current path $\leftarrow P'$ and $E_{BEST} \leftarrow E^b$.
      break
    end if
    $b = b + 1$.
  end while
  $a = a + 1$.
end while
Return current path and related output.

Now, we would like to summarize the mathematical programming formulations used within the matheuristic. Table 4 compares MISOCP formulation of the PEVEEP; namely, (PEVEEP-MISOCP), and the mathematical programming formulations used within the matheuristic in terms of their objective functions, whether each formulation is run on a fixed path or over the whole network, whether total time limit constraint is imposed or not, and whether the speed values for the PEV on each arc are decision variables (DVs) or parameters.

5.2. ILS algorithm

As an alternative to the proposed matheuristic, we consider an iterated local search (ILS) algorithm for the PEVEEP. ILS algorithm has been successfully used in different routing problems in the literature, (see e.g., Hashimoto, Yagiura, & Ibaraki, 2008; Lourenço, Martin, & Stützle, 2003). The algorithm is an iterative procedure which starts with different initial solutions and aims to improve them by neighborhood search procedures. When the stopping condition is met, the algorithm returns the best solution found.

We now describe the initialization method and the three neighborhood search procedures that we use in the proposed ILS algorithm.

5.2.1. Initial path construction

The initialization methods used within the proposed matheuristic construct an initial path by the help of some mathematical programming formulations. In our ILS algorithm, we construct the initial path differently and get rid of the use of such formulations. The initialization method of the ILS algorithm starts with the origin node and moves to an adjacent node in such a way that the distance to the destination is now shorter. If among the adjacent nodes of the origin there are more than one such node, one of them is selected uniformly at random. In the following steps, we continue connecting the lastly added node to an adjacent node which brings us closer to the destination. If there are more than one such candidate node, we again choose one of them uniformly at random. At some point, this procedure will end up with a path which joins the origin and destination nodes. This path will be the initial path of the ILS algorithm which may be neither energy feasible nor time feasible.

5.2.2. Neighborhood search procedures

The PEVEEP has three main concerns: energy feasibility, time feasibility, and energy optimization. In our ILS algorithm, we consider three neighborhood search procedures in order to handle these concerns. Before explaining the neighborhood search procedures, we make a

definition. Given a solution (i.e., a path joining origin and destination nodes, speed values of the PEV on each arc of the path, the charging stations the PEV stops by, and the recharging amounts), the energy slack of a node on this solution is the amount of energy the PEV needs to bring its energy level to the minimum charge level of the battery (m) at the arrival to the node. To clarify this, let us assume that for a given solution and a node on the path of this solution, the PEV either does not have enough energy to reach this node, or it reaches the node with an energy level which is less than the minimum charge level of the battery. In this case, the energy slack of this node would be the minimum amount of additional energy the PEV needs in order to arrive at the node with an energy level equal to the minimum charge level of the battery. On the other hand, if the PEV is able to reach the node with an energy level that is at least the minimum charge level of the battery, then the energy slack of this node would be equal to zero. For a given solution, we define the total energy slack (TES) of it as the sum of the energy slacks of all the nodes on the path of this solution. If $TES = 0$, then the solution is energy feasible. Otherwise (if $TES < 0$), the solution is energy infeasible. We use TES as one of the criteria to evaluate a newly generated neighborhood solution.

We now explain how the proposed ILS algorithm improves the initial solution using three neighborhood search procedures. Once a path is obtained by the initialization method, the initial solution is constructed considering that the speed value on each arc (i, j) is equal to $V_{ij} = min\{max\{V_{opt}, L_{ij}\}, U_{ij}\}$. Moreover, the PEV can recharge the battery whenever a station is visited on the path in case there is a requirement. To determine the recharging amounts, an energy consumption routine is defined which first calculates the energy consumption on each arc assuming that the speed of the PEV is equal to $V_{ij} = min\{max\{V_{opt}, L_{ij}\}, U_{ij}\}$. Let us assume that the station nodes are visited in the order $s_1, s_2, \ldots, s_\ell$ on this path from origin to destination. Starting with the destination, we go backward and calculate the sum, es_ℓ, of all energy consumptions on the arcs visited between s_ℓ and the destination. If es_ℓ exceeds $(C - m)$, the PEV stops at s_ℓ and charges its battery up to $min\{C, es_\ell + m\}$. Otherwise, the PEV may or may not stop at s_ℓ. For this case, we evaluate the sum, $es_{\ell-1} + es_\ell$, of all energy consumptions of the arcs visited between the destination and $s_{\ell-1}$. If this sum exceeds $(C - m)$, then the PEV stops at s_ℓ and charges its battery up to $min\{C, es_\ell + m\}$ at this node. Otherwise, the PEV does not stop at s_ℓ and the backward procedure continues until the origin is reached. The energy consumption routine just explained helps us construct the initial solution which may be energy infeasible or time infeasible. Hence, the developed neighborhood search procedures are defined not only to improve the solution in terms of the energy consumption of the PEV but also to obtain an energy and a time feasible solution.

Once an initial solution is constructed, first, **S**tation **I**nsertion (SI) procedure is applied. Then, the **R**eplacement of **E**xpensive **N**odes (REN) and **S**hort **C**ut (SC) procedures are applied successively on the solution returned by the **immediately** previous procedure. All these procedures makes the energy calculations using the energy consumption routine defined previously. Before giving the details of the Algorithm, we first define the neighborhood search procedures.

Station Insertion: Once an initial solution is generated, the station insertion (SI) procedure checks its energy feasibility. If the solution is energy feasible (i.e., if $TES = 0$), then the SI procedure terminates. Otherwise, the procedure generates neighborhood solutions one after another by selecting two nodes on the given path and one recharging station, say s, which is not on the path, uniformly at random. The neighborhood solution is generated by connecting the selected two nodes via a shortest path including node s. The energy feasibility of the new solution is checked. If TES of the new solution improves, it is taken as the best solution. If $TES > 0$, iterations continue in a similar way. On the other hand, if $TES = 0$ or the number of iterations reaches a predetermined value, K, the procedure terminates. An illustration of one iteration of the SI procedure is provided in Fig. 2. In this example, a path starts at node 1 and ends at node 5 (see Fig. 2(a)). The SI procedure selects nodes 2 and 4 on the path (see Fig. 2(b)), and s from the set S. The initial path is then modified by rerouting the vehicle from node 2 to node 4 through a shortest path including node s (see Fig. 2(c)).

Replacement of Expensive Nodes: The replacement of expensive nodes (REN) procedure is applied as the second neighborhood generating structure on the solution returned by the SI procedure. The REN procedure tries to find a neighborhood solution that has a lower amount of energy consumption by exchanging some parts of the path and does not have a worse TES value. If the energy consumption value of the neighborhood solution is not lower or its TES value is larger, then the neighborhood solution is not accepted and the procedure looks for another neighborhood solution. Otherwise, the neighborhood solution is accepted as the best solution if it is either time feasible or the corresponding total travel time is lower. The procedure continues until the number of iterations reaches a predetermined value, K'. The neighborhood solutions are constructed by selecting two nodes on the given path uniformly at random, removing the partial path between them, and reconnecting the selected nodes via the shortest path between them. An illustration of the REN procedure is shown in Fig. 3. Fig. 3(a) shows a complete path starting at node 1 and ending at node 5. Nodes 2 and 4 are selected (see Fig. 3(b)), and they are reconnected by the shortest path between them (see Fig. 3(c)).

Short Cut: The solution returned by the REN procedure is finally given to the shortest cut (SC) procedure to generate new neighborhood solutions. The SC procedure generates a neighborhood solution by connecting two non-adjacent nodes (selected uniformly at random) on the path of the given solution by a direct arc between them if such an arc exists. The new solution is accepted as the best solution if the same criteria in the REN procedure are met. That is, if the new solution is not better than the current one in terms of the total energy consumption or has a worse TES value (which corresponds to degree of energy infeasibility), the procedure looks for other neighborhood solutions. Otherwise, if the new solution is time feasible or its corresponding total travel time is better than that of the current solution, then it is accepted as the best solution. The procedure continues until the number of iterations reaches a predetermined value, K'', or there is no direct arc between any two non-adjacent nodes on the path. Fig. 4 depicts how an iteration of the SC procedure is performed on an example path. In Fig. 4(a), a path starting and ending at nodes 1 and 5, respectively, is shown. On this path, two non-adjacent nodes, 2 and 4, are selected (see Fig. 4(b)), and the path is modified by directly connecting them (see Fig. 4(c)).

The pseudocode of the proposed ILS algorithm is given in Algorithm 4. Step 3 represents a loop corresponding to different starting points. In Step 4 of Algorithm 4, the initial path construction is called to construct the a^{th} initial path. Once an initial path is constructed, its total energy consumption value is calculated via the energy consumption procedure and accepted as the best energy consumption value on hand. Then, SI, REN, and SC procedures are called sequentially (see Steps 5, 6, and 7). All these procedures take the output of the immediately previous procedure as their input. After all the procedures are applied, (EEFP-MISOCP) formulation is called to calculate the decision variables and the objective function value in Step 8. If the recently calculated objective function value is better than the E_{BEST}, then Current Path and E_{BEST} are updated (see Steps 9 and 10). Note that all the procedures are called M times (see Step 11). At the end, current path and its related output are returned.

At the beginning of the ILS algorithm, the solution returned by the SI procedure can be energy feasible or not. If the given solution for the REN procedure is energy feasible it is guaranteed that the returned solution by the REN procedure which is given to the SC procedure is energy feasible as well. Similarly, if the given solution for the SC procedure is energy feasible, the returned solution will be so as well. Moreover, energy consumption improvement and time feasibility are two criteria that are considered in the REN and SC procedures. In other

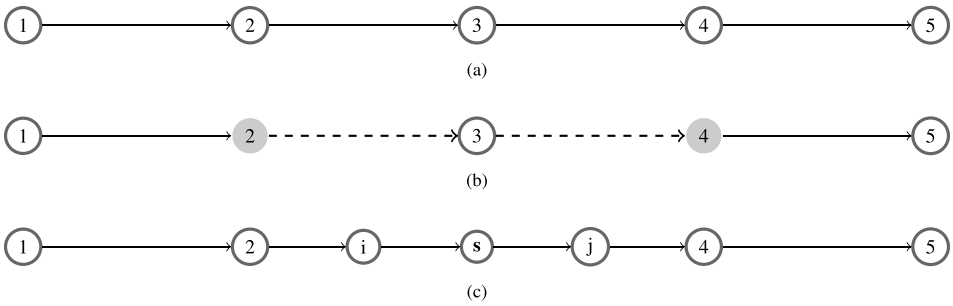

Fig. 2. An illustration of the SI procedure, (a) An energy infeasible path from node 1 to node 5, (b) Selection of two nodes, 2 and 4, and station s (not shown in figure), (c) Neighborhood solution obtained by connecting nodes 2 and 4 via the shortest path including s.

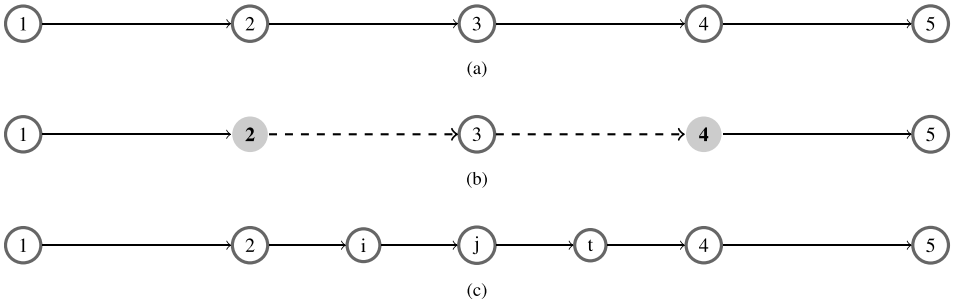

Fig. 3. An illustration of the REN procedure on a path from node 1 to node 5 (a), Selection of nodes 2 and 4 (b), Modifying the path between 2 and 4 (c).

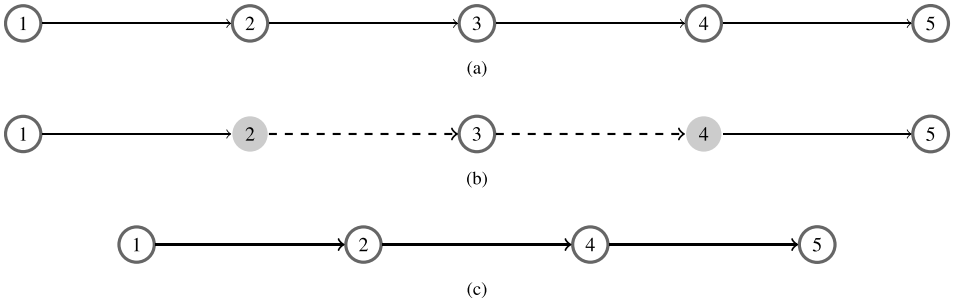

Fig. 4. An illustration of the SC procedure on a path from node 1 to node 5 (a), Selection of two nodes, 2 and 4 (b), Replacing the path between 2 and 4 by a direct arc (c).

Algorithm 4 ILS Algorithm

Input: All parameters in the (PEVEEP-MINLP) formulation, V_{opt}.
Output: A path, the values of all decision variables of (PEVEEP-MINLP) including speed of the PEV on each arc, stations the PEV stops at, recharge amounts, total energy consumption, and total travel time.
while $a \leq M$ **do**
    Call the initial path construction method and let E_{BEST} be the total energy consumption value and Current path is the resulted path.
    Call SI for the initial path.
    Call REN for given path by SI procedure.
    Call SC for given path by REN procedure and denote the resulted path by P'.
    Call $(EEFP-MISOCP)$ for P' and denote its optimal objective function value by E^a.
    if $E^a \leq E_{BEST}$ **then**
        Current path $\leftarrow P'$ and $E_{BEST} \leftarrow E^a$.
    end if
    $a = a + 1$.
end while
Return the current path and related output.

words, the returned solutions by these two procedures cannot be worse than their given solutions in terms of the total energy consumption and TES. Once an initial solution is processed by these three neighborhood solution generating procedures, the (EEFP-MISOCP) formulation (see Section 5.1.2) is called for the path corresponding to the resulting best solution. If the solution is feasible, then the ILS algorithm returns this solution as the best one. Otherwise, the algorithm returns empty. In total time, the ILS algorithm is called for M number of initial solutions, i.e., replications. The best solutions of all replications are evaluated, and the overall best solution is returned at the end.

6. Computational experiments

In this section, we discuss our observations on solving numerous instances of the PEVEEP using the PEVEEP-MISOCP formulation and the proposed heuristic solution approaches, i.e., the proposed matheuristic and ILS algorithm. The PEVEEP-MISOCP formulation as well as the involved formulations in the matheuristic are solved by CPLEX 12.10.0, while C++ language is used to code the algorithms. All experiments are performed in an environment with an Intel(R) Core(TM) i7-8550U CPU @ 1.80 GHz, 8 GB RAM and Windows 10.

In the rest of this section, all experimental settings including instance generation methods and parameter settings in the algorithms are

described in Section 6.1. The PEVEEP-MISOCP formulation, matheuristic, and ILS algorithm are compared in terms of solution quality and time in Section 6.2.

6.1. Experimental settings

To the best of our knowledge, there is no study in the literature that considers speed optimization and energy efficiency for PEVs simultaneously. Therefore, there is no available benchmark instances suitable for the PEVEEP in the literature. Hence, we use available network instances defined for the mixed capacitated arc routing problem that are publicly accessible (see Belenguer, Benavent, Lacomme, & Prins, 2006; Gouveia, Mourão, & Pinto, 2010). All the networks are incomplete, and have directed arcs with predetermined costs on each. We use arc costs in the provided instances as the distances of the corresponding arcs in our problem. In total, instances with 6 different node sizes, 24, 41, 50, 146, 195, and 321, are used.

The vehicle-dependent parameter values are the same as those provided in Section 3 for all instances. Moreover, the values of the minimum battery capacity, maximum battery capacity, and the initial charge level of the vehicle, are taken as $m = 1.5 \times 10^8$, $C = 2m$, and $I = 1.5m$ J, respectively. For all networks, lower and upper speed limits on each arc are generated uniformly at random where the lower speed limit, L, is selected uniformly at random between 40 and 70 km/h, and the upper speed limit, U_{ij}, is selected uniformly at random between 80 and 120 km/h. Remember that with the parameters used, the speed value, V_{opt}, which minimizes the energy consumption of the PEV, is equal to 46.93 km/h (see Fig. 1 in Section 3). If for an arc (i, j), L is less than V_{opt}, then the best speed value minimizing the energy consumption of the PEV is equal to $\min\{V_{opt}, U_{ij}\}$ assuming that there are no total time limits. Otherwise, the best speed value is L if the total time limit, T, allows. When there is a total time limit restriction, the best speed value of the PEV may be different from the stated values.

In our instances, there are recharging stations on some of the nodes, namely the station nodes. In order to analyze the effect of the number of station nodes on solutions, we define different station densities (SDs), which are 20%, 30%, and 40%, for each node size. SD represents the probability that a node on the network will be a recharging station. In this regard, the expected total number of station nodes on a network is equal to SD $\times$ $|N|$. To be able to make a better analysis for the effect of SDs on the solutions, 3 sub-instances are generated from each instance using each SD value. Total time limit, T, is taken as loose in the initial experiments (for the networks whose node sizes are lower than 195, T is selected as 10^6 s, while for the remaining networks T is taken as 5×10^6 s). In a following experiment the effects of making the total time limit tighter are investigated on the energy consumption and the average speed values of the PEV and the optimal path the PEV should follow.

The proposed heuristic solution approaches, i.e., the matheuristic and the ILS algorithm, have a few parameters to be set. For the matheuristic, two parameters, k and K, are used in both of the improvement algorithms ($E - Improvement$ and $T - Improvement$). We selected $k = K = 5$ for both improvement algorithms based on some preliminary experiments. In the ILS algorithm, the values of the parameters, M, K' and K'', are taken as 5, 15, and 5, respectively after some preliminary experiments.

6.2. Computational results

In this section, we analyze the performances of all solution approaches we propose for the PEVEEP. For each instance and each SD, three sub-instances are used. First, the PEVEEP is solved using the (PEVEEP - MISOCP) formulation under loose total time limits. Table 5 provides the average values for the optimal results over three sub-instances for each instance setting, i.e., specific size and SD. The values in the table are obtained by solving $5 \times 3 \times 3 = 45$ sub-instances.

Table 5
Optimal solutions using (PEVEEP - MISOCP) formulation under loose total time limits.

|N|	SD	Average energy consumption value (J $\times$ 10^6)	Average solution time (s)
24	20	58.2	1.5
	30	60.2	1.8
	40	58.2	1.5
41	20	194.0	48.8
	30	194.0	9.3
	40	204.0	87.2
50	20	95.3	17.4
	30	94.3	26.9
	40	91.4	23.6
146	20	117.1	3938.2
	30	116.1	4582.2
	40	114.2	4597.2
195	20	252.0	22 312.4
	30	251.6	20 482.8
	40	252.2	37 098.1

Each row corresponds to an instance setting with node size in the first column (titled as "|N|") and SD in the second column. The third and fourth columns provide the average objective function values and average solution times.

Table 5 shows that the average solution times are less than 2 s for the instances whose node sizes are 24. For the instances whose node sizes are less than or equal to 50, the optimal solution can be reached within 100 s on average. In general, as the instance sizes get larger, the solution times get larger as well (one exception to this is when we go from instances with 41 nodes to 50 nodes). While we can solve instances with 146 nodes within 1.5 h, instances with 195 nodes take much longer to solve. We also tried three sub-instances with 321 nodes, and observed that CPLEX is not able to solve any of them in three days.

Second, the PEVEEP is solved using the matheuristic and ILS algorithm using loose total time limits. The matheuristic is solved with two different initialization methods. Average optimality gaps and solution times over three sub-instances are provided in Table 6. For all but sub-instances with 321 nodes, the optimality gaps are calculated as the difference between the objective function values of the heuristic solutions and exact solutions divided by the objective function values of the exact solutions multiplied by 100. For the sub-instances with 321 nodes, the optimality gaps are calculated with respect to the best known objective function values. In Table 6, rows correspond to instance settings and three groups of columns correspond to matheuristic with first initialization method, matheuristic with second initialization method, and ILS algorithm. The resulting average solution times over 3 sub-instances and optimality gaps are provided in the related columns. Bold numbers in the table reflect the best solution approach in terms of optimality gap where ties are broken by looking at the average solution times.

For the node size 24, all solution approaches solve the sub-instances to optimality. When the node size is 41, all sub-instances are solved to optimality by all solution approaches except the ones with SD = 40% that are solved by ILS algorithm. Instances with 50 nodes are solved to optimality except the ones with SD = 20% that are solved by the matheuristic with second initialization method. For the node sizes less than or equal to 50, the performances of all solution approaches are almost perfect. All sub-instances are solved within 27 s by all solution approaches, and almost all of the optimality gaps are 0. Matheuristic with first initialization method stands out from other approaches by finding the optimal solutions for all these sub-instances. For the instances with 146 nodes, the ILS algorithm finds better quality solutions, while matheuristic with second initialization method finds solutions faster. For the instances with 195 and 321 nodes, the best solution approach is matheuristic with second initialization method

Table 6
Comparison of the performances of heuristic solution approaches under loose total time limits.

|N|	SD	Matheuristic with first initialization method		Matheuristic with second initialization method		ILS algorithm	
		Average solution time (s)	Average optimality gap (%)	Average solution time (s)	Average optimality gap[a] (%)	Average solution time (s)	Average optimality gap[a] (%)
24	20	**5.3**	**0.0**	6.8	0.0	9.1	0.0
	30	**5.8**	**0.0**	6.7	0.0	7.9	0.0
	40	5.9	0.0	**5.7**	**0.0**	9.7	0.0
41	20	17.0	0.0	**11.5**	**0.0**	21.2	0.0
	30	**16.5**	**0.0**	18.7	0.0	20.4	0.0
	40	14.8	0.0	**9.1**	**0.0**	21.8	0.5
50	20	**21.2**	**0.0**	18.3	1.6	25.3	0.0
	30	**16.4**	**0.0**	17.3	0.0	26.5	0.0
	40	19.8	0.0	**14.9**	**0.0**	25.1	0.0
146	20	219.8	0.9	138.8	5.7	**225.2**	**0.5**
	30	415.7	0.9	130.8	5.7	**155.2**	**0.0**
	40	412.7	0.0	**155.6**	**0.0**	209.8	0.0
195	20	–	–	**255.2**	**0.0**	272.2	0.5
	30	–	–	**209.0**	**0.0**	249.1	0.0
	40	–	–	**175.3**	**0.0**	236.1	0.1
321	20	–	–	**288.1**	**0.0**	1000.2	0.2
	30	–	–	**288.3**	**0.1**	721.4	0.2
	40	–	–	**278.0**	**0.0**	646.1	0.1

[a]Optimality gaps for sub-instances with 321 nodes are calculated with respect to the best known solutions.

followed by the ILS algorithm. On the other hand, matheuristic with first initialization method cannot return any solution within one hour because (TEP-MILP) formulation cannot construct the initial solution in the given time. This is represented via a dash, "–", in the table.

When we compare the solution times of the (PEVEEP-MISOCP) formulation with the heuristic approaches for instances with ≥146 nodes (see Tables 5 and 6), it can be seen that matheuristic with second initialization method and the ILS algorithm find solutions much faster than the exact method. It is remarkable that matheuristic with second initialization method solve all sub-instances with 195 nodes to optimality within 5 min on average while the same takes more than 5 h for the exact method.

Third, the effect of the total time limit on the average speed and total energy consumption of the PEV, the constructed path, and the solution time is analyzed on some sub-instances with 50 and 146 nodes with selected values of SDs. It is expected that the PEV may need to go faster on average under tighter total time limits which may result in an increase in the total energy consumption. Also, the chosen path may subject to change with the change in total time limit. In these experiments, first the total travel time of the PEV under a loose total time limit is computed. When the total time limit is loose, there are alternative optimal solutions where the PEV may stop and wait unnecessarily at some charging stations. To get rid of unnecessary stops, we multiply the total travel time of the PEV with a small positive constant ϵ and add this to the objective function. In our experiments, ϵ is taken as 10^{-3}. Once the total travel time of the PEV under a loose time limit is obtained, we multiply this with a parameter $\alpha \in (0, 1]$ and obtain a tighter total time limit. If α equals one, the total time limit is at its original level. Note that α values are chosen in such a way that the problem is still feasible.

For each sub-instance, the constructed path, objective function value (energy cons. val.), average speed, and solution time are given in Table 7. Each row shows us the instance properties including different α values. Node sizes, SDs, and sub-instance numbers are shown in the first, second, and third columns of the table, respectively. Average speed values are calculated over the network by dividing the sum of the speed values of all arcs to the total number of arcs on the constructed path. It can be seen from the table that the smaller the α values, the larger the average speed values. Solution times are not much affected by changes in the total time limit. In some cases, a decrease in α resulted in a change in the constructed path. For example, if we look at the solutions of the sub-instance 3 with 50 nodes, 30% SD and α values equal to 1 and 0.93, it can be seen that PEV changes its path to achieve lower energy consumption within the given total time limit. These two solutions are shown in Fig. 5 for illustration purposes. In Fig. 5, a subset of all arcs and nodes of the network are displayed and some possible paths between nodes are represented via dashed lines. The origin node is node 1 and the destination node is node 50. Solid thin lines show some of the real arcs of the network while solid thick lines are the arcs of the optimal path of the PEV. Nodes 5, 23, 39, 41, 42, and 49 are among the station nodes. Note that as all nodes are not shown in the figure, there may be some other station nodes in the network. The optimal path is found as $1-15-23-29-37-44-45-50$, when $\alpha = 1$ (see Fig. 5(a)) and is found as $1-15-23-24-29-37-44-45-50$, when $\alpha = 0.93$ (see Fig. 5(b)).

In both cases, the PEV stops at node 23 in order to recharge its battery. When $\alpha = 1$, the total time limit is not restrictive and therefore the PEV drives at a speed which is equal to $min\{max\{V_{opt}, L_{ij}\}, U_{ij}\}$ (V_{opt} = 13.04 m/s) on each arc (i, j) on the constructed path. On the other hand, when $\alpha = 0.93$, the PEV changes its path and needs to speed up on some of the arcs. The speed values of the PEV for both α values are shown in Table 9 together with the upper and lower speed limits of the arcs. The first column in this table shows the arcs, while the second and third columns give the lower and upper speed limits, respectively, the fourth and fifth columns show the speeds of the PEV on the arcs of the constructed path when α equals 1 and 0.93, respectively. A dash, "–", represents that the arc is not on the constructed path. When $\alpha = 1$, while the PEV goes at V_{opt} on arcs $(44, 45)$ and $(45, 50)$, it needs to speed up when $\alpha = 0.93$. In addition, when $\alpha = 0.93$, the PEV also goes faster than $max\{V_{opt}, L_{ij}\}$ on arcs $(1, 15)$, $(23, 24)$, $(24, 29)$, and $(29, 37)$.

A more significant change occurs in the optimal path when we look at the solutions of the sub-instance 2 with 146 nodes, 30% SD and α values equal to 1 and 0.92. In this case, all the nodes except the origin and destination nodes have changed in the optimal constructed path.

To illustrate the trade off between the total time limit and the increase in energy consumption and the average speed, the percent changes in average speed and energy consumption are presented in Fig. 6. The Figs. 6(a), 6(b), and 6(c), from left to right in this figure correspond to the three groups of sub-instances with 50 nodes shown in Table 7 from top to bottom, respectively. Each dot in each sub-figure corresponds to an α value shown in Table 7 (where dot on the origin corresponds to $\alpha = 1$), while the x and y axes show the percent changes in average speed values and the percent changes in the total energy consumption of the PEV, respectively, with respect to the values when

Table 7
Analysis of PEV speed with changing T on networks with 50 and 146 nodes.

|N|	SD	Sub-instance	α	Path found	Energy Cons. Val. (J × 10^6)	Average speed (m/s)	Solution time (s)
50	20	1	1	1-15-23-28-27-35-36-38-46-45-50	100.3	14.3	14.9
			0.98	1-15-23-28-27-35-36-38-46-45-50	100.4	14.6	32.5
			0.93	1-15-23-28-27-35-36-38-46-45-50	101.3	15.9	43.7
			0.87	1-15-24-29-37-44-45-50	105.2	18.4	38.8
50	30	2	1	1-15-24-29-37-44-45-50	91.4	14.8	32.5
			0.91	1-15-24-29-37-44-45-50	93.5	17.3	218.3
			0.90	1-15-24-29-37-44-45-50	94.3	17.8	31.7
			0.89	1-15-24-29-37-44-45-50	95.6	18.6	36.9
50	30	3	1	1-15-23-29-37-44-45-50	96.5	15.6	15.5
			0.99	1-15-23-29-37-44-45-50	96.5	15.7	32.9
			0.94	1-15-23-29-37-44-45-50	97.3	16.7	39.7
			0.93	1-15-23-24-29-37-44-45-50	98.1	17.2	219.7
146	20	1	1	1-69-70-71-72-83-98-99-112-113-127-135-146	117.1	14.9	2482.9
			0.96	1-69-70-71-72-83-98-99-112-113-127-135-146	117.8	15.9	6446.6
			0.94	1-69-70-71-72-83-98-99-112-113-127-135-146	118.7	16.6	6616.4
			0.92	1-69-70-71-72-83-98-99-112-113-127-135-146	120.7	17.7	11 591.1
146	30	2	1	1-69-70-71-72-83-98-99-112-113-127-135-146	117.1	14.9	3549.1
			0.96	1-69-70-71-72-83-98-99-112-113-127-135-146	117.7	15.9	5777.9
			0.94	1-69-70-71-72-83-98-99-112-113-127-135-146	118.7	16.6	6868.4
			0.92	1-79-80-94-108-121-122-123-124-144-145-146	120.7	17.6	5835.4
146	30	3	1	1-79-80-94-95-122-123-124-144-145-146	114.2	15.9	5307.3
			0.98	1-79-80-94-95-122-123-124-144-145-146	114.7	16.6	3289.3
			0.96	1-79-80-94-95-122-123-124-144-145-146	116.2	17.6	4896.1
			0.94	1-79-80-94-95-122-123-124-144-145-146	121.9	20.1	1773.3

Table 8
Comparison of the performances of heuristic solution approaches under changing T values.

|N|	SD	Sub-instance	Matheuristic with first initialization method		Matheuristic with second initialization method		ILS algorithm	
			Average solution time (s)	Average optimality gap (%)	Average solution time (s)	Average optimality gap (%)	Average solution time (s)	Average optimality gap (%)
50	20	1	21.7	0.0	27.0	0.0	**21.7**	**0.0**
	30	2	**19.4**	**0.0**	21.2	0.0	20.8	0.0
	30	3	**16.9**	**0.0**	33.4	0.0	22.7	0.0
146	20	1	182.0	1.5	125.9	7.0	**184.2**	**0.0**
	30	2	224.2	1.9	**137.6**	**1.8**	107.3	3.4
	30	3	157.3	0.0	171.9	0.0	**132.5**	**0.0**

Table 9
Speed values of the sub-instance 3 with 50 nodes and 30% SD, $\alpha = 1$ and $\alpha = 0.93$.

Arc (i, j)	L_{ij} (m/s)	U_{ij} (m/s)	V_{ij} (m/s) $\alpha = 1$	V_{ij} (m/s) $\alpha = 0.93$
(1, 15)	14.17	26.11	14.17	16.55
(15, 23)	18.89	23.61	18.89	18.89
(23, 29)	14.72	29.72	14.72	–
(23, 24)	16.39	22.22	–	16.55
(24, 29)	15.28	30.00	–	16.55
(29, 37)	15.83	24.17	15.83	16.55
(37, 44)	19.17	22.50	19.17	19.17
(44, 45)	11.67	22.78	13.04	16.55
(45, 50)	11.11	33.06	13.04	16.55

$\alpha = 1$. For example, Fig. 6(a) shows that if the value of α decreases from 1 to 0.98, 0.93, and 0.87 for sub-instance 1 with 20% SD, the average speed increases by 2.4, 11.6, and 29.0%, respectively, resulting in an increase in average energy consumption levels, by 0.1, 1.0, and 4.9%, respectively. This expected change is due to the shape of the energy consumption function. The first derivative of this function with respect to speed is positive for speed values that are above the optimal speed (V_{opt}) which means that the change in energy consumption per unit increase in speed is positive if the speed is above the optimal speed. Moreover, the second derivative of this function with respect to speed is also positive which means that the change in energy consumption per unit increase in speed gets larger with higher speed values (that are above the optimal speed).

Fourth, performances of heuristic algorithms are compared on medium (50) and large (146) size instances under tight total time limits on the instances in Table 7. Average optimality gaps and solution times over α values used for that instance are provided in Table 8. In Table 8, rows correspond to instance settings and three groups of columns correspond to matheuristic with first initialization method, matheuristic with second initialization method, and ILS algorithm. The resulting average solution times and optimality gaps are provided in the related columns. Bold numbers in the table reflect the best solution approach in terms of optimality gap where ties are broken by looking at the average solution times. For the node size 50, all instances are solved optimally at most within 34 s. Matheuristic with the second initialization method is dominated by the other alternatives in terms of solution time. For the node size 146, the second sub-instance is solved with small optimality gaps by all heuristic algorithms. The other sub-instances are solved with 0 optimality gap by the ILS algorithm which is the fastest one in this case. Moreover, matheuristic with second initialization method is not able to solve 5 out 24 sub-instances. This is because none of the shortest k paths is time feasible for these 5 sub-instances. In this case, optimality gaps and solution times are calculated excluding these ones.

Lastly, a special network, shown in Fig. 7, is constructed in order to show that the shortest paths are not necessarily the most energy efficient paths. This network has 40 nodes clustered into two groups where the first cluster includes nodes 1 to 20, and the second nodes 21 to 40. These two clusters are connected by the help of bridges introduced between the last six nodes (from 15 to 20) of the first cluster and the first six nodes (from 21 to 26) of the second cluster. Each

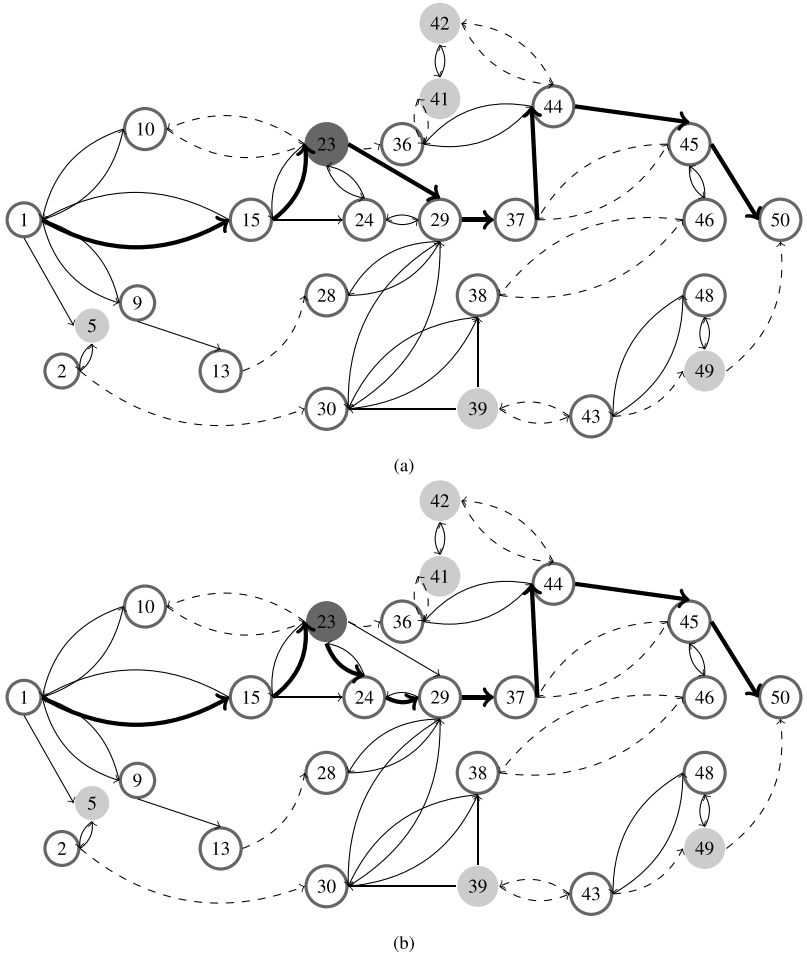

Fig. 5. Solution of the sub-instances 3 with 50 nodes, 30% SD, $\alpha = 1$ (a) and $\alpha = 0.93$ (b).

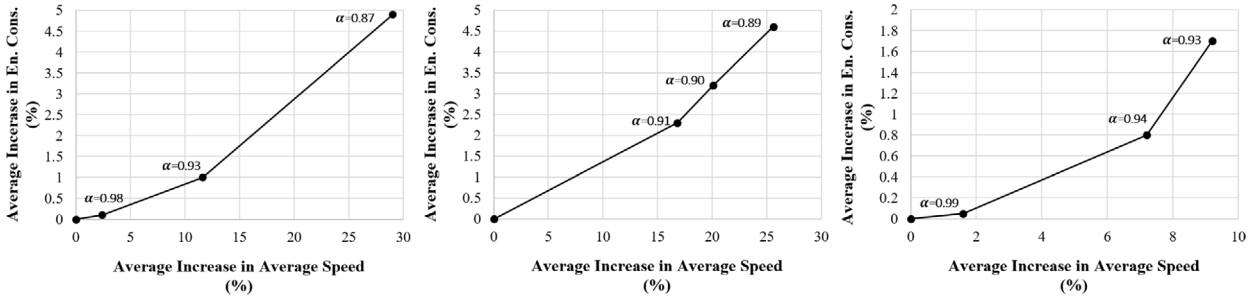

Fig. 6. Average percent change in energy consumption values with respect to changes in average speed values based on different α values for three sub-instances with 50 nodes ((a), (b), and (c) correspond to sub-instances 1, 2, and 3, resp.).

node except nodes 15 to 20 has at most four outgoing arcs. In the first cluster, the arc lengths are distributed uniformly at random between 15 and 20 km. This cluster is assumed to be in a rural area whose road segments are short and not appropriate to drive very fast. In this cluster, for each arc (i, j), L_{ij} follows a uniform distribution between 30 and 50 km/h, and U_{ij} follows a uniform distribution between 60 and 90 km/h. No recharging station is included in this cluster. The second cluster is assumed to be in an urban area whose road segments are longer and appropriate to drive faster. In this cluster, the arc lengths are distributed uniformly at random between 40 and 60 km. Moreover, for each arc (i, j), L_{ij} follows a uniform distribution between 60 and 80 km/h, and U_{ij} follows a uniform distribution between 90 and 120 km/h. As the network is sparse and the arc lengths and speed limits are randomly generated, some nodes turn out to never appear in short paths. For example, nodes 23 and 26 do not show up in the shortest 10,000 paths. Assuming that the charging stations are located at nodes 23 and 26 and taking an appropriate tight total time limit, all of the shortest 10,000 paths are infeasible. This instance shows that enumerating the short paths does not guarantee a feasible solution for the PEVEEP.

For the constructed network, as there are only a few feasible paths between origin and destination, the proposed solution approaches have difficulties in finding a solution. The solutions obtained by using PEVEEP-MISOCP, matheuristic with first initialization method, and ILS algorithm are given in Table 10. In addition to the network properties

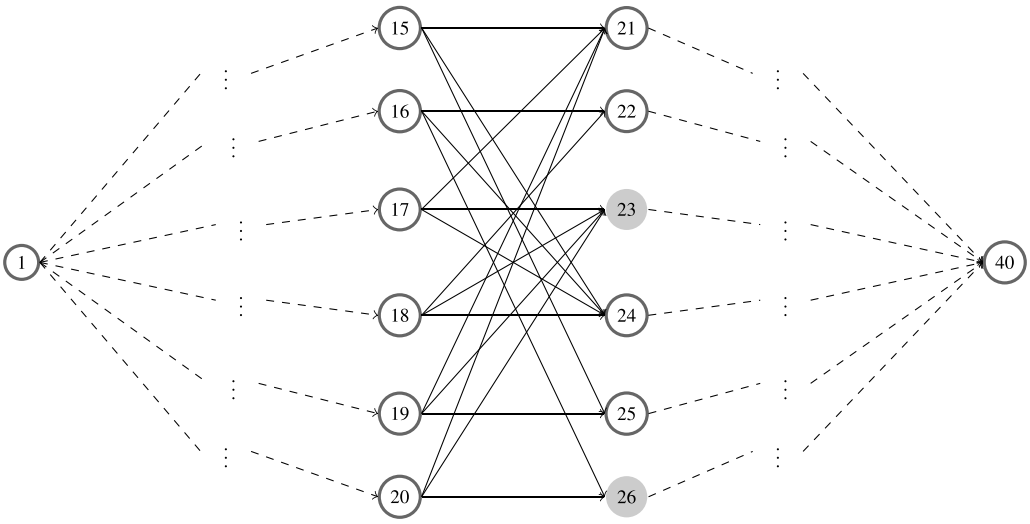

Fig. 7. A special network example.

Table 10
Analysis on a special network with 40 nodes.

|N|	SD	Algorithm used	Path found	Energy consumption value (J × 10^6)	Solution time (s)	Optimality gap (%)
40	40	Exact algorithm	1-4-7-10-13-16-23-27-28-32-36-40	163.4	433.5	0.0
		Matheuristic with first initialization	1-4-6-9-12-15-23-27-34-35-39-40	165.0	41.7	1.0
		ILS algorithm	1-4-7-10-13-16-23-27-34-35-39-40	164.2	20.0	0.5

displayed in the first two columns, one can see the used solution approaches, the constructed paths, the total energy consumption values, the solution times, and the optimality gaps in the remaining columns of Table 10. The objective function value is equal to 163.4 × 10^6 J in the optimal solution. This solution is obtained in approximately 434 s using PEVEEP-MISOCP formulation. If we compare this network with the other networks of similar node sizes in Table 6, the solution time is much larger here. This may be because of the mentioned complexity of the constructed network. Moreover, the ILS algorithm performs better than the matheuristic with first initialization method. Its optimality gap and solution time are smaller. ILS algorithm is preferable when it is compared to the matheuristic with first initialization method, but both heuristic methods can be used in place of PEVEEP-MISOCP in order to get a solution in a shorter period of time. As matheuristic with second initialization method starts with a randomly chosen shortest path, it is unable to solve this instance. When it is executed, it cannot construct an initial solution when k is ≤10,000 due to not being able to find an energy feasible path.

7. Conclusion

Electric vehicles are important in decarbonizing transportation. Their usage rates increase day by day because of the economic and environmental benefits. Both theoretically challenging and practically important problems arise in routing electric vehicles. In this direction, a problem about energy optimization of a plug-in electric vehicle (PEV) that aims to find a path between origin and destination over a network is studied in this paper; namely, the PEVEEP. The stations the PEV stop by, the charge amounts, and the speed the PEV goes along each arc are decided in addition to the path joining origin and destination. In the literature, all of the studies trying to find an energy efficient path use speed as a parameter. Here, it is considered as a continuous decision variable. The energy consumption function used in this study is a nonlinear function of the speed. This function represents the behavior of the battery on the PEV based on some parameters.

An MISOCP formulation is used to optimally solve the PEVEEP. It is seen that the formulation becomes inadequate for larger instances. Hence, some heuristic approaches are developed for the problem as well. The developed matheuristic is composed of three steps each using a different mathematical formulation. Based on the initialization method, two versions of the matheuristic are proposed. The matheuristic usually returns optimal solutions quickly for small size instances for both initialization methods. For larger size instances, matheuristic with second initialization method outperforms the other. In addition, an ILS algorithm is developed for the PEVEEP which is good at finding optimal or near optimal solutions faster for large size instances compared to the exact algorithm.

Computational experiments show that the PEV needs to go faster under tighter total time limits by sacrificing some energy. Another managerial insight of the results of our experiments is that the optimal path of a PEV may be affected by the total time limit restriction. An optimal path may become non-optimal with a decrease in total time limit which shows that one cannot fix a path in advance and always use it independent of the total time limit. Hence, solving the PEVEEP after speed limits (legal or based on congestion) and the total time limit are realized is crucial for energy efficiency.

The studied problem and the proposed solution approaches may find use in energy efficient path finding for EVs in real life. The heuristic solution approaches may even be integrated in some software to dynamically find energy efficient paths thanks to their reasonable computational times. For example, instances with 321 nodes can be solved within 5 min using the matheuristic with second initialization method. For problems involving a fleet of PEVs, e.g., an electric bus fleet of a municipality, the solution approaches proposed in this paper can be used as a foundation. Likewise, they can be adapted to solve similar problems involving different types of vehicles with electric engines, e.g., auto-guided vehicles (AGV), unmanned aerial vehicles (UAV), and autonomous mobile robots (AMR).

The problem and the solution approaches studied in this paper may have some limitations. In this study, the provided speed limits are not time dependent. However, in real life, due to traffic congestion, the speed limits may dynamically change over time and hence decisions should be updated under dynamically changing speed limits. Moreover,

there may be some uncertainties in charging infrastructures in real life. During the trip of the PEV, if a charging station is out of service or occupied by a crowd of other users, the problem instance may need to be updated accordingly.

As a future work, a time-dependent version of this problem can be studied where the speed limits of the arcs are subject to change by time. In our problem, we only considered a single PEV. In future studies, a fleet of PEVs can be considered and different versions of the vehicle routing problem can be studied. Moreover, the developed solution approaches may be modified to find energy efficient paths for different types of vehicles including AGVs, AMRs, and UAVs.

CRediT authorship contribution statement

Bilgenur Erdoğan: Conceptualization, Methodology, Software, Data curation, Formal analysis, Writing – original draft, Writing – review & editing. **Mustafa Kemal Tural:** Supervision, Investigation, Validation, Conceptualization, Methodology, Writing – review & editing. **Arsham Atashi Khoei:** Investigation, Validation, Methodology, Writing – original draft, Writing – review & editing.

Data availability

Data will be made available on request.

References

Abousleiman, R., & Rawashdeh, O. (2014). *Energy efficient routing for electric vehicles using particle swarm optimization: Technical report SAE technical paper*.

Abousleiman, R., Rawashdeh, O., & Boimer, R. (2017). Electric vehicles energy efficient routing using ant colony optimization. *SAE International Journal of Alternative Powertrains*, *6*, 1–14.

Alizadeh, F., & Goldfarb, D. (2003). Second-order cone programming. *Mathematical Programming*, *95*, 3–51.

Arslan, O., Yıldız, B., & Karaşan, O. E. (2015). Minimum cost path problem for plug-in hybrid electric vehicles. *Transportation Research Part E: Logistics and Transportation Review*, *80*, 123–141.

Bac, U., & Erdem, M. (2021). Optimization of electric vehicle recharge schedule and routing problem with time windows and partial recharge: A comparative study for an urban logistics fleet. *Sustainable Cities and Society*, *70*, Article 102883.

Bektaş, T., & Laporte, G. (2011). The pollution-routing problem. *Transportation Research, Part B (Methodological)*, *45*, 1232–1250.

Belenguer, J.-M., Benavent, E., Lacomme, P., & Prins, C. (2006). Lower and upper bounds for the mixed capacitated arc routing problem. *Computers & Operations Research*, *33*, 3363–3383.

Bruglieri, M., Pezzella, F., Pisacane, O., & Suraci, S. (2015). A matheuristic for the electric vehicle routing problem with time windows. arXiv preprint arXiv:1506.00211.

De Cauwer, C., Van Mierlo, J., & Coosemans, T. (2015). Energy consumption prediction for electric vehicles based on real-world data. *Energies*, *8*, 8573–8593.

Demir, E., Bektaş, T., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, *223*, 346–359.

Demir, E., Bektaş, T., & Laporte, G. (2014). The bi-objective pollution-routing problem. *European Journal of Operational Research*, *232*, 464–478.

Dijkstra, E. W., et al. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*, 269–271.

Erdelić, T., & Carić, T. (2019). A survey on the electric vehicle routing problem: variants and solution approaches. *Journal of Advanced Transportation*, 2019.

Franceschetti, A., Honhon, D., Van Woensel, T., Bektaş, T., & Laporte, G. (2013). The time-dependent pollution-routing problem. *Transportation Research, Part B (Methodological)*, *56*, 265–293.

Gouveia, L., Mourão, M. C., & Pinto, L. S. (2010). Lower bounds for the mixed capacitated arc routing problem. *Computers & Operations Research*, *37*, 692–699.

Hashimoto, H., Yagiura, M., & Ibaraki, T. (2008). An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, *5*, 434–456.

IBM (2021). Cplex optimizer (12.10). URL: https://www.ibm.com/support/pages/downloading-ibm-ilog-cplex-optimization-studio-v12100.

IEA, I., UNSD, W., et al. (2019). *Tracking sdg 7: the energy progress report 2019*. Washington DC: WHO.

Kramer, R., Subramanian, A., Vidal, T., & Lucídio dos Anjos, F. C. (2015). A matheuristic approach for the pollution-routing problem. *European Journal of Operational Research*, *243*, 523–539.

Krause, R. M., Carley, S. R., Lane, B. W., & Graham, J. D. (2013). Perception and reality: Public knowledge of plug-in electric vehicles in 21 US cities. *Energy Policy*, *63*, 433–440.

Kucukoglu, I., Dewil, R., & Cattrysse, D. (2021). The electric vehicle routing problem and its variations: A literature review. *Computers & Industrial Engineering*, Article 107650.

Kumar, S. N., & Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. *Intelligent Information Management*, 66–74.

Li, Y., Zhang, L., Zheng, H., He, X., Peeta, S., Zheng, T., et al. (2017). Nonlane-discipline-based car-following model for electric vehicles in transportation-cyber–physical systems. *IEEE Transactions on Intelligent Transportation Systems*, *19*, 38–47.

Lin, C., Choy, K. L., Ho, G. T., Chung, S. H., & Lam, H. (2014). Survey of green vehicle routing problem: past and future trends. *Expert Systems with Applications*, *41*, 1118–1138.

Liu, T., Tan, W., Tang, X., Zhang, J., Xing, Y., & Cao, D. (2021). Driving conditions-driven energy management strategies for hybrid electric vehicles: A review. *Renewable and Sustainable Energy Reviews*, *151*, Article 111521.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics* (pp. 320–353). Springer.

Lu, J., Chen, Y., Hao, J.-K., & He, R. (2020). The time-dependent electric vehicle routing problem: Model and solution. *Expert Systems with Applications*, Article 113593.

Ma, B., Hu, D., Chen, X., Wang, Y., & Wu, X. (2021). The vehicle routing problem with speed optimization for shared autonomous electric vehicles service. *Computers & Industrial Engineering*, Article 107614.

Montoya, A., Guéret, C., Mendoza, J. E., & Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research, Part B (Methodological)*, *103*, 87–110.

Regulation, E. (2018). *Regulation (EU) 2018/1999 of the European parliament and of the council of 11 2018 on the governance of the energy union and climate action, amending regulations (EC) No 663/2009 and (EC) No 715/2009 of the european parliament and of the council: Technical report directives 94/22/EC, 98/70/EC, 2009/31/EC, 2009/73/EC, 2010/31/EU, 2012/27 ...*.

Sachenbacher, M., Leucker, M., Artmeier, A., & Haselmayr, J. (2011). Efficient energy-optimal routing for electric vehicles. In *Twenty-fifth AAAI conference on artificial intelligence*.

Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, *48*, 500–520.

Strehler, M., Merting, S., & Schwan, C. (2017). Energy-efficient shortest routes for electric and hybrid vehicles. *Transportation Research, Part B (Methodological)*, *103*, 111–135.

Sweda, T. M., & Klabjan, D. (2012). Finding minimum-cost paths for electric vehicles. In *2012 IEEE international electric vehicle conference* (pp. 1–4). IEEE.

Tiwari, V., Aditjandra, P., & Dissanayake, D. (2020). Public attitudes towards electric vehicle adoption using structural equation modelling. *Transportation Research Procedia*, *48*, 1615–1634.

Vincent, F. Y., Redi, A. P., Hidayat, Y. A., & Wibowo, O. J. (2017). A simulated annealing heuristic for the hybrid vehicle routing problem. *Applied Soft Computing*, *53*, 119–132.

Wu, X., Freese, D., Cabrera, A., & Kitch, W. A. (2015). Electric vehicles' energy consumption measurement and estimation. *Transportation Research Part D: Transport and Environment*, *34*, 52–67.

Wu, X., He, X., Yu, G., Harmandayan, A., & Wang, Y. (2015). Energy-optimal speed control for electric vehicles on signalized arterials. *IEEE Transactions on Intelligent Transportation Systems*, *16*, 2786–2796.

Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *Management Science*, *17*, 712–716.

Zhou, B., & Zhao, Z. (2022). Multi-objective optimization of electric vehicle routing problem with battery swap and mixed time windows. *Neural Computing and Applications*, 1–24.