Expert Systems with Applications 161 (2020) 113593

Contents lists available at ScienceDirect

Expert Systems with Applications

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

The Time-dependent Electric Vehicle Routing Problem: Model and solution

Ji Lu[a], Yuning Chen[b,*], Jin-Kao Hao[c], Renjie He[b]

[a] National University of Defense Technology (NUDT), Changsha, China
[b] National University of Defense Technology, Changsha, China
[c] University of Angers, France

ARTICLE INFO

Article history:
Received 17 October 2019
Revised 22 May 2020
Accepted 22 May 2020
Available online 9 June 2020

Keywords:
Green logistics
Time-dependent vehicle routing
Efficient constraint handling
Congestion

ABSTRACT

We study a new problem named the Time-dependent Electric Vehicle Routing Problem (TDEVRP) which involves routing a fleet of electric vehicles to serve a set of customers and determining the vehicle's speed and departure time at each arc of the routes with the purpose of minimizing a cost function. We propose an integer linear programming (ILP) model to formulate the TDEVRP and show that the state-of-the-art commercial optimizer (CPLEX) can only solve instances of very limited sizes (with no more than 15 customers). We thus propose an iterated variable neighbourhood search (IVNS) algorithm to find near-optimal solutions for larger instances. The key ingredients of IVNS include a fast evaluation method that allows local search moves to be evaluated in constant time $O(1)$, a variable neighbourhood descent (VND) procedure to optimize the node sequences, and a departure time and speed optimization procedure (DSOP) to optimize the speed and departure time on each arc of the routes. The proposed algorithm demonstrates excellent performances on a set of newly created instances. In particular, it can achieve optimal or near-optimal solutions for all small-size instances (with no more than 15 customers) and is robust for large-size instances where the gap between the average and the best solution value is consistently lower than 2.38%. Additional experimental results on 40 benchmark instances of the closely related Time-Dependent Pollution Routing Problem indicate that the proposed IVNS algorithm also performs very well and even discovers 39 new best-known solutions (improved upper bounds).

© 2020 Published by Elsevier Ltd.

1. Introduction

Recent years have witnessed a rapid increase in the use of electric vehicles, as many countries have passed regulations to restrict the production of fossil vehicles in order to reduce greenhouse gas emissions. With the advance of electric vehicle technology and social awareness of global warming, there is an increasing number of people choosing electric vehicles to replace conventional fossil vehicles. Moreover, governments and companies are continually improving the required infrastructures to boost the popularization of electric vehicles.

Electric vehicles are powered by electric engines and they use batteries to store energy. They produce no gas emission and little noise, and the required recharge and maintenance fees are usually lower than those of the traditional fossil vehicles. However, as indicated in Schneider, Stenger, and Goeke (2014), short driving range and long recharging time are two main factors that handicap the expansion of electric vehicles. Also, the requirement of frequent recharging inevitably leads to extra travel distance of electric vehicles.

Additionally, traffic congestion is another important factor that disadvantages the use of electric vehicles (Figliozzi, 2011). On the one hand, congestion leads to long travel time, which induces extra costs (e.g., late arrival at customers in logistics applications) (Franceschetti, Honhon, Woensel, Bektas, & Laporte, 2013). On the other hand, congestion increases the amount of energy consumed by electric vehicles and further limits their driving range. Efficient routing algorithms for electric vehicles under congestion will mitigate these difficulties and thus ease the utilization of electric vehicles in general and in logistics applications especially.

There exist numerous studies on various electric vehicle routing problems and their extensions with time window constraints (Caceres-Cruz, Arias, Guimarans, Riera, & Juan, 2014). However, to the best of our knowledge, the electric vehicle routing problem with *time dependency constraint* has not been investigated in the literature. In this work, we take the first step by studying a new

* Corresponding author.
E-mail addresses: 15580814918@163.com (J. Lu), cynnudt@hotmail.com (Y. Chen), hao@univ-angers.fr (J.-K. Hao), renjiehe@nudt.edu.cn (R. He).

https://doi.org/10.1016/j.eswa.2020.113593
0957-4174/© 2020 Published by Elsevier Ltd.

problem named the Time-dependent Electric Vehicle Routing Problem (TDEVRP), which includes a number of important features that are relevant to real-life applications of electric vehicles:

- Congestion Period. We define two congestion periods in the start and end of a given planning horizon, which typically correspond to the morning peak and evening peak of a day.
- Recharging Decision. The limited driving range of electric vehicles and the increase of energy consumption caused by congestion require frequent recharging. The decisions on when and which station to recharge are essential for minimizing the total cost.
- Energy Consumption. In line with Bektas and Laporte (2011) and Demir, Bektas, and Laporte (2012), we adopt the energy consumption model proposed by Barth, Younglove, and Scora (2005) and Barth, Wu, and Boriboonsomsin (2015) where the weight, the travel speed, and the travel distance of a given vehicle determine the energy consumption of a trip.

The TDEVRP also integrates two realistic models: the time-dependent vehicle routing problem (TDVRP) and the electric vehicle routing problem with time window (EVRPTW). As such, the proposed problem is a useful model to encompass more real-life applications of electric vehicles.

1.1. Related work

Our proposed TDEVRP is built on top of the basic electric vehicle routing model with additional constraints such as time window, vehicle capacity and congestion. This study belongs to the domain of green logistics (Akyelken, 2016), where researchers from different communities have conducted much research. We provide below a brief review of recent and closely related works of this study. For a comprehensive survey of green VRP and time-dependent routing problems, we refer the readers to Lin, Choy, Ho, Chung, and Lam (2014) and Gendreau, Ghiani, and Guerriero (2015).

The study of greenhouse gas emissions in logistics began with the investigation of the Pollution Routing Problem (PRP) (Bektas & Laporte, 2011). The goal of the PRP is to minimize the greenhouse gas emission of the vehicles. Bektas and Laporte (2011) indicated that the speed of a vehicle and the traffic condition should be considered when one calculates the energy consumption and greenhouse gas emission for a trip. An important feature of the PRP is the flexible speed model that allows drivers to drive at any speed below a maximum value. Another feature of the PRP is the energy consumption model which takes into account realistic factors such as travel distance, travel speed and vehicle weight. To make our TDEVRP close to reality, we adopt the energy consumption and flexible speed model.

Malandraki and Daskin (1992) and Malandraki and Dial (1996) first proposed a mixed interger formulation for time-dependent vehicle routing problem(TDVRP) and a heuristic algorithm to solve the problem. Ichoua, Gendreau, and Potvin (2003) proposed a travel time model which guarantees the satisfaction of first-in-first-out(FIFO) property. Kok, Hans, and Schutten (2012) and Figliozzi (2011) studied the impact of congestion including late arrival at customers and increased costs. These early works on TDVRP, opposed to our work in this paper, did not take into account the flexible travel speed. Franceschetti et al. (2013) proposed the Time-Dependent Pollution Routing Problem (TDPRP), which extends the PRP by taking into account the effects of traffic congestion on the routing of vehicles and green house gas emission. The authors derived a set of characterizations of the optimal solution for a single-arc version of the TDPRP and introduced strategies to alleviate the influence of congestion. Their study also revealed the influences brought by the congestion, such as a dramatically increased fuel consumption and possible post-service delay at customer nodes.

Erdogan and Miller-Hooks (2012) introduced a Green Vehicle Routing Problem (GVRP) where a fleet of mixed fueled vehicles is scheduled to reduce environmental impact. Schneider et al. (2014) presented an electric vehicle routing problem with time windows where only electric vehicles are involved. However, as pointed out by Schneider et al. (2014), the limited driving range causes great challenges on the real-life applications of electric vehicles, especially in industrial fields like logistics. Goeke and Schneider (2015) studied the effectiveness of routing a mixed fleet of electric and conventional vehicles. Also, many extensions have been investigated on the application of electric vehicles with real-life constraints such as using battery swapping stations (Chen, Qi, & Miao, 2016), adopting partial recharging strategy (Keskin & Catay, 2016) and using a mixed fleet of heterogeneous electric vehicles (Hiermann, Puchinger, Ropke, & Hartl, 2016). We note that all the above studies on electric vehicles assume a constant speed.

As technology advances, the capability of electric vehicles is being enhanced. The issue related to electric vehicle routing becomes increasingly important for their applications in logistics. Since traffic congestion significantly increases the energy consumption of electric vehicles, it is a primordial factor that should be considered when studying electric vehicle routing problems such as the EVRPTW. Keskin, Laporte, and Catay (2019) considered the capacity of charging stations and modeled the time-dependent waiting time at charging stations. Shao, Guan, Ran, He, and Bi (2017) proposed a electric vehicle routing problem with charging time and variable travel time. Although very recent, these studies still assume a constant speed. However, Franceschetti et al. (2013) proved in their paper that the optimal speed is not a constant value, but a flexible value that can vary below a maximum speed. Given this result, we adopt the flexible speed model and take into consideration the influence of the speed variation on energy consumption.

In terms of the solution methods for electric vehicle routing problems with constant speed, a number of effective algorithms exist in the literature. Most of them fall into the category of heuristic methods due to the NP-hardness of the problems. Nagata, Braysy, and Dullaert (2010) introduced a time travel approach which enables efficient evaluation of many local search operators. A dedicated operator for the EVRPTW was present in Schneider et al. (2014). A hybrid genetic algorithm was introduced in Vidal, Crainic, Gendreau, and Prins (2013) to solve a large class of routing problems, based on which Hiermann et al. (2016) developed a large neighbourhood search heuristic combined with a dedicated local search procedure to solve the EVRPTW with a heterogeneous fleet.

For studies with flexible speed, Kramer, Maculan, Subramanian, and Vidal (2015a, 2015b) developed a departure time and speed optimization procedure (DSOP) to optimize speed and departure time on each arc, which serves as a sub-subroutine in an adaptive large neighbourhood search heuristic (ALNS) for the PRP (Demir et al., 2012). Franceschetti et al. (2013) and Franceschetti et al. (2016) extended these techniques to the TDPRP and reported high-quality results on benchmark instances.

After a thorough investigation of the literature, we find that efficient local search algorithms are still missing in the literature for the VRP variants with an energy model considering multiple factors(e.g., weight, speed, distance). This is because an efficient evaluation method for local search operators is not available. The state-of-the-art algorithm relies on destroy operators and repair operators under the ALNS framework. We believe the performance of this algorithm can be further improved if a local search procedure is appropriately incorporated. The variable neighborhood search serves as a framework for local search operators and illustrates good performance for many VRP variations including electric vehicle routing problem with mixed fleet (Hiermann et al., 2016; Rezgui, Siala, Aggounemtalaa, & Bouziri, 2019), vehicle routing problem with divisible deliveries and pickups (Kalayci & Kaya, 2016; Polat, 2017) and other variations (Xu & Cai, 2018). Therefore we adopt the IVNS as the general framework of our algorithm (see Section 3).

The main motivation of this work is to study the routing problem of electric vehicles with congestion, flexible speed and a realistic energy consumption model. This model is close to real-life applications but absent in the literature. An efficient solving algorithm for instances with realistic sizes is also required to improve the transportation capability of electric vehicles. Moreover, we seek to enhance the solving efficiency of VRP variations with time-dependent constraints(e.g. TDPRP, TDEVRP) by introducing new efficient evaluation method for local search operators.

1.2. Contributions of this work

The main contributions of this work are summarized as follows:

1. Problem and model: The Time-dependent Electrical Vehicle Routing Problem (TDEVRP) proposed in this work integrates the well-studied TDVRP and EVRPTW which considers recharging decisions, congestion condition, and other traditional factors, such as capacity and time window constraint. An integer linear programming (ILP) model is presented to formulate the problem, which enables the use of general ILP solvers such as CPLEX.
2. Solution algorithm: We develop an iterated variable neighbourhood search (IVNS) algorithm which integrates a set of complementary search components including an initial solution construct process, a variable neighborhood descent (VND), a perturbation procedure and a departure time and speed optimization process (DSOP). All these algorithm components are specially designed for the proposed TDEVRP.
3. Incremental evaluation method: we propose a concatenation operator that allows O(1) evaluation for local moves of all adopted standard neighborhood structures, which greatly speeds up the search process of our proposed IVNS algorithm. We show also that the method can be easily generalized to other time-dependent problems (e.g., agile satellite scheduling (Chu, Chen, & Tan, 2017), time-dependent orientation problem (Gavalas, Konstantopoulos, Mastakas, Pantziou, & Vathis, 2015)).
4. Benchmark instances: We define a set of TDEVRP benchmark instances that we make publically available. These instances are adapted from the EVRPTW benchmark instances proposed by Goeke and Schneider (2015) by introducing time-dependent features(Congestion).
5. Computational results: We perform intensive computational experiments with the proposed IVNS algorithm and the general ILP solver CPLEX. Results show that while CPLEX is successful for small instances with up to 15 nodes, IVNS is indispensable for solving larger instances. We also test IVNS on the set of 40 benchmark instances of the tightly-related TDPRP. The results indicate that IVNS not only dominates the state-of-the-art ALNS algorithm but also discovers new best solutions (improved upper bounds) for 39 instances.

The rest of the paper is organized as follows. A detailed description of the TDEVRP and its ILP formulation is presented in Section 2. Section 4 introduces the concatenation operators proposed in this work, based on which we develop an variable neighbourhood search algorithm in Section 3. Section 5 presents the computational results and the conclusions are drawn in Section 6.

2. Problem description

The TDEVRP is defined on a complete graph $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ where a set of customers, a set of recharging stations, and a depot are included. We denote the set of customers as $\mathcal{C}$, the set of recharging stations as $\mathcal{F}$ and the depot as $\mathcal{N}_0$. We use $\mathcal{F}'$ as a set of dummy nodes representing the recharging stations in $\mathcal{F}$ multiple times. This is because a station may be visited more than once in a day and multiple instances of the station are needed to distinguish these visits. We define $\mathcal{N}'_0$ as a set of dummy nodes of the depot for the same reason. With these definition, we can define $\mathcal{N}$ as $\mathcal{N} = \mathcal{N}'_0 \cup \mathcal{C} \cup \mathcal{F}'$. $\mathcal{A}$ is the set of arcs connecting each pair of nodes in $\mathcal{N}$.

A homogeneous fleet of K vehicles is available to serve all customers, and each vehicle is characterised by a capacity limit of Q units, an energy capacity of Y units and an acquisition cost of h units. We assume that a vehicle would fully recharge itself each time it visits a recharging station and a fixed recharging time E is required. Each customer i in $\mathcal{C}$ is associated with a non-negative demand q_i, a fixed service time p_i and a service time window $[l_i, u_i]$ in which the service must start. If a vehicle arrives at customer i earlier than l_i, it waits until l_i and starts service. The start and end of the planning horizon are denoted as l_0 and u_0. Without loss of generality, we assume l_0 equals 0.

The goal of the TDEVRP is to serve each customer within their service time window under congestion conditions and minimize the cost function. The cost of a trip consists of three parts: an acquisition cost for the vehicle, driver's wages, and the energy cost. Driver's wage is dependent on the duration of the trip, which covers the period from the departure from the depot until the return to the depot. The energy consumption model, as proposed in Franceschetti et al. (2013), is decided by three factors: speed, distance, and load. Unlike traditional VRP, under congestion situation, it may be better to wait at customer nodes after service until the end of congestion than to continue travel immediately (Franceschetti et al., 2013). We call this period of waiting as a post-service delay. Therefore finding a routing plan is to decide: (i) the set of vehicle routes starting and ending at the depot, (ii) the speed on each arc, (iii) the departure time from each node. As an extension of the traditional NP-hard Vehicle Routing Problem (VRP) (Laporte, 1992), the TDEVRP is computationally challenging.

In what follows, the time-dependency character of the problem is introduced in Section 2.1 and the cost function is introduced in Section 2.2. Then an ILP model is presented in Section 2.3.

2.1. Time-dependency

In the TDEVRP, the travel speed is dependent on the departure time. In congestion periods, a vehicle runs at a low speed, which in turn increases the driving time. In the work of Franceschetti et al. (2013) and Jabali, Woensel, and Kok (2012), a two-level speed function is used to model the cases where the congestion period is followed by the free-flow period. Such a function models the cases where congestion happens in the first half of the day.

In this work, we model the congestion by a three-level speed function, as shown in Fig. 1. There are two congestion periods at the beginning and end of the planning horizon, corresponding to the morning and evening peak hours when congestion usually happens. Starting from 0, the morning peak period lasts until T_1 with a congestion speed v_c. Then a free-flow period follows until T_2 with a free-flow speed limit v_f $(v_f > v_c)$. The evening peak period lasts

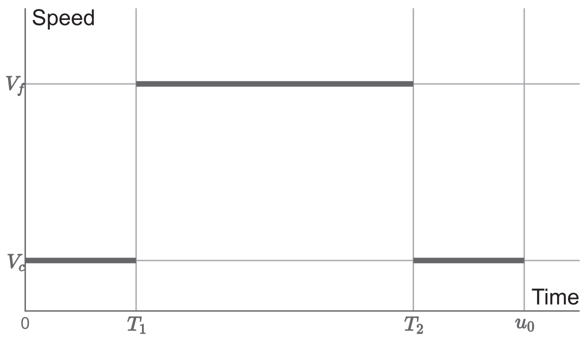

Fig. 1. Time-dependent speed profile. The planning horizon consists of a congestion period of length T_1, a free-flow period of length $T_2 - T_1$ and another congestion period of length $u_0 - T_2$.

until the end of the planning horizon at u_0. Vehicles cannot move at a speed level higher than the given speed limit of each period.

As v_c has a low value, driving during congestion with a speed lower than v_c is never optimal (Franceschetti et al., 2013), which means the speed in congestion should always be v_c. However the best travel speed on free-flow period can be a value lower than v_f and should be decided properly to reduce the travel cost.

As a result, the time needed to travel across a distance of d units is dependent on the departure time and travel speed. We adopt the method proposed by Ichoua et al. (2003) to calculate the travel time. This method does not assume a constant speed over the entire length of an arc. Instead, the speed changes when the boundary between two consecutive time periods is crossed so that the FIFO property is guaranteed. Let $T(t_s, v, d)$ denote the duration of a trip, which starts from t_s and travels across a distance of d units with free-flow speed v. Assume $v = v_f$, we then have the following piecewise function ($d < min((T_2 - T_1)v_f, T_1 v_c)$) (see Fig. 2):

$$T(t_s, v_f, d) = \begin{cases} \frac{d}{v_c}, & 0 \leqslant t_s \leqslant T_1 - \frac{d}{v_c} \\ T_1 - t_s + \frac{d - v_c(T_1 - t_s)}{v_f}, & T_1 - \frac{d}{v_c} \leqslant t_s \leqslant T_1 \\ \frac{d}{v_f}, & T_1 \leqslant t_s \leqslant T_2 - \frac{d}{v_f}, \\ T_2 - t_s + \frac{d - v_f(T_2 - t_s)}{v_c}, & T_2 - \frac{d}{v_f} \leqslant t_s \leqslant T_2 \\ \frac{d}{v_c}, & T_2 \leqslant t_s \leqslant u_0 \end{cases} \quad (1)$$

2.2. Cost function

To model the energy consumption of traveling under congestion, we modify the energy consumption model introduced by Bektas and Laporte (2011). Specifically, we replace fossil fuel in the model with electric power. The model describes the energy consumption of a vehicle as a function of speed, weight, distance, and other factors. In line with Franceschetti et al. (2016), we consider speed, weight, and distance as contributory factors, and we set other parameters to be 0.

Let $E(d, v, f)$ denote the energy consumed by a vehicle travelling distance d at speed v with load f, which is given by:

$$E(d, v, f) = \alpha(u + f)d + \beta\frac{d}{v} + \gamma d v^2, \quad (2)$$

The energy consumption is divided into three parts: weight module, engine module, and speed module corresponding to the first, second, and third term of the equation (Franceschetti et al., 2013).

We have μ as curb weight, δ as driver's wage, and λ as electric price per unit. Also, α, β, γ are vehicle constants which are determined by vehicle parameters whose settings are listed in Table 1

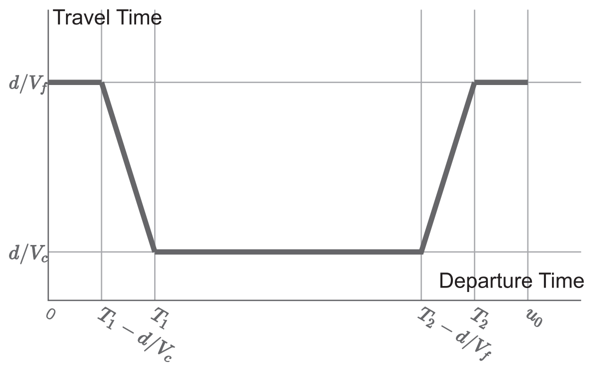

Fig. 2. Time-dependent travel time profile. There are a number of critical points at which the gradient changes: 0, $T_1 - \frac{d}{v_c}$, T_1, $T_2 - \frac{d}{v_f}$, T_2, u_0. 0, T_1, T_2 and u_0 are the adjacent points of different speed levels defined by Fig. 1. $[T_1 - \frac{d}{v_c}, T_1]$ and $[T_2 - \frac{d}{v_f}, T_2]$ are two transition periods in which the trip covers both the congestion period and the free-flow period.

Table 1
Cost function parameter settings. The values listed in the table are calculated with the same methods as Franceschetti et al. (2013).

Notation	Description	Unit	Value
α	weight module constant	m/s^2	0.27272
β	engine module constant	kJ/s	33
γ	speed module constant	kg/m	4.58339
λ	energy cost per unit	£/kwh	0.14
δ	driver's wage per unit time	£/s	0.0022
μ	curb weight	kg	6350
h	vehicle acquisition cost	£	200

and the calculation methods of these values are referred to Franceschetti et al. (2013).

Let $C(d, v, f)$ denote the total travelling cost including the acquisition cost of vehicles, drivers' wage and energy cost, then we get a function similar to Eq. (2):

$$C(d, v, f) = \lambda\alpha(\mu + f)d + (\lambda\beta + \delta)\frac{d}{v} + \lambda\gamma d v^2 + h, \quad (3)$$

Eqs. (2) and (3) give the relationship between cost and influential factors including speed, load and distance. To better illustrate the components of the cost function, we give an illustration of the cost of each component for driving 100 km under different speed values (e.g., 1 m/s, 25 m/s). The result is shown in Table 2.

From the table we see that when driving at a low speed, the engine module will cause a great cost. The engine module cost contributes more than 80% to the total cost of low speed driving. And the driver cost is higher as the journey takes more time. While at high speed, the main cost is the speed module cost, which contributes around 40% to the total cost. We see that when the speed changes from 1 m/s to 25 m/s, the engine module has decreased

Table 2
Illustration of different components of cost. The result shown is the cost consumed by different cost components. The cost is calculated as an empty vehicle drives 100 km with the speed of 1 m/s and 25 m/s respectively.

Description	Cost of 1 m/s(£)	Cost of 10 m/s(£)
weight module cost ($\lambda\alpha\mu d$)	67.34	67.34
engine module cost ($\lambda\beta\frac{d}{v}$)	1283	51.32
speed module cost ($\lambda\gamma d v^2$)	0.1782	111.4
driver cost ($\delta\frac{d}{v}$)	220	22
vehicle acquisition cost	h	h

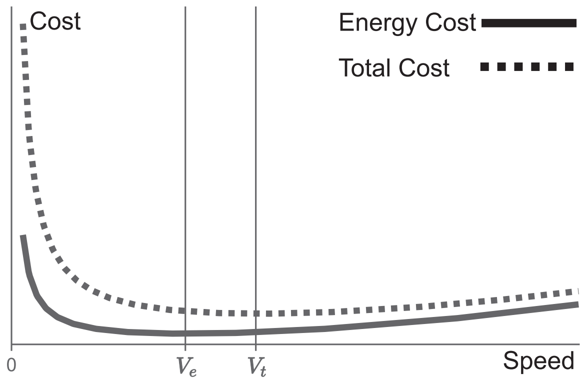

Fig. 3. Travel cost per unit distance as a function of speed. Two important speed values (Franceschetti et al., 2013) are presented: V_e which minimizes the total energy consumed and V_t which minimizes the total cost.

from a very large value to a small one while the speed module has increased from a very small value to a larger one. The weight module stay unchanged and driver cost decreases too. This result echos Eq. 3 as the engine module and driver cost is related with v^{-1}. Weight module is independent on speed and the speed module is related to v^2. When speed increases, the engine module and driver cost will decrease and the speed module will increase exponentially.

For a given route, the load and the distance are given a fixed value while the travel speed on each arc is a decision variable. Fig. 3 shows how energy and total cost vary with speed. From the figure we can see a common character of mobile vehicles: driving too slow or too fast both lead to an increase in energy consumption and extra cost. As a balance between driver cost and energy cost, there is one optimal speed value, which minimizes the total cost.

2.3. An Integer Linear Programming Formulation for TDEVRP

In this section, we present an ILP model to formulate the TDEVRP problem. As the objective function (Eq. (3)) is non-linear to the speed value, we discretize the speed value into k levels as in Bektas and Laporte (2011). Let $R = \{1, 2, \dots, k\}$ denote the k speed levels and we have $v_c = v^1 < v^2 < \dots < v^k = v_f$. We also define v^{mr}as the vehicle speed in time interval m with a speed level $r, r \in R$, which indicates $v^{1r}, v^{2r}, v^{5r} = v_c$ (we note that the optimal speed for the congestion period is always v_c). Let M be the set of time intervals and $(b^{m-1}, b^m]$ denote the $m-th$ interval as shown in Fig. 2, where $M = \{1, 2, 3, 4, 5\}$. Then we have $b^0 = 0,\ b^1 = T_1 - d/v^2,\ b^2 = T_1,\ b^3 = T_2 - d/v^4,\ b^4 = T_2,\ b^5 = u_0$. Time intervals 1, 3 and 5 have a constant speed while the speed of intervals 2 and 4 changes at T_1 and T_2.

The model uses the following decision variables:

(1) x_{ij}: is a binary variable which equals 1 if a vehicle visit node j after i, 0 otherwise.
(2) z_{ij}^{mr}: is a binary variable which equals 1 if a vehicle visits node j after i, leaving node i during the m-th time interval with free-flow speed v^r.
(3) f_{ij}: indicates the total amount of commodity flowing from node i to node j.
(4) t_{ij}^{mr}: is a variable that equals the time at which a vehicle leaves node i during the m-th period to node j with speed v^{mr}.
(5) w_i: indicates the time instant at which the service at node i starts.
(6) y_i: indicates the energy level of a vehicle at node i.

To put it simply, we rewrite the travel time from node i to node j which starts at t_{ij}^{mr} with speed v^{mr} as $T(t_{ij}^{mr}, v^{mr}, i, j)$ and the corresponding energy consumption as $E(t_{ij}^{mr}, v^{mr}, i, j)$, shown in Eqs. (4) and (5).

Given these definitions, we present the ILP model of the TDEVRP as follows:

Minimize:

$$\sum_{(i,j)\in\mathcal{A}}\sum_{r\in R}\sum_{m\in M}\lambda E(t_{ij}^{mr}, v^{mr}, i, j) \tag{6}$$

$$+\sum_{i\in\mathcal{C}\cup\mathcal{F}'}\delta\left(\sum_{j\in\mathcal{N}}\sum_{m\in M}\sum_{r\in R}t_{ij}^{mr} - \sum_{j\in\mathcal{N}}\sum_{m\in M}\sum_{r\in R}t_{ji}^{mr}\right) \tag{7}$$

$$+\sum_{i\in N_0'}\delta\left(w_i - \sum_{j\in N_0'}\sum_{m\in M}\sum_{r\in R}t_{ji}^{mr}\right) \tag{8}$$

$$+\sum_{j\in\mathcal{N}}hx_{0j} \tag{9}$$

subject to:

$$\sum_{j\in\mathcal{N}}x_{0j} \leqslant K \tag{10}$$

$$\sum_{i\in\mathcal{N}}x_{ij} = 1, \quad \forall j\in\mathcal{C} \tag{11}$$

$$\sum_{i\in\mathcal{N}}x_{ij} \leqslant 1, \quad \forall j\in\mathcal{F}' \tag{12}$$

$$\sum_{i\in\mathcal{N}}x_{ji} - \sum_{i\in\mathcal{N}}x_{ij} = 0, \quad \forall j\in\mathcal{N} \tag{13}$$

$$\sum_{j\in\mathcal{N}}f_{ji} - \sum_{j\in\mathcal{N}}f_{ij} = q_i, \quad \forall i\in\mathcal{C} \tag{14}$$

$$(q_j+\mu)x_{ij} \leqslant f_{ij} \leqslant x_{ij}(\mu+Q-q_i), \quad \forall (i,j)\in\mathcal{A} \tag{15}$$

$$\sum_{i\in\mathcal{N}}\sum_{m\in M}\sum_{r\in R}(t_{ij}^{mr} + T(t_{ij}^{mr}, v^{mr}, i, j)) \leqslant w_j, \quad \forall j\in\mathcal{C} \tag{16}$$

$$\sum_{j\in\mathcal{N}}\sum_{m\in M}\sum_{r\in R}w_i + p_i \leqslant t_{ij}^{mr}, \quad \forall i\in\mathcal{C} \tag{17}$$

$$l_i \leqslant w_i \leqslant u_i, \quad \forall i\in\mathcal{C} \tag{18}$$

$$0 \leqslant y_j \leqslant y_i - E(t_{ij}^{mr}, v^{mr}, i, j) + Y(z_{ij}^{mr} - 1), \quad \forall r\in R,\ m\in M,\ i\in\mathcal{C},\ j\in\mathcal{N} \tag{19}$$

$$0 \leqslant y_j \leqslant Y - E(t_{ij}^{mr}, v^{mr}, i, j), \quad \forall r\in R, m\in M, i\in\mathcal{F}', j\in\mathcal{N} \tag{20}$$

$$y_0 = Y \tag{21}$$

$$z_{ij}^{mr}b_{ij}^{m-1} \leqslant t_{ij}^{mr} \leqslant z_{ij}^{mr}b_{ij}^{m}, \quad \forall (i,j)\in\mathcal{A},\ m\in M,\ r\in R \tag{22}$$

$$\sum_{r\in R}\sum_{m\in M}z_{ij}^{mr} = x_{ij}, \quad \forall (i,j)\in\mathcal{A} \tag{23}$$

$$z_{ij}^{mr} \in \{0,1\}, \quad \forall (i,j)\in\mathcal{A},\ m\in M,\ r\in R \tag{24}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j)\in\mathcal{A} \tag{25}$$

The goal of the TDEVRP problem is to minimize a cost function, which consists of three parts: the energy cost, drivers' wages, and a fixed cost. Constraint (10) limits the total number of vehicles used. Constraint (11) ensures that every customer is visited exactly once and constraint (12) indicates a station may not be visited in any route. Constraint (13) guarantees the integrity of routes. Constraint (14) models the commodity flow on each arc and constraint 15 ensures the capacity constraint is satisfied.

Time relations are ensured by constraints (16)–(18). Constraints (16) and (17) model the travel time and service time. Constraint (18) ensures the time window constraint is satisfied. The time-dependent travel time is modeled in $T(t_{ij}^{mr}, v^{mr}, i, j)$ introduced in Eq. (4). This variable models the travel time on an arc as a linear function of the departure time and simplify the ILP representation of the problem.

Energy consumptions are restricted by constraints (19) and (20), which confine the energy consumption driving from node i to node j, differing in whether node i is a customer or a station. Constraint (21) requires that vehicles are fully charged when leaving depot. The time-dependent energy consumption is modeled in $E(t_{ij}^{mr}, v^{mr}, i, j)$ introduced in Eq. (5). Along with $T(t_{ij}^{mr}, v^{mr}, i, j)$, these two variables illustrate the new features of TDEVRP.

Constraints (22) and (23) define the relation between decision variables. Constraints (24) and (25) confine the values of the decision variables.

Using the above ILP model, the preliminary experimental results showed that CPLEX is able to solve instances with very limited size(less than 15 customers) given a time limit of 5 h. It is unsurprising since the TDEVRP is a very challenging NP-hard problem. To solve large-sized real-life problems, we resort to a heuristic method that is presented in the next section.

3. An Variable Neighbourhood Search Algorithm for TDEVRP

3.1. Main scheme

In this section, we propose an iterated variable neighbourhood search (IVNS) algorithm to solve the TDEVRP, as shown in Algorithm 1. The algorithm follows the iterated local search (ILS) (Lourenco, Martin, & Stutzle, 2010) framework and is composed of four main search components.

1. an initial solution construction procedure to generate a starting solution of the algorithm (Section 3.2). The saving heuristic proposed by Solomon (1987) is adopted to produce a solution with a fleet of K (the given max vehicle number) vehicles.
2. a variable neighbourhood descent procedure (VND) (Section 3.3) which focus on optimizing the node sequences and reducing the length of routes by assuming a fixed free-flow speed and zero post-service delay. Four dedicated neighbourhoods (induced by the move operators Insert, Swap, Exchange, Reverse, see Section 3.3.1) is adopted within an iterated improvement procedure. Violations for time window constraints and energy constraints are allowed during the VND procedure. When no improvement can be found in any one of the neighbourhoods, a labeling procedure proposed by Hiermann et al. (2016) is used to reoptimize the positions of recharging stations within the routes and the following make-feasible procedure tries to bring the solution back to the feasible region.
3. a departure time and speed optimization procedure (Section 3.4) which efficiently optimizes the speed and departure time on each arc based on the concatenation operators we propose in Section 4.
4. a perturbation phase accepting non-improving moves to diversify the search (Section 3.5) after each round of optimization.

IVNS starts with an initial solution generated by the procedure given in Section 3.2. It then repeats a number of ‘while’ loops. At each loop, IVNS first applies the VND procedure to optimize the node sequence and reduce the length of routes. Then a DSOP procedure follows to optimize speed and departure time on each arc. After the DSOP, one loop of search ends and the best solution found so far is updated if a better feasible solution is found. At the beginning of next loop, a perturbation procedure is adopted to help the search to escape from local optimum and displace the search process to a distant zone of the search space (the VND will finally reach a local optimum when it terminates, therefore the perturbation procedure is always required). The search continues until a max loop limit π is reached. If no better solution is found within τ rounds, the search rolls back to the best feasible solution found.

Algorithm 1. Iterated Variable Neighbourhood Search for the TDEVRP

```
Input: Initial solution $s$
Output Best feasible solution found $s_f^*$
1: $s^* \leftarrow s, i \leftarrow 0, i' \leftarrow 0$
2: while $i \leqslant \pi$ do
3:   if $i > 0$ then
4:     $s' \leftarrow Perturbation(s)$
5:   else
6:     $s' \leftarrow s$
7:   end if
8:   $s' \leftarrow VariableNeighbourhoodDescent(s')$
9:   $s' \leftarrow DSOP(s')$
10:    if $eva(s') < eva(s^*)$ then
11:      $s^* \leftarrow s'$
12:    end if
13:    if $IsFeasible(s')$ and $eva(s_f^*) > eva(s')$ then
14:      $s_f^* \leftarrow s', i' \leftarrow 0$
15:    else
16:      $i' \leftarrow i' + 1$
17:    end if
18:    if $i' = \tau$ then
19:      $s \leftarrow s^*, i' \leftarrow 0$
20:    else
21:      $s \leftarrow s'$
22:    end if
23:    $i \leftarrow i + 1$
24: end while
25: return $s_f^*$
```

3.2. Initial solution construction

We use the saving heuristic proposed by Solomon (1987) to construct the initial solution. It starts with a solution in which each customer is served by a dedicated vehicle. At each step, the algorithm evaluates the concatenations of all pairs of routes and performs the best concatenation to reduce the number of vehicles. This procedure continues until no improvement can be found. Note that infeasibility is allowed during the construction for energy and time window constraints, and the violation is evaluated based on Eq. (37). Thus the initial solution is not necessarily feasible.

As the saving heuristic is not able to insert charging stations into the routes, a labeling algorithm proposed in Hiermann et al. (2016) is adopted to add charging stations into the routes constructed by the saving heuristic. This algorithm is also used after each run of VND.

We note that Mancini (2017) proposed a Multi Start Constructive Heuristic (MSCH) specially designed for TDVRP. We tested the algorithm on a set of representative instances. The results indicate that the MSCH algorithm shows no difference with the Clark and Wright (CW) in the aspect of final solution quality. However, the MSCH requires the IVNS to run more iterations to reach the best solution and will induce more algorithm running time.

This is because the route number in solutions produced by CW is very close to the final solution while MSCH produces solutions with too many routes. Additional iterations are required to reduce the number of routes in the solution. Therefore we conclude that CW is more suitable for TDEVRP. In addition, the CW is a classic and well-known heuristic which is still very popular in recent

VRP literature (e.g., Koc, Bektas, Jabali, & Laporte (2014), Franceschetti et al. (2016)). That is the reason why the CW heuristic is used in this paper.

3.3. Variable Neighbourhood Descent procedure

The proposed IVNS algorithm includes a Variable Neighbourhood Descent procedure which optimizes a given solution (a constructed initial solution or a perturbed solution) to a local optimum. The pseudo-code of the VND procedure is shown in Algorithm 2.

The VND procedure employs four neighbourhoods (that rely on four move operators, namely Insert, Swap, Exchange and Reverse whose definitions are introduced in Section 3.3.1) in a cyclic manner. Specifically, VND switches to the next neighbourhood once no improvement can be found in the current neighbourhood, and it terminates when no improvement can be found in any of the four neighbourhoods. A subsequent labeling procedure is responsible for the optimization of the placement of the charging stations (within the optimized routes). Since the VND procedure allows search in infeasible space, a make-feasible procedure (Section 3.3.3) is required to ensure solution feasibility. One prerequisite of the VND procedure is to assume a fixed free-flow speed and null post-service delay for this is the necessary condition to use the efficient evaluation method (see Section 4).

Algorithm 2. Variable Neighbourhood Decent

```
Input: Current solution $s$
Output: Best solution found $s^*$
1: $s^* \leftarrow s$
2: while true do
3:    $s' \leftarrow s^*$
4:    $s^* \leftarrow$ Decent in Insert Neighbourhood$(s^*)$
5:    $s^* \leftarrow$ Decent in Swap Neighbourhood$(s^*)$
6:    $s^* \leftarrow$ Decent in Exchange Neighbourhood$(s^*)$
7:    $s^* \leftarrow$ Decent in Reverse Neighbourhood$(s^*)$
8:    if $s' = s^*$ then
9:       break
10:   end if
11: end while
12: $s^* \leftarrow$ Labeling$(s^*)$
13: $s^* \leftarrow$ MakeFeasible$(s^*)$
14: return $s^*$
```

3.3.1. Neighbourhoods

The proposed VND procedure and labeling procedure rely on six move operators that are commonly used for other VRP variants (Lin et al., 2014). We recall the definitions of these move operators below with an illustration in Fig. 4 where a cycle represents a normal node and a triangle indicates a charging station:

1. *Insert* (Fig. 4(a)) Select a node and insert it into a proper location of another route. This operator introduces three new arcs and requires three concatenations to evaluate the move.
2. *Swap* (Fig. 4(b)) Select two nodes from different routes and exchange their positions. This operator introduces four new arcs and requires four concatenations to evaluate the move.
3. *Exchange* (Fig. 4(c)) Exchange two subsequences from two different routes. Choose two proper positions in two routes and exchange the subsequences after the positions. This operator introduces two new arcs and requires two concatenations to evaluate the move.
4. *Reverse* (Fig. 4(d)) Select a proper position in a route and invert the sequence of adjacent nodes. Note that the complexity of evaluating a Reverse move is directly related to the total number of nodes reversed. For efficiency reason only two nodes are allowed to be inverted in this work.
5. *InsertRemoveStation* (Fig. 4(e)) Insert a charging station into a route or remove a charging station from a route. This move is used in the labeling algorithm which optimizes the charging stations in a route.

The well-known nearest-neighbor heuristic is used to accelerate the search process by reducing the explored neighborhoods. Specifically a candidate list is constructed for each node containing the geographically nearest ω neighboring nodes. The value of ω should be appropriately decided since a large ω typically requires more running time while a small ω restricts the search in a limited region (which may reduce search diversity). Typically, a suitable ω value is determined through computational experiments. Before the examination of each neighbourhood, all nodes are randomly shuffled, which usually helps to improve the performance of the VND procedure.

3.3.2. Labeling

In this work we also adopt the labeling algorithm proposed by Hiermann et al. (2016) to optimize the location of charging stations in routes. It is designed for electric vehicle routing problems but can be easily adapted to TDEVRP with the concatenation operators we propose (see Section 4). Provided with a set of available charging stations, the labeling algorithm can provide with the optimal recharging plan for a given sequence of customer nodes. Note that the labeling algorithm do not guarantee the feasibility of energy constraints for there are routes with no feasible recharging plans. In this case, the VND turns to a make-feasible procedure to handle the violated constraints.

3.3.3. Make-feasible

A make-feasible procedure is adopted in case that the VND procedure provides an infeasible solution. It is conducted as follows. The penalty parameters in Eq. (37) are multiplied by ζ, and the VND procedure restarts with the infeasible solution as its input. In case the resulting solution remains infeasible, the penalty parameter is further multiplied by ζ, and such a multiplication is allowed at most three times. If no feasible solution is reached after three turns, the algorithm rolls back to the best feasible solution reached in the search.

3.4. Departure time and speed optimization procedure

The VND procedure only optimizes the route sequence. We now focus on the departure time and speed optimization of each arc of the resulting solution. Franceschetti et al. (2013) introduced an algorithm to optimize the departure time and speed of a route by eliminating all the time windows at customer nodes and regarding the whole sequence as a single-arc route (SINGLE-ARC-DSOP). In case of time window violation, the algorithm restarts on subsequences divided by the node with time window violation. The algorithm may have to restart for multiple times to find a feasible solution, and there is no guarantee that a feasible solution can be found.

Based on the proposed efficient evaluation method for the concatenation operation presented in Section 4, we propose a DSOP procedure which guarantees the feasibility of time window constraint and optimizes the speed and departure time on each arc efficiently.

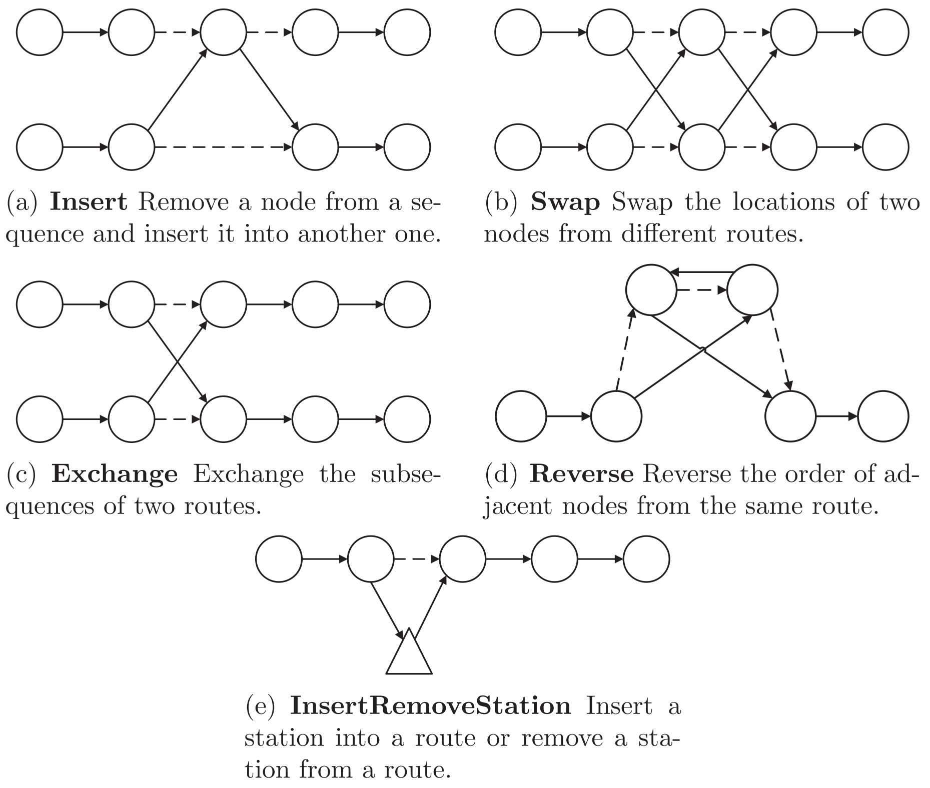
(a) **Insert** Remove a node from a sequence and insert it into another one.
(b) **Swap** Swap the locations of two nodes from different routes.
(c) **Exchange** Exchange the subsequences of two routes.
(d) **Reverse** Reverse the order of adjacent nodes from the same route.
(e) **InsertRemoveStation** Insert a station into a route or remove a station from a route.

Fig. 4. The move operators used in VND. Solid lines represent the arcs added by the move and dashed lines are the removed arcs.

Algorithm 3 DSOP

```
Input: A route $\sigma$
Output: Departure time $t_i$ and Speed $v_i$ for each node in $\sigma, i \in \{1, \dots, n-1\}$
1: initialize intervariables with max free flow speed $v_f$
2: if $TW(\sigma) \geqslant 0, \forall t_0 \in S(TW(\sigma))$ then
3:     the route is infeasible.
4: end if
5: for $i \in \{1, \cdots, n-1\}$ do
6:     if $i = 1$ then
7:         $l'_i \leftarrow 0$
8:     else
9:         $l'_i \leftarrow t_{i-1} + \frac{d_{i-1,i}}{v_{i-1}}$
10:    end if
11:    $pre \leftarrow$ dummy node of $n_i$ with new time window $[l'_i, LS(n_i)]$.
12:    $post \leftarrow$ dummy node of $n_{i+1}$ with new time window $[l_{i+1}, LS(n_{i+1})]$.
13:    $t_i, v_i \leftarrow$ SINGLE-ARC-DSOP$(pre, post)$
14: end for
```

The DSOP procedure (see Algorithm 3) first initializes all the inter variables for each customer with the max free-flow speed v_f. It is justified by the fact that if a route violates time window constraint with a max free-flow speed, it is also infeasible for any speed lower than v_f. First, we can ensure the feasibility of the time window constraint by the inter variable $TW(\sigma)$. Then we optimize the speed and departure time on each arc progressively. For each arc, l'_i is the earliest start from n_i and $LS(n_i)$ is the latest start ensuring the feasibility of the sequence after n_i. Then DSOP uses two dummy nodes with new time windows covering the propagation of time window constraint to obtain a feasible solution, while SINGLE-ARC-DSOP is used to calculate the optimal departure time and speed on a single-arc route (Franceschetti et al., 2013).

3.5. Perturbation

To diversify the search, each time the IVNS algorithm is trapped in a local optimum, it restarts its search with a new starting solution generated by the perturbation procedure. The key idea of the perturbation procedure is to accept non-improvement moves which may deteriorate the solution and thus lead the search to a distant zone of the search space. For a solution s and a perturbation move which creates a neighboring solution s', the new solution s' is accepted if (i) s' results in an evaluation variance no less than ξ ($\xi(eva(s) - eva(s') \geqslant \xi)$ where ξ is a negative value, (ii) s' reduces the total cost $(cost(s) - cost(s') \geqslant 0)$. The perturbation is applied to the local optimum solution provided by the VND procedure.

4. An efficient evaluation method for concatenation operation

In the context of vehicle routing problems, local search move operators usually first break the routes of the current solution into pieces and then apply a constant number of concatenation operations to recombine the pieces into a new solution. The evaluation of this new solution from scratch is very time-consuming. We note that this new solution is slightly different from the previous solution. If the new solution can be evaluated incrementally, it would significantly accelerate the local search process. For this purpose, we develop an efficient evaluation method for the concatenation operation.

Let σ_1 and σ_2 denote two sequences of nodes, each of which is associated with a number of feature values (listed in Table 3 in the context of EVRPTW). Concatenating these two sequences results in a new sequence $\sigma_1 \oplus \sigma_2$ and the associated feature values of the

Table 3
Features associated with a route. Storage of these feature values allows efficient evaluation for a route of EVRPTW.

Feature	Explanation
Load	the total load on the route
Dist	the total distance traveled
Dur	the duration of traveling across the route
TW	the time window violation of the route
ES	the earliest start time from depot
LS	the latest start time from depot
e	a boolean value indicating whether the route contains a charging station
EF	the energy needed to reach the first charging station
EL	the energy needed from the last charging station to the end of the route
EV	the energy violation of the route

concatenated sequence can be calculated efficiently using those of its composing subsequences. This is based on the fact that a solution resulting from a bounded number of arc exchanges on another solution can be evaluated efficiently (in $O(1)$) based on proper data structures characterizing the original solution (Vidal, Crainic, Gendreau, & Prins, 2015).

There are many efficient methods for evaluating concatenation operations for other VRP variants (Schneider et al., 2014; Vidal et al., 2013). However, in the context of TDVRP, no such evaluation method is available, which limits the application of local search on these problems. The first reason is that the travel time and energy consumption between two nodes are dependent on the departure time and travel speed. Different departure time and speed lead to different travel time and energy consumption, and this information cannot be represented by a single value. Storing and calculating all these values corresponding to each departure time is computationally too expensive, and it requires too much memory. Another reason is that the duration of σ_1 varies with departure time, which in turn influences the start time of σ_2. The propagation effect of the departure time dependency makes the evaluation rather complicated.

In this section, we introduce an efficient method that can overcome the difficulties mentioned above, and realizes the evaluation of the concatenation operation for TDEVRP in $O(1)$. We first introduce in Section 4.1 the analytical results that are useful for presenting the efficient calculation method of the feature values of the concatenated sequence based on the values of the composing subsequences in Section 4.2. With the feature values, one can compute the cost of a concatenated sequence (corresponding to a route) based on the proposed cost function. We note that different departure time of the vehicle at the head of the route (concatenated sequence) lead to different route costs, and we aim to identify the one leading to the minimal cost. In Section 4.3, we present a method that achieves fast identification of such an optimal departure time.

4.1. Analytical results for the concatenation operation

As shown in Fig. 2, the travel time of a fixed distance is dependent on the departure time for a given free-flow speed. Additionally, we discover that given a fixed free-flow speed and no post-service delay, the linear correlation with departure time also applies to total duration, energy consumption, and other feature values characterizing the routes.

Theorem 1. *For a TDEVRP instance with a fixed free-flow speed and no post-service delay, the duration, and energy consumption of a route is linearly dependent on the departure time. The total cost is a continuous piecewise linear function of departure time.*

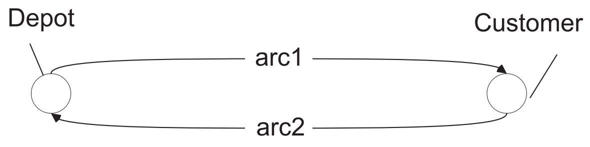

Fig. 5. A simple example with one customer.

Proof. Consider a depot-customer-depot route visiting only one customer, shown in Fig. 5, the duration of the route consists of: (i) the travel time on the first arc, (ii) the travel time on the second arc, (iii) service time at customer node (iv) waiting time before service in case of early arrival.

Let t_s denote the departure time from the depot. The travel time on the first arc is linearly dependent on t_s. The waiting time at the customer node and departure time from the customer node, which is linearly related to the travel time, are also linearly dependent on t_s. Since the travel time of the second arc is linearly dependent on the departure time from the customer node, it is linearly dependent on t_s as well. The linear correlation with t_s is propagated because the travel time on each arc is linearly dependent on the start of the arc. Further, the duration of the route, which is the sum of several components linearly dependent on t_s, is a continuous piecewise linear function of t_s. Similarly, one can easily prove that the energy consumption and other feature values in Table 3 are also a continuous linear piecewise function of t_s. □

According to Franceschetti et al. (2013) and Bektas & Laporte (2011), the optimal speed of arcs in free-flow period falls in interval $[v_e, v_f]$. Fixing the free-flow speed to a low value (e.g., v_e) may restrict the search space in a small region, therefore, excludes high-quality solutions, while fixing the value to a high speed (e.g., v_f) results in high cost for each arc and therefore increases the total cost. Therefore the free-flow speed should be selected properly according to the problems at hand (essentially affected by the cost function curve in Fig. 3).

4.2. An efficient feature value calculation method for the concatenation operation

For a concatenation operation $\sigma_1 \oplus \sigma_2$, we use footnotes to represent the associated route of a feature value. Featues with footNote 1 are the features of σ_1. FootNote 2 and 12 denotes the feature values of σ_2 and $\sigma_1 \oplus \sigma_2$. In the proposed method we extend the feature values used in EVRPTW in the context of departure time For example, $Dur_1(t_s)$ denotes the duration of sequence σ_1 which starts at t_s and $Dur_{12}(t_s)$ denotes the duration of $\sigma_1 \oplus \sigma_2$ which starts at t_s. For simplicity the definition of these features are referred to the original work (Vidal et al., 2015). The value t_s indicates the departure time from the start of σ_1 (starting depot) and t'_s indicates the departure time from the start of σ_2 (customer node or charging station). Note that a list instead of a number is needed to store the information of some features as the feature values vary with the start time of the sequence. Those features irrelevant to departure time such as distance and earliest start are denoted as $Dist_{12}$ and ES_{12}.

In addition to the features in Table 3, new features are required to model the propagation of linear correlation with departure time. For a concatenation σ_1 to σ_2, let $FT(t_s)$ denote the finish time of the first sequence σ_1 with departure time t_s. Let $RT(t_s)$ denote the reach time at σ_2 when σ_1 starts at t_s. By definition $FT(t_s) = t_s + Dur_1(t_s)$, $RT(t_s) = FT(t_s) + T(FT(t_s), v', i, j)$, where node i is the end of σ_1, j is the start of σ_2 and v' is the chosen free-flow speed. Also for simplicity we denote the travel time on arc (i,j) $T(FT(t_s), v', i, j)$ as $T_{ij}(t_s)$ and the energy consumption $E(FT(t_s), v', i, j)$ as $E_{ij}(t_s)$.

Table 4
Features associated with a route. Storage of these feature values allows efficient evaluation for a route of TDEVRP.

Feature	Explanation
$Load_1,\ Load_2,\ Load_{12}$	the total load for route $\sigma_1,\ \sigma_2$ and $\sigma_1 \oplus \sigma_2$
$Dist_1,\ Dist_2,\ Dist_{12}$	the distance traveled
$t_s,\ t_s'$	departure time from the starting of $\sigma_1,\ \sigma_2$
d_{ij}	distance between σ_1 and σ_2
$Dur_1(t_s),\ Dur_2(t_s'),\ Dur_{12}(t_s)$	the duration of traveling across the route if departure from starting depot at t_s
$TW_1(t_s),\ TW_2(t_s'),\ TW_{12}(t_s)$	the time window violation
$FT(t_s)$	the finish time of the first sequence σ_1
$RT(t_s)$	the arrival time at σ_2
$T_{ij}(t_s),\ E_{ij}(t_s)$	the time and energy consumed from the last node of σ_1 to the start of σ_2
$ES_1,\ ES_2,\ ES_{12}$	the earliest start time from depot
$LS_1,\ LS_2,\ LS_{12}$	the latest start time from depot
$e_1,\ e_2,\ e_{12}$	a boolean value indicating whether the route contains a charging station
$EF_1(t_s),\ EF_2(t_s'),\ EF_{12}(t_s)$	the energy needed to reach the first charging station
$EN_1(t_s),\ EN_2(t_s'),\ EN_{12}(t_s)$	the energy consumed by traversing the route
$EL_1(t_s),\ EL_2(t_s'),\ EL_{12}(t_s)$	the energy needed from the last charging station to the end of the route
$EV_1(t_s),\ EV_2(t_s'),\ EV_{12}(t_s)$	the energy violation of the route

For clarity purposes, we list all feature values of TDEVRP in Table 4.

4.2.1. Feature value calculation

With these definitions, we present the concatenation of $\sigma_1 \oplus \sigma_2$ and the features characterizing the routes as follows:

Load and distance. The load and distance are calculated as follows:

$$Load_{12} = Load_1 + Load_2 \tag{26}$$

$$Dist_{12} = Dist_1 + d_{ij} + Dist_2 \tag{27}$$

where node i is the end node of σ_1 and node j is the start node of σ_2. The load of $\sigma_1 \oplus \sigma_2$ is simply the sum of load of σ_1 and σ_2. Also, the distance of $\sigma_1 \oplus \sigma_2$ is the total distance of the two composing sequences plus the distance from the end node of σ_1 to the start node of σ_2. The σ_1 and σ_2 are two partial routes and may not include the starting or ending depot. Therefore $Dist_1$ do not include the distance to the ending depot and $Dist_2$ do not include the distance from the starting depot.

Time features. Let $RT^{-1}()$ denote the inverse function of $RT()$ (e.g., the $RT^{-1}(ES_2)$ equals to the departure time from depot which makes the arrival time at σ_2 equal to ES_2). We first calculate ES_{12} and LS_{12} to determine the time interval in which the optimal departure time falls:

$$ES_{12} = \begin{cases} ES_1, & ES_2 \leqslant RT(ES_1) \\ LS_1, & RT(LS_1) \leqslant ES_2 \\ RT^{-1}(ES_2), & else \end{cases} \tag{28}$$

$$LS_{12} = \begin{cases} ES_1, & LS_2 \leqslant RT(ES_1) \\ LS_1, & RT(LS_1) \leqslant LS_2 \\ RT^{-1}(LS_2), & else \end{cases} \tag{29}$$

ES_{12} and LS_{12} are calculated based on the first-in-first-out property. By definition of $ES(\sigma)$ and $LS(\sigma)$ (Vidal et al., 2013), $[ES_{12}, LS_{12}]$ is reduced from $[ES_1, LS_1]$. The interval $[ES_1, LS_1]$ is reduced if extra waiting time or time window violation in σ_2 is induced when σ_1 is placed before σ_2.

If the earliest reach time does not lead to extra waiting time ($ES_2 \leqslant RT(ES_1)$), ES_{12} equals ES_1 and the interval is not changed. If the latest reach time leads to extra waiting at σ_2 ($RT(LS_1) \leqslant ES_2$), ES_{12} equals LS_1 and the interval is reduced to a number. If starting at the earliest time leads to extra waiting time while starting at the latest time does not, ES_{12} is the earliest departure time at which exactly no extra waiting time is induced. Let RT^{-1} be the inverse function of RT, and in this case ES_{12} is then $RT^{-1}(ES_2)$. LS_{12} can be calculated in a similar way. If choosing any departure time within $[ES_1, LS_1]$ does not lead to extra violation ($RT(LS_1) \leqslant LS_2$), LS_{12} equals LS_1. If the earliest start results in violation at σ_2 ($LS_2 \leqslant RT(ES_1)$), the interval is reduced to a number and LS_{12} equals ES_1. For intermediate situations LS_{12} equals to $RT^{-1}(LS_2)$.

For each $t_s \in [ES_{12}, LS_{12}]$, the duration $Dur_{12}(t_s)$ and time window violation $TW_{12}(t_s)$ are calculated as:

$$Dur_{12}(t_s) = Dur_1(t_s) + Dur_2(t_s') + T_{ij}(t_s) + \Delta WT \tag{30}$$

$$TW_{12}(t_s) = TW_1(t_s) + TW_2(t_s') + \Delta TW \tag{31}$$

where $t_s' = RT(t_s) + \Delta WT - \Delta TW$, $\Delta WT = max(ES_2 - RT(t_s) + TW_1(t_s), 0)$ and $\Delta TW = max(RT(t_s) - TW_1(t_s) - LS_2, 0)$. t_s' represents the actual start at σ_2 after concatenation. As in the original definition, ΔWT and ΔTW are the increments of the waiting time and time window violations. Note that when calculating t_s' a ΔTW is subtracted to repair the time window violation (Nagata et al., 2010).

State of charge. Similar to Schneider et al. (2014), we use the following features to characterize a route: (i) e_{12}, a boolean value indicating whether route $\sigma_1 \oplus \sigma_2$ contains a charging station or not, (ii) $EF_{12}(t_s)$, a list storing the amount of energy required to travel from start of the route to the first charging station, (iii) $EL_{12}(t_s)$, a list storing the energy required to travel from the last station to the end of the route, (iv) $EV_{12}(t_s)$ indicating the energy violation of route $\sigma_1 \oplus \sigma_2$ with start at t_s. Note that when a route contains no charging station, the corresponding $EF_{12}(t_s)$ and $EL_{12}(t_s)$ are the total energy needed to traverse the whole route. An additional feature $EN_{12}(t_s)$ is also used to indicate the total energy consumed.

$$e_{12} = e_1 \bigvee e_2 \tag{32}$$

$$EN_{12}(t_s) = EN_1(t_s) + \Delta EN_1 + E_{ij}(t_s) + EN_2(t_s') \tag{33}$$

$$EF_{12}(t_s) = \begin{cases} EF_1(t_s) + \Delta EF_1, & if\ e_1 \\ EF_1(t_s) + \Delta EF_1 + E_{ij}(t_s) + EF_2(t_s') - \Delta EV, & else \end{cases} \tag{34}$$

$$EL_{12}(t_s) = \begin{cases} EL_2(t_s'), & if\ e_2 \\ EL_1(t_s) + \Delta EL_1 + E_{ij}(t_s) + EL_2(t_s') - \Delta EV, & else \end{cases} \tag{35}$$

$$EV_{12}(t_s) = EV_1(t_s) + \Delta EV_1 + EV_2(t_s') + \Delta EV \tag{36}$$

where $t_s' = RT(t_s) + \Delta WT + \Delta TW$, $\Delta EV = max(EL_1(t_s) + \Delta EL_1 + E_{ij}(t_s) + EF_2(t_s') - Y, 0)$. $\Delta EN_1,\ \Delta EF_1,\ \Delta EL_1,\ \Delta EV_1$ denote the change in the feature values of σ_1 in weight module due to the concatenation of σ_2.

For a concatenation $\sigma_1 \oplus \sigma_2$, the calculation of feature values characterizing the new route follows three steps: (i) calculate $FT(t_s)$ and $RT(t_s)$, (ii) calculate feature values irrelevant to $t_s : Load_{12}, Dist_{12}, ES_{12}$ and LS_{12}, (iii) calculate the rest of feature values for each $t_s \in [ES_{12}, LS_{12}]$.

4.2.2. Index set calculation

We define the critical point set of a linear piecewise function where the gradient changes (e.g. $0, T_1 - \frac{d}{v_c}, T_1, T_2 - \frac{d}{v_f}, T_2, u_0$ in Fig. 2) as the index set of the function. Recall that the features associated with a route are either irrelevant to or (piecewise) linearly dependent on t_s. Therefore to calculate the feature values corresponding to each departure time, it is unnecessary to calculate every time instant in $[ES_{12}, LS_{12}]$. Instead, it suffices to examine the critical points of the index set.

The calculation of the index set for each feature is done in a summation manner (see Eqs. (30), (31), (34)–(36)). The index set of features on the left hand side is the union of the index sets on the right hand side. For example, let $S(var)$ denote the critical point set of a feature var, then for $Dur_{12}(t_s)$ we have $S(Dur_{12}(t_s)) = S(Dur_1(t_s)) \cup S(Dur_2(t'_s)) \cup S(T_{12}(t_s)) \cup S(\Delta WT)$, in which $S(Dur_2(t'_s))$ and $S(\Delta WT)$ should be calculated accordingly. Note that those feature values in $S(Dur_{12}(t_s))$ exceeding the interval $[ES_{12}, LS_{12}]$ are removed.

4.2.3. Complexity

By definition the size of the index set for each feature is determined by the time structure of a route, namely, the number of critical points for each feature of a route σ is $O(size(\sigma))$. So the complexity for calculating feature values with respect to each t_s is $O(1)$.

As a result, to calculate and store the feature values characterizing a route σ, the complexity is $O(size(\sigma))$. Therefore, updating the feature values after a move can be performed in $O(size(\sigma)^2)$, given that the feature values are stored on every node to characterize the subsequences.

4.3. An efficient method for fast identification of the optimal departure time

The algorithm proposed in this work allows infeasibility of a candidate solution. For a concatenation operation $\sigma_1 \oplus \sigma_2$ the evaluation function for start time t_s is:

$$\begin{aligned} eva(\sigma_1 \oplus \sigma_2) = \lambda EN_{12}(t_s) + \delta Dur_{12}(t_s) + h + \rho_e EV_{12}(t_s) \\ + \rho_c max(Load_{12} - Q, 0) + \rho_t TW_{12}(t_s) \end{aligned} \tag{37}$$

The first three terms of Eq. (37) are the costs of the trip and the last three terms are penalty values for constraint violations where $\rho_c,\ \rho_e,\ \rho_t$ are penalty weights for the constraint violations of vehicle capacity, load and time window.

Different departure times result in different costs for a route, and the one leading to the minimal cost is selected. According to Theorem 1, the cost function is a continuous linear piecewise function of the departure time. The optimal departure time t_s^* belongs necessarily to its critical point set. Therefore, to evaluate a route, we just need to calculate the feature values for each t_s^* in the index set. Usually, this operation requires $O(size(\sigma))$ time. However, as we show below, we can identify a more efficient way to calculate the minimal cost.

Theorem 2. *Given a fixed free-flow speed and no post-service delay, the optimal departure time is the one minimizing the duration of the route.*

Proof. Theorem 2 exploits the fact that the optimal trip is the one with the shortest duration.Indeed, given that the cost of a route being composed of driver cost, energy cost, and a fixed acquisition cost of the vehicle, the route with the shortest duration minimizes driver's cost. Recall that the earliest and the latest departure time of a route (see Section 4.2.1) will result in exactly the same waiting duration (Schneider et al., 2014). Then the time consumed at customer nodes and charging stations is fixed for a route. Therefore, minimal route duration indicates the minimal travel time. As the free-flow speed is fixed, this also indicates that more distance is traveled within a free-flow period and therefore minimal energy cost. Then we can conclude that the route with the shortest duration is the route with the minimal energy cost and therefore the minimal total cost. □

Lemma 1. *For a route σ in Theorem 2 with a departure time interval $[ES_\sigma, LS_\sigma]$, the optimal departure time t_s^* is given by:*

$$t_s^* = \begin{cases} ES_\sigma, & ES_\sigma > T_1 \\ LS_\sigma, & LS_\sigma < T_1, FT_\sigma(LS_\sigma) \leqslant T_2 \\ T_1, & ES_\sigma < T_1 < LS_\sigma, FT_\sigma(T_1) \leqslant T_2 \\ \arg\min_{t_s} Dur_\sigma(t_s), & else \end{cases} \tag{38}$$

Proof. When $ES_\sigma > T_1$, the travel starts in a free-flow period and may be followed by a congestion period. Based on the principle that the vehicle should travel as much distance as possible in free-flow period, starting at ES_σ results in the longest travel distance in free-flow speed and therefore the shortest duration of the route. Choosing a departure time later than ES_σ may postpone more trip into congestion period and $t_s^* = ES_\sigma$. Similarly, we can obtain the optimal departure time for a route if condition $LS_\sigma < T_1,\ FT_\sigma(LS_\sigma) \leqslant T_2$ or $ES_\sigma < T_1 < LS_\sigma,\ FT_\sigma(T_1) \leqslant T_2$ is satisfied. For other occasions there is no explicit conclusion and we need to calculate Dur_σ to determine the optimal departure time. However we can use an approximation of t_s^* to calculate the optimal departure time in $O(1)$.

Lemma 2. For a route σ in Lemma 1, we can use $t_s'^*$ to approximate the optimal value of the departure time. Denote the driving distance at free-flow period as $dist_f$, the distance at congestion period as $dist_c$, the energy cost of the weight module as C_w. Then the worst case deviation of the cost evaluation from the optimal value is $\frac{dist_c(c_c - c_f)}{(dist_c + dist_f)c_f + Cost_w}$:

$$t_s^* = \begin{cases} ES_\sigma, & ES_\sigma > T_1 \\ LS_\sigma, & LS_\sigma < T_1 \\ T_1, & ES_\sigma < T_1 < LS_\sigma \end{cases} \tag{39}$$

where c_c and c_f are the driving costs per unit distance with speed levels v_c and v_f including the energy cost of the engine module, the energy cost of the speed module and the driver cost (Eq. (3)).

Proof. Denote the total waiting and serving time of the route as T_s and the energy cost of the weight module as $Cost_w$. Then the cost of the route departure at $t_s'^*$ is $Cost(\sigma, t_s'^*) = \delta T_s + c_f dist_f + c_c dist_c + Cost_w$. By Lemma 1 we know that if $dist_c = 0, Cost(\sigma, t_s'^*) = Cost(\sigma, t_s^*)$. In the worst case, departure at t_s^* allows to travel an extra free-flow distance of $dist_c$. Fig. 6 shows an example of such a case.

In the worst case, we can calculate the cost with the optimal departure time $Cost(\sigma, t_s^*) = \delta T_s + c_f(dist_f + dist_c) + Cost_w$. Then we can calculate the worst case deviation from the optimal cost: $\frac{Cost(\sigma,t_s'^*) - Cost(\sigma,t_s^*)}{Cost(\sigma,t_s^*)} = \frac{dist_c(c_c - c_f)}{\delta T_s + c_f(dist_f + dist_c) + Cost_w} < \frac{dist_c(c_c - c_f)}{(dist_c + dist_f)c_f + Cost_w}$. This is estimated to be a very small value. In the problem setting introduced in Section 2, assume a total waiting time of 1 h and zero commodity transferred, the worst case deviation is only 5.8%.

Note that the result given by Eqs. (38) and (39) is not the only optimal value, and there are other possible optimal departure times.

With Lemma 2 the evaluation of a concatenation $\sigma_1 \oplus \sigma_2$ follows three steps: (i) calculate $RT(t_s),\ ES_{12}$ and LS_{12}. (ii) calculate the approximate optimal departure time $t_s'^*$ according to Eq. (39). (iii) calculate variables characterizing the route with departure time $t_s'^*$ and evaluate the route based on these variables. The complexity of the concatenation evaluation is therefore $O(1)$. Note that the concatenation operations works under the condition that the

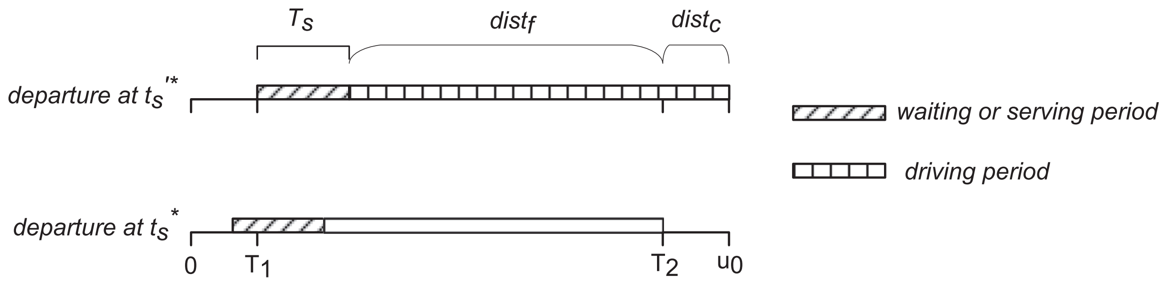

Fig. 6. A simple example of worst case evaluation with only one customer. In this case departure at t_s^* will drive the most extra distance ($dist_c$) within free-flow period compared with the case departure from $t_s'^* = T_1$. The distance from the depot and the node after the depot equals or close to 0.

free-flow speed is fixed and there is no post-service delay. This is why our VND procedure also works under the same condition.

5. Computational results

To evaluate the efficiency of the proposed IVNS algorithm, we carry out extensive experiments on a set of TDEVRP benchmark instances we created. We compare the results of IVNS with those produced by CPLEX on small-sized instances. We also report the computational results on large-sized instances. To compare the performance of the IVNS algorithm with state-of-the-art algorithms, we run IVNS on a set of popular benchmark instances of the related Time-dependent Pollution Routing Problem (TDPRP) and report the comparative results.

If not stated otherwise, the presented results are the average of 20 independent runs. The instances and results are available online.[1]

The IVNS algorithm was coded in C++ and run on a Linux cluster system with an AMD Opteron 4184 processor (2.8 GHz and 2 GB RAM) running Ubuntu 12.04. For the general CPLEX solver, we used the latest version 12.6.

5.1. Parameter settings

The proposed IVNS algorithm relies on a set of correlated parmeters. To achieve a reasonable tuning of the parameters, we adopt the iterated F-race (IFR) method which alllows an automated parameter configuration (Birattari, Yuan, Balaprakash, & Stutzle, 2010). Table 5 presents the final values for each parameter, together with the range of values determined by preliminary experiments. We use the 100-node instances as training set and tuning budget is 1000 runs of IVNS.

We conclude the influence of the parameters as follows. The solution quality increases as the values of ω ad π rise and remains steady after a particular point. These paricular points for π) and ω need to be identified in order to guarantee the good performance of IVNS. The speed value (v_s) has a significant impact on the final output. As we have analysed in Section 2.2, the speed parameter should be set a value neither too high nor too low. The tuning results show that the total cost is 50% lower when choosing the best value ($90km/h$) than choosing another worse value ($120/54km/h$). For the rest of the parameters (ξ, ζ, τ, ρ_t, ρ_c, ρ_e), the algorithm performance is not sensitive to their settings.

5.2. TDEVRP benchmark Instances

The benchmark instances of the TDEVRP problem proposed in this paper were modified from the benchmark instances introduced by Goeke and Schneider (2015) for the related EVRPTW problem. The benchmark set is composed of nine sets of a number of nodes varying from 10 to 200, 20 instances for each size (180 instances in total). The instances contain: (i) distances between each pair of nodes which is based on the actual geographical distance of randomly selected cities from the United Kingdom, (ii) demand, service duration and time window at each customer node. (iii) max number of vehicles. We raise the number of the available vehicles for all instances to make them feasible in the time-dependent setting. We note that for the instance TDEVRP_20_17, no feasible solution exists, so we fix the instance data to allow feasible solutions by slightly altering the distance matrix. Table 6 presents the parameters in the time-dependent model.

5.3. Results on the TDEVRP instances

This section presents the results of the proposed IVNS algorithm on the set of 180 TDEVRP instances we introduce in Section 5.2.

First, we solve the ILP model introduced in Section 2.3 by CPLEX on small instances and compare the results with those of the IVNS algorithm. Franceschetti et al. (2013) indicated that it is never optimal to drive under v_e in a free-flow period. So we discretize the free-flow speed into 15 levels from v_e to v_f when using CPLEX. Experimental results show that within a time limit of 5 h CPLEX can only solve instances with 10 and 15 nodes. The results are shown in Table 7.

Table 5
Parameters for the IVNS algorithm.

	Description	Test range	Value
v_s	fix free-flow speed used for local search	$[54, 120]$	90
ω	size of neighbourhood for each node	$[10, 20]$	15
ζ	violation parameter increment multiplier	$[5, 50]$	10
ξ	threshold value to accept a bad move	$[-5, -15]$	-10
π	max iteration number	$[10, 30]$	15
τ	loop to roll back without improvement	$[1, 5]$	3
ρ_t	violation weight for time violation	$[0.0005, 0.0015]$	0.001
ρ_c	violation weight for loading violation	$[0.05, 0.15]$	0.1
ρ_e	violation weight for energy violation	$[0.001, 0.01]$	0.005

Table 6
Parameters to character the TDEVRP benchmark instances.

Notation	Description	Unit	Value
T_1	end of morning peak	s	3600
T_2	start of evening peak	s	28,800
u_0	length of planning horizon	s	32,400
v_f	max free-flow speed	km/h	90
v_c	max congestion speed	km/h	10
Y	vehicle battery capacity	kwh	200
Q	vehicle loading capacity	kg	3650

[1] https://www.researchgate.net/project/TDEVRP.

Table 7
Comparison with CPLEX on the set of 20 small TDEVRP instances with 10 and 15 nodes. Column *CPLEX* gives the optimal solution found by CPLEX. Columns *Best* and *Avg* present the best and average value found by the proposed IVNS algorithm. Columns *Dev* show the deviation between the best value found by IVNS algorithm and the optimal value found by CPLEX ($\frac{Best-CPLEX}{CPLEX}$). In the case that CPLEX does not converge to an optimal integer solution within the time limit, the best integer solution found is presented as Bold numbers.

Instance ID	Instances with size 10				Instances with size 15			
	CPLEX (£)	IVNS			CPLEX (£)	IVNS		
		Best (£)	Avg (£)	Dev (%)		Best (£)	Avg (£)	Dev (%)
1	202.24	202.293	205.87	0.02	328.72	333.719	338.69	1.52
2	241.11	240.407	243.37	−0.29	231.16	235.095	235.31	1.7
3	232.82	233.89	234.34	0.45	321.56	321.695	331.4	0.04
4	216.92	214.428	217.45	−1.14	337.53	342.306	343.86	1.41
5	201.65	201.629	201.89	−0.01	345.86	345.86	400.57	0.0
6	250.17	257.435	263.79	2.9	254.57	258.51	288.78	1.54
7	221.53	221.506	222.73	−0.01	290.98	295.724	305.39	1.63
8	285.89	289.865	290.21	1.39	193.64	193.622	193.78	0.0
9	203.04	203.026	203.44	0.0	306.02	305.996	306.12	0.0
10	218.21	221.373	221.37	1.44	238.94	238.94	255.76	0.0
11	324.86	326.512	340.21	0.5	303.44	307.376	317.4	1.29
12	208.86	206.888	207.11	−0.94	**348.01**	348.01	389.66	0.0
13	222.01	226.397	232.62	1.97	**284.2**	281.49	308.78	−0.95
14	199.44	199.422	203.42	0.0	**375.81**	375.82	428.29	0.0
15	152.26	152.26	156.16	0.0	243.64	249.599	249.84	2.44
16	212.75	212.211	212.6	−0.25	226.78	230.722	230.9	1.73
17	210.59	210.84	233.35	0.11	321.63	322.935	325.78	0.4
18	**192.64**	192.313	192.89	−0.16	352.93	352.33	415.32	−0.17
19	195.93	199.873	201.57	2.01	194.32	193.304	193.98	−0.52
20	196.55	196.529	196.6	−0.01	247.72	252.696	253.21	2.0

Table 8
Computational results of IVNS on the set of 20 TDEVRP instances with 10 to 20 nodes. Column *Avg* and *Best* presents the average and best results for each instance respectively. Column *Gap* shows the gap between the best and average values ($\frac{Avg-Best}{Best}$)

Instance ID	Instances with size 10			Instances with size 15			Instances with size 20		
	Avg (£)	Best (£)	Gap (%)	Avg (£)	Best (£)	Gap (%)	Avg (£)	Best (£)	Gap (%)
1	205.87	202.293	1.73	338.69	333.719	1.46	380.85	370.897	2.61
2	243.37	240.407	1.21	235.31	235.095	0.09	413.06	396.661	3.97
3	234.34	233.89	0.19	331.4	321.695	2.92	252.09	248.831	1.29
4	217.45	214.428	1.38	343.86	342.306	0.45	395.12	369.244	6.54
5	201.89	201.629	0.12	400.57	345.86	13.65	344.11	338.615	1.59
6	263.79	257.435	2.4	288.78	258.51	10.48	422.87	386.84	8.52
7	222.73	221.506	0.54	305.39	295.724	3.16	275.31	274.616	0.25
8	290.21	289.865	0.11	193.78	193.622	0.08	329.62	322.354	2.2
9	203.44	203.026	0.2	306.12	305.996	0.04	391.57	374.258	4.42
10	221.37	221.373	0.0	255.76	238.94	6.57	333.18	330.105	0.92
11	340.21	326.512	4.02	317.4	307.376	3.15	483.2	460.18	4.76
12	207.11	206.888	0.1	389.66	348.01	10.68	373.81	357.741	4.29
13	232.62	226.397	2.67	308.78	281.49	8.83	383.38	367.933	4.02
14	203.42	199.422	1.96	428.29	375.82	12.25	515.66	481.429	6.63
15	156.16	152.26	2.49	249.84	249.599	0.09	371.05	363.837	1.94
16	212.6	212.211	0.18	230.9	230.722	0.07	407.21	379.913	6.7
17	233.35	210.84	9.64	325.78	322.935	0.87	352.94	343.46	2.76
18	192.89	192.313	0.29	415.32	352.33	15.16	434.12	419.436	3.38
19	201.57	199.873	0.84	193.98	193.304	0.34	383.54	369.819	3.57
20	196.6	196.529	0.03	253.21	252.696	0.2	399.31	388.7	2.65

As shown in Table 7, the proposed IVNS algorithm can achieve near-optimal results for all small TDEVRP instances with small deviations (less than 3%), and for 20 out of 40 instances IVNS algorithm achieves the optimal solution. In 11 cases, the deviation is a small negative value, implying that IVNS algorithm achieves better results on these instances than CPLEX. This is because when solving by CPLEX, the discretized speed levels may make it impossible to reach the theoretical optimal speed for some arcs, while IVNS algorithm allows continuous speed values.

The average running times of the IVNS algorithm on 10-node and 15-node instances are 9.13 s and 20.31 s respectively, while the average running times of CPLEX are correspondingly 2.3 h and 4.2 h (the running times of those instances that cannot be solved to optimality within 5 h is set to 5 h).

Tables 8–10 present the results of the IVNS algorithm on all TDEVRP instances. And Table 12 lists the average running times across all problem sizes.

The proposed IVNS is robust as there is a low gap between the best and average results. While for many small instances, the IVNS algorithm can find near-optimal solutions, its performance on these instances is relatively unstable across different runs with a gap up to 15.13%. However, the algorithm is more robust on larger instances with a maximum gap of 2.38% than on small instances.

5.4. Sensitivity analysis on the influence of congestion

In this section, we present additional experimental results to show how congestion affects the solution of TDEVRP. For this

Table 9
Computational results of IVNS on the set of 20 TDEVRP instances with 25 to 75 nodes.

Instance ID	Instances with size 25			Instances with size 50			Instances with size 75		
	Avg (£)	Best (£)	Gap (%)	Avg (£)	Best (£)	Gap (%)	Avg (£)	Best (£)	Gap (%)
1	347.82	331.283	4.75	698.71	691.508	1.03	1139.09	1126.85	1.07
2	404.11	401.43	0.66	712.73	697.54	2.13	1003.88	987.647	1.61
3	262.22	260.598	0.61	734.84	730.364	0.6	1014.58	999.119	1.52
4	312.1	310.442	0.53	874.2	861.184	1.48	957.94	945.524	1.29
5	389.2	387.02	0.56	753.46	718.24	4.67	1060.8	1051.63	0.86
6	354.99	352.113	0.81	706.76	700.362	0.9	1095.98	1082.06	1.27
7	400.21	379.181	5.25	653.09	648.197	0.74	1119.8	1109.08	0.95
8	402.72	390.056	3.14	678.89	673.115	0.85	1127.96	1111.93	1.42
9	354.47	353.434	0.29	798.24	785.035	1.65	1089.32	1067.54	1.99
10	420.78	415.977	1.14	823.28	809.94	1.62	1141.61	1118.95	1.98
11	431.12	426.999	0.95	734.42	729.513	0.66	849.2	841.777	0.87
12	468.68	461.475	1.53	668.86	661.142	1.15	1009.94	997.03	1.27
13	301.17	299.268	0.63	740.48	724.357	2.17	1140.47	1123.11	1.52
14	435.23	432.258	0.68	772.87	763.148	1.25	1081.93	1058.83	2.13
15	448.02	417.725	6.76	705.25	689.133	2.28	1155.89	1117.44	3.32
16	405.58	403.759	0.44	681.32	663.649	2.59	1081.97	1065.46	1.52
17	599.68	546.677	8.83	567.78	561.369	1.12	1081.59	1053.66	2.58
18	480.73	460.557	4.19	805.9	795.754	1.25	1020.55	998.183	2.19
19	475.52	470.545	1.04	701.62	692.772	1.26	990.91	975.238	1.58
20	416.61	412.914	0.88	792.51	776.725	1.99	1074.16	1066.56	0.7

Table 10
Computational results of IVNS on the set of 20 TDEVRP instances with 100 to 200 nodes.

Instance ID	Instances with size 100			Instances with size 150			Instances with size 200		
	Avg (£)	Best (£)	Gap (%)	Avg (£)	Best (£)	Gap (%)	Avg (£)	Best (£)	Gap (%)
1	1456.4	1445.83	0.72	1799.37	1769.58	1.65	2567.89	2546.3	0.84
2	1376.99	1352.64	1.76	2020.74	1998.58	1.09	2376.76	2343.83	1.38
3	1302.37	1283.4	1.45	1778.93	1750.01	1.62	2481.4	2457.71	0.95
4	1323.34	1297.83	1.92	1997.98	1974.03	1.19	2366.7	2338.67	1.18
5	1298.46	1287.71	0.82	1832.57	1809.48	1.25	2613.52	2590.69	0.87
6	1435.71	1415.35	1.41	1849.4	1828.3	1.14	2315.23	2293.08	0.95
7	1255.17	1236.22	1.5	2055.86	2037.36	0.89	2449.78	2425.61	0.98
8	1314.83	1293.74	1.6	1911.77	1879.55	1.68	2547.4	2524.33	0.9
9	1202.39	1183.98	1.53	1998.18	1970.42	1.38	2281.17	2247.07	1.49
10	1265.83	1244.25	1.7	1972.06	1925.08	2.38	2640.66	2614.16	1.0
11	1443.91	1418.68	1.74	2022.53	2004.99	0.86	2368.86	2328.47	1.7
12	1244.98	1223.68	1.71	2102.89	2082.58	0.96	2537.59	2520.48	0.67
13	1345.18	1331.01	1.05	1926.1	1906.12	1.03	2536.13	2519.35	0.66
14	1492.79	1470.98	1.46	1997.99	1971.02	1.34	2464.45	2434.39	1.21
15	1544.15	1528.26	1.02	1768.88	1736.14	1.85	2513.15	2482.86	1.2
16	1198.24	1185.76	1.04	2005.45	1972.74	1.63	2490.58	2456.53	1.36
17	1509.6	1496.77	0.84	2001.53	1974.24	1.36	2613.34	2585.42	1.06
18	1311.06	1302.23	0.67	1979.81	1959.11	1.04	2469.01	2448.67	0.82
19	1245.76	1225.2	1.65	2137.99	2091.43	2.17	2243.83	2219.79	1.07
20	1483.14	1468.86	0.96	2087.89	2060.86	1.29	2587.0	2555.01	1.23

purpose, we extend the traffic peak period from one hour to three hours. In this case, a day is composed of a 3-h morning peak period, a 3-h free-flow period and a 3-h evening peak period. Detailed results are reported in Table 11.

From Table 11 we can see that compared to the 1-h case, the 3-h congestion requires more cost to complete the task with an average increase of 11.09% (maximum 23.58%). The increase of energy cost is larger than the increase of total cost in percentage for each instance (averagely 15.87%). We can conclude that the energy consumption contributes more than other costs (e.g., driver, fixed cost). This conclusion is expected because Fig. 3 shows that a low travel speed results in an increase in both energy cost and total cost.

From Table 11 we can also see that multiple times of recharging are required when congestion becomes longer. For 11 (out of 20) instances multiple-recharging for at least one vehicle is required. In the worst case (see Instance 15) 4 vehicles need to recharge for more than one time during a day. While in the 1-h congestion setting, there is no need of multiple-recharging.

We can therefore conclude that a long congestion may lower the transportation efficiency (at large extent) in the context of electric vehicle because multiple-recharging inevitably induces extra travel distance and requires more travel time. In this regard, electric vehicles suffers more from a long congestion than traditional vehicles. It is more crucial for electric vehicles to avoid congestion, or they should be equipped with replaceable batteries to avoid multiple recharging along the road.

5.5. Analysis on the effectiveness of the concatenation operators

To validate the effectiveness of the proposed evaluation method, we provide a set of comparison results where the fast evaluation method is not adopted. In this case, for each move of the VND procedure, the new route has to be evaluated by calculating the costs and constraint violations arc by arc from the beginning to the end of the route(the complexity is $O(n)$). As the comparison experiment gets exactly the same result with

Table 11
Comparison of 100-node instances with different congestion time. The columns *Total Cost* shows the total cost required (1-h/3-h) and *Increase* shows the increase of total cost in percentage ($\frac{3-hour-1-hour}{1-hour}$). The following two columns present the value and increase of energy cost. The last column *Multiple Recharging* shows the number of routes with multiple rechargings in the solution.

Instance ID	Total Cost (£)	ALNS		IVNS	
		Increase (%)	Energy Cost (£)	Increase (%)	Multiple Recharging
1	1456.4/1539.94	5.73	518.65/572.54	10.39	0/0
2	1376.99/1645.11	19.47	415.8/499.97	20.24	0/1
3	1302.37/1480.86	13.7	420.79/494.17	17.43	0/0
4	1323.34/1379.32	4.23	471.96/518.09	9.77	0/1
5	1298.46/1320.32	1.68	413.27/438.29	6.05	0/0
6	1435.71/1620.81	12.89	559.62/646.55	15.53	0/1
7	1255.17/1477.44	17.7	478.86/607.79	26.92	0/0
8	1314.83/1368.85	4.1	445.51/465.96	4.59	0/0
9	1202.39/1234.1	2.63	415.78/443.55	6.67	0/0
10	1265.83/1534.56	21.22	474.48/585.55	23.4	0/2
11	1443.91/1554.39	7.65	447.42/518.67	15.92	0/2
12	1244.98/1346.06	8.11	480.12/532.54	10.91	0/2
13	1345.18/1625	20.8	499.54/615.71	23.25	0/1
14	1492.79/1634.97	9.52	562.74/665.16	18.2	0/0
15	1544.15/1873.42	21.32	612.11/765.42	25.04	0/4
16	1198.24/1230.35	2.67	433.81/475.71	9.65	0/0
17	1509.6/1670.54	10.66	498.43/581.99	16.76	0/2
18	1311.06/1363.68	4.01	426.92/459.54	7.64	0/0
19	1245.76/1539.52	23.58	476.03/618.16	29.85	0/1
20	1483.14/1634.63	10.21	508.09/605.97	19.26	0/2

Table 12
Average running time of LSH on all instances of each size.

Problem size	10	15	20	25	50	75	100	150	200
Running time (s)	9.13	20.31	35.8	123.97	514.25	610.03	888.71	2104.3	4571.59
Comparison (s)	113.73	197.49	424.38	1654.46	7384.88	9002.49	12893.5	27823.03	66963.23

Section 5.3, we only presents the running time for both sets in Table 12.

The running time of the proposed IVNS can be up to 4500 s for some 200-node instances. Such a computational overhead is quite acceptable given the very high complexity of the TDEVRP problem. While for the comparison set the running time is much longer for all problem sizes. For 200-node instances the running time without the fast evaluation method can be 66,963 s (more than 18 h). This result proves the fact that proposed fast evaluation method greatly enhances the calculation efficiency of local search moves and therefore the local search method can serve as an alternative solution method for time-dependent vehicle routing problems.

Table 13
Comparison of IVNS and ALNS on the set of 20 TDPRP benchmark instances with 100 nodes. Columns *Congestion* is the calculated congestion period for each instance. *Best* and *Avg* shows the best and average result of the two algorithms. *Gap* shows the gap between the best and average values($\frac{Avg-Best}{Best}$). Column *Improv* shows the improvement of average results achieved by IVNS compared to ALNS ($\frac{Avg_{ALNS}-Avg_{IVNS}}{Avg_{ALNS}}$). Last row gives the two-sided p value of Wilcoxon signed-rank test of the *Best* and *Avg* columns for two algorithms.

Instance ID	Congestion	ALNS			IVNS			
		Avg (£)	Best (£)	Gap (%)	Avg (£)	Best (£)	Gap (%)	Improv (%).
1	13788	1775.85	1739.33	2.09	1738.72	1730.52	0.47	2.13
2	13269	1661.87	1622.12	2.45	1614.63	1607.92	0.41	2.92
3	13663	1573.72	1549.33	1.57	1551.9	1544.39	0.48	1.4
4	14037	1565.55	1547.92	1.13	1548.39	1539.26	0.59	1.1
5	13713	1574.33	1552.31	1.41	1525.63	1514.34	0.74	3.19
6	14217	1791.34	1759.96	1.78	1771.72	1763.57	0.46	1.1
7	13631	1528.59	1518.92	0.63	1502.26	1490.89	0.76	1.75
8	12826	1537.15	1530.04	0.46	1505.69	1493.23	0.83	2.08
9	13288	1421.93	1409.89	0.85	1405.25	1398.11	0.51	1.18
10	13524	1555.16	1520.6	2.27	1487.71	1483.49	0.28	4.53
11	13845	1740.56	1727.62	0.74	1700.98	1694.57	0.37	2.32
12	13783	1532.78	1520.6	0.8	1462.22	1455.65	0.45	4.82
13	13822	1692.68	1680.68	0.71	1661.2	1647.76	0.81	1.89
14	13292	1789.21	1763.17	1.47	1742.85	1735.28	0.43	2.66
15	13688	1887.12	1854.64	1.75	1824.06	1819.01	0.27	3.45
16	14651	1506.27	1489.38	1.13	1481.36	1465.35	1.09	1.68
17	14651	1801.15	1776.99	1.35	1750.37	1743.18	0.41	2.9
18	14451	1585.78	1570.27	0.98	1562.55	1556.58	0.38	1.48
19	13983	1500.69	1464.38	2.47	1469.26	1456.91	0.84	2.13
20	13961	1856.4	1839.29	0.93	1818.89	1800.3	1.03	2.06
p					8.85E−5	1.03E−4		

Table 14
Comparison of IVNS and ALNS on the set of 20 TDPRP benchmark instances with 200 nodes.

Instance ID	Congestion	ALNS			IVNS			
		Avg (£)	Best (£)	Gap (%)	Avg (£)	Best (£)	Gap (%)	Improv (%)
1	13305	2961.79	2940.28	0.73	2932.34	2912.33	0.68	1.0
2	13991	2846.0	2822.32	0.83	2819.6	2807.22	0.44	0.93
3	14094	2999.55	2985.75	0.46	2942.98	2916.05	0.92	1.92
4	13983	2798.91	2775.31	0.85	2782.5	2757.63	0.9	0.58
5	13171	3033.96	2998.85	1.17	2991.92	2976.2	0.52	1.4
6	13959	2746.86	2734.37	0.45	2720.73	2707.23	0.49	0.96
7	14070	2924.4	2901.6	0.78	2887.71	2875.81	0.41	1.27
8	14137	3042.72	3025.49	0.56	3024.97	3010.53	0.47	0.58
9	13892	2732.12	2706.26	0.95	2676.01	2661.57	0.54	2.09
10	13997	3214.45	3185.81	0.89	3159.71	3145.67	0.44	1.73
11	13968	2808.32	2793.15	0.54	2787.32	2769.1	0.65	0.75
12	12883	2941.13	2906.05	1.2	2897.13	2871.65	0.88	1.51
13	13614	3047.95	3010.19	1.25	3001.91	2986.18	0.52	1.53
14	13775	2890.13	2876.86	0.46	2874.23	2863.25	0.38	0.55
15	13349	2967.42	2949.03	0.62	2912.04	2893.8	0.63	1.9
16	13617	2936.96	2906.59	1.04	2900.74	2883.45	0.59	1.24
17	13540	3146.65	3105.67	1.31	3050.23	3036.01	0.46	3.16
18	13492	2892.18	2875.36	0.58	2859.66	2849.44	0.35	1.13
19	13875	2687.2	2656.33	1.16	2655.06	2643.38	0.44	1.21
20	13913	3117.7	3073.94	1.42	3062.87	3050.66	0.4	1.79
p					8.85E−5	8.85E−5		

5.6. Results on TDPRP instances

To further validate the effectiveness of the proposed IVNS algorithm, we conducted experiments on the closely related TDPRP instances. We compare the IVNS algorithm with the state-of-the-art ALNS algorithm proposed by Franceschetti et al. (2016), which holds the best-known results in the literature for the tested instances.

The problem instances proposed by Franceschetti et al. (2016) were created based on the PRPLIB in which the same graphs as the TDEVRP instances are used. The planning horizon consists of an initial congestion period and a free-flow period (a two-level speed function). The length of the congestion period is calculated based on the time windows of customer nodes to make sure there are feasible solutions to the instances. Moreover, drivers are paid from the beginning of the planning horizon instead of their departure time. In this case we simply replace the electric energy cost with fossil fuel cost and no other adaptations of IVNS is required.

The IVNS algorithm achieves almost the same results as ALNS on the small instances with no more than 25 nodes, therefore we focus on the larger instances with 100 and 200 nodes. The comparative results between INVS and ALNS are reported in Tables 13 and 14.

From the results, we can see that the proposed IVNS outperforms ALNS and achieves better average values on all 100-node and 200-node problem instances of TDPRP. For 39 out of 40 instances IVNS find new best results. Moreover, IVNS is more robust as the gap is lower than ALNS. Results of Wilcoxon sighed-rank test validates the advantages of IVNS with a very small two-sided p value.

Note that Theorem 2 does not apply when the drivers are paid from the beginning of the planning horizon, and this enhances the complexity of a local search move evaluation to $O(n)$, which causes a significant increase on algorithm running time.

6. Conclusion

In this paper, a new problem called the Time-dependent Electric Vehicle Routing Problem (TDEVRP) is introduced. TDEVRP considers the routing of electric vehicles under congestion. Time window, vehicle capacity, and battery charging constraints are also considered. This problem is close to reality, but it is NP-hard and very difficult from the computational perspective.

We propose an ILP model to formulate the problem. Based on this model, experimental results show that CPLEX is only able to solve problems of very limited size (with less than 15 nodes). To solve large-sized problems, we develop an variable neighbourhood search (IVNS) algorithm. IVNS follows the well-known iterated local search framework and is composed of a variable neighbourhood descent procedure (VND) and a perturbation procedure. The VND examines four neighbourhoods in a cyclic manner until no improvement can be found in any of the neighbourhoods. The efficiency of the IVNS algorithm is largely determined by the evaluation of an elementary concatenation operation which is frequently invoked in local search moves. For this reason, we proposed an efficient method to evaluate the concatenation operation with time complexity of O(1). We note that this method can apply to other time-dependent problems even if the linear relationship does not apply (e.g., agile satellite scheduling (Chu et al., 2017), time-dependent orientation problem (Gavalas et al., 2015)).

To evaluate the performance of the proposed IVNS algorithm, we created a set of TDEVRP instances that were extended from the benchmark instances of EVRPTW. Experimental results show that the IVNS demonstrates excellent performance. For small instances, the proposed IVNS can find optimal or near-optimal results very fast (within seconds). For large size instances, the gap between the average and best solution value is tiny (no more than 2.38%) for most instances. Furthermore, we compare the proposed IVNS with the state-of-the-art ALNS, which produces the best result in literature. We test these two algorithms on time-dependent pollution routing problem instances, and the comparative results show that the IVNS algorithm easily dominates ALNS. In particular, the IVNS discovers 39 new best-known solutions out of 40 benchmark instances available in the literature.

We conducted an additional experiment to show that a long congestion may lower the transportation efficiency (at large extent) in the context of electric vehicle due to multiple-charging. Based on this result, we would suggest the decision makers to avoid congestion routes when they plan routes for their vehicle drivers and to use vehicles with replaceable batteries to avoid multiple recharging.For future work, we would like to consider partial charging and replacable battery in the TDEVRP. It is because, in reality, drivers may prefer a partial charging in certain situations (Keskin & Catay, 2016). And a replacable battery will certainly solve the case of multiple recharging.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Ji Lu: Conceptualization, Methodology, Software, Writing - original draft. **Yuning Chen:** Writing - review & editing, Supervision, Project administration. **Jin-Kao Hao:** Validation, Formal analysis, Investigation. **Renjie He:** Resources, Funding acquisition.

Acknowledgment

We are grateful to the anonymous referees for their valuable suggestions and comments which helped ups to improve the presentation of the paper. The work is partially supported by the National Natural Science Foundation of China, Grant 71701203.

References

Akyelken, N. (2016). Green logistics: Improving the environmental sustainability of logistics. *Transport Reviews, 31*, 547–548.
Barth, M., Younglove, T., & Scora, G. (2005). Development of a heavy-duty diesel modal emissions and fuel consumption model. PATH research report,..
Barth, M. J., Wu, G., & Boriboonsomsin, K. (2015). Intelligent transportation systems and greenhouse gas reductions. *Current Sustainable/Renewable Energy Reports, 2*, 90–97.
Bektas, T., & Laporte, G. (2011). The pollution-routing problem. *Transportation Research Part B Methodological, 45*, 1232–1250.
Birattari, M., Yuan, Z., Balaprakash, P., & Stutzle, T. (2010). F-race and iterated f-race: An overview. In *Experimental methods for the analysis of optimization algorithms* (pp. 311–336).
Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., & Juan, A. A. (2014). Rich vehicle routing problem: Survey. *ACM Computing Surveys, 47*, 1–28.
Chen, J., Qi, M., & Miao, L. (2016). The electric vehicle routing problem with time windows and battery swapping stations. In *IEEE international conference on industrial engineering & engineering management*. .
Chu, X., Chen, Y., & Tan, Y. (2017). An anytime branch and bound algorithm for agile earth observation satellite onboard scheduling. *Advances in Space Research, 60*, 2077–2090.
Demir, E., Bektas, T., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research, 223*, 346–359.
Erdogan, S., & Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review, 48*, 100–114.
Figliozzi, M. A. (2011). The impacts of congestion on time-definitive urban freight distribution networks co emission levels: Results from a case study in Portland, Oregon. *Transportation Research Part C, 19*, 766–778.
Franceschetti, A., Demir, E., Honhon, D., Woensel, T. V., Laporte, G., & Stobbe, M. (2016). A metaheuristic for the time-dependent pollution-routing problem. *European Journal of Operational Research, 259*, 972–991.
Franceschetti, A., Honhon, D., Woensel, T. V., Bektas, T., & Laporte, G. (2013). The time-dependent pollution-routing problem. *Transportation Research Part B, 56*, 265–293.
Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., & Vathis, N. (2015). Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operations Research, 62*, 36–50.
Gendreau, M., Ghiani, G., & Guerriero, E. (2015). Time-dependent routing problems: A review. *Computers & Operations Research, 64*, 189–197.
Goeke, D., & Schneider, M. (2015). Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research, 245*, 81–99.
Hiermann, G., Puchinger, J., Ropke, S., & Hartl, R. F. (2016). The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research, 252*, 995–1018.
Ichoua, S., Gendreau, M., & Potvin, J. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research, 144*, 379–396.
Jabali, O., Woensel, T. V., & Kok, A. G. D. (2012). Analysis of travel times and co2 emissions in time-dependent vehicle routing. *Production & Operations Management, 21*, 1060–1074.
Kalayci, C. B., & Kaya, C. (2016). An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications, 66*, 163–175.
Keskin, M., & Catay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C, 65*, 111–127.
Keskin, M., Laporte, G., & Catay, B. (2019). Electric vehicle routing problem with time-dependent waiting times at recharging stations. *Computers & Operations Research, 107*, 77–94.
Koc, C., Bektas, T., Jabali, O., & Laporte, G. (2014). The fleet size and mix pollution-routing problem. *Transportation Research Part B-Methodological, 70*, 239–254.
Kok, A. L., Hans, E. W., & Schutten, J. M. J. (2012). Vehicle routing under time-dependent travel times. *Computers & Operations Research, 39*, 910–918.
Kramer, R., Maculan, N., Subramanian, A., & Vidal, T. (2015a). A speed and departure time optimization algorithm for the pollution-routing problem. *European Journal of Operational Research, 247*, 782–787.
Kramer, R., Subramanian, A., Vidal, T., & Cabral, L. D. A. F. (2015b). A matheuristic approach for the pollution-routing problem. *European Journal of Operational Research, 243*, 523–539.
Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research, 59*, 231–247.
Lin, C., Choy, K. L., Ho, G. T. S., Chung, S. H., & Lam, H. Y. (2014). Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications, 41*, 1118–1138.
Lourenco, H. R., Martin, O. C., & Stutzle, T. (2010). Iterated local search: Framework and applications. *Handbook of Metaheuristics, 146*, 363–397.
Malandraki, C., & Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science, 26*, 185–200.
Malandraki, C., & Dial, R. B. (1996). A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research, 90*, 45–55.
Mancini, S. (2017). A combined multistart random constructive heuristic and set partitioning based formulation for the vehicle routing problem with time dependent travel times. *Computers & Operations Research, 88*, 290–296.
Nagata, Y., Braysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 37*, 724–737.
Polat, O. (2017). A parallel variable neighborhood search for the vehicle routing problem with divisible deliveries and pickups. *Computers & Operations Research, 85*, 71–86.
Rezgui, D., Siala, J. C., Aggounemtalaa, W., & Bouziri, H. (2019). Application of a variable neighborhood search algorithm to a fleet size and mix vehicle routing problem with electric modular vehicles. *Computers & Industrial Engineering, 130*, 537–550.
Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Publications of Darmstadt Technical University Institute for Business Studies, 48*, 500–520.
Shao, S., Guan, W., Ran, B., He, Z., & Bi, J. (2017). Electric vehicle routing problem with charging time and variable travel time. *Mathematical Problems in Engineering, 2017*, 1–13.
Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research, 35*, 254–265.
Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research, 40*, 475–489.
Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2015). Timing problems and algorithms: Time decisions for sequences of activities. *Networks, 65*, 102–128.
Xu, Z., & Cai, Y. (2018). Variable neighborhood search for consistent vehicle routing problem. *Expert Systems with Applications, 113*, 66–76.