Published in "Transportation Research Part E: Logistics and Transportation Review", 2018, vol. 118, pp. 392-420, which should be cited to refer to this work.
DOI: 10.1016/j.tre.2018.08.005

The Dial-a-Ride Problem with Electric Vehicles and Battery Swapping Stations

Mohamed Amine Masmoudi [a,*], Manar Hosny[b], Emrah Demir[c], Konstantinos N. Genikomsakis[d], Naoufel Cheikhrouhou [a]

a Geneva School of Business Administration, University of Applied Sciences Western Switzerland (HES-SO), 1227 Carouge, Switzerland
b Computer Science Department, College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Saudi Arabia
c Panalpina Centre for Manufacturing and Logistics Research, Cardiff Business School, Cardiff University, Cardiff CF10 3EU, UK

Abstract

The Dial-a-Ride Problem (DARP) consists of designing vehicle routes and schedules for customers with special needs and/or disabilities. The DARP with Electric Vehicles and battery swapping stations (DARP-EV) concerns scheduling a fleet of Electric Vehicles (EVs) to serve a set of pre-specified transport requests during a certain planning horizon. In addition, these EVs can be recharged by swapping their batteries with charged ones from any battery-swap stations. This study presents three enhanced Evolutionary Variable Neighborhood Search (EVO-VNS) algorithms to solve the DARP-EV. Extensive computational experiments highlight the relevance of the investigated problem and confirm the efficiency of the proposed EVO-VNS algorithms in producing high quality solutions.

Keywords: Electric Vehicles, Dial-a-Ride problem, Battery swapping, Evolutionary variable neighborhood search algorithm

1. Introduction

People with mobility impairment face difficulties in accessing their basic needs, particularly with regard to public transportation and essential healthcare services (Cordeau and Laporte, 2007). The Americans with Disabilities Act (ADA) states that people with disabilities should have the same rights with respect to ease of access to public transportation as other people (ADA, 2009). Such legislations, besides an increase in public awareness to facilitate the lives of the disabled, have led to a substantial demand for specialized transport services that cater for the needs of these people.

Arguably one of the most challenging problems of specialized transport services is the well-known Dial-a-Ride Problem (DARP), which consists of determining vehicle routes for a set of customers (or patients) who need special transport services. In the DARP, it assumed that each user requests transportation from a specific origin to a specific destination. In other words, we have a pair of requests that are connected to the same user: the outbound request (from origin to destination) and the inbound request (from destination to origin). However, it is not mandatory that the customer is transported directly to the destination (i.e., customers may share rides) (Muelas et al., 2013). These requests are also specified within certain desired pickup or drop off times. The traditional objective in the standard Vehicle Routing Problems (VRPs) is to minimize the total transportation costs. In DARPs, though, the satisfaction of customers (e.g., maximum waiting and ride times) is also taken into consideration.

Real-life applications of the DARP may have additional requirements, depending on the vehicle fleet characteristics. This research incorporates several such requirements, namely, *i*) a fleet of electric vehicles; *ii*) recharging stations with battery-swap services; and *iii*) a realistic energy consumption function to estimate the total energy consumption. In what follows, we explain these new concepts in detail.

First, in most DARPs, the transport is executed by a fleet of gasoline-fueled internal combustion engine vehicles. However, these vehicles are known to be a main source of harmful emissions (i.e., air pollution and greenhouse gases (GHGs)) (Demir et al., 2015; Wang et al., 2016). In order to tackle the emissions problem, the use of electric vehicles (EVs) has received considerable attention over the past few years in the field of the Vehicle Routing Problem (VRP) (see, e.g., Schneider et al., 2014; Goeke and Schneider, 2015; Roberti and Wen, 2016; Desaulniers et al., 2016; Hiermann et al., 2016; Hof et al., 2017; Schiffer and Walther, 2017). This relatively new trend is well-known as the Electric Vehicle Routing Problem (E-VRP), which is strongly related to the field of green vehicle routing, since EVs are operated by a clean and renewable energy source (Schneider et al., 2014). More specifically, when a VRP considers any type of alternative fuel vehicles, it is called a green VRP (G-VRP), while if only EVs are considered, it is called E-VRP. This has inspired us to consider electric vehicles in the context of DARP, which, to the best of our knowledge, has not been previously studied in the literature.

Our DARP-EV arises specifically in healthcare services that concern non-emergency transportation of patients, where different patients are transported from certain origins (e.g., homes) to certain destinations (e.g., healthcare service locations) to receive treatment, medical examination or physical therapy. Such real-life applications are common in the field of healthcare transport services as in Australia (Schilde et al., 2011), Hong-Kong, China (Zhang et al., 2015), Italy (Detti et al., 2017), and Germany (Beaudry et al., 2010). However, new technological developments (e.g., electric vehicles) and new challenges (e.g., environmental pollution) require new adaptations for the well-known transportation problems, as studied in Wu et al. (2015); Travesset-Baro et al. (2015) and Martínez-Lao et al. (2017). One such real world transport application service for the transportation of the elderly and the disabled is operated by the company FlexCité, France. FlexCité is a private transport that offers services for the exclusive use of people with limited mobility. FlexCité has a fleet of electric vehicles (type: Electron II TPMR), where each one contains different capacity modes of transportation, with 9 seats place for the disabled person and/or for their accompanies and three wheelchairs places. Other different types of electric vehicles (for example, ambulances and mini-buses) can be found in the company "Cruise Car" in the US and the company "PAM75" in France, with electric vehicle type Nissan e-nv200.

One important difference between the E-VRP and the traditional VRP, though, is that EVs have limitations in terms of driving range, and thus they need to be frequently recharged during their service route. The limitation in the driving range of EVs and considering the need for recharging at specialized stations have been recently studied in the literature (Erdoğan and Miller-Hooks, 2012; Schneider et al., 2014; Goeke and Schneider, 2015; Desaulniers et al., 2016). The main objective of the studied E-VRPs is to plan routes efficiently while considering both customers' visits as well as frequent visits to recharging stations during the working day. Our research is similar to the studied E-VRPs (e.g. Schneider et al., 2014; Goeke and Schneider, 2015), where we incorporate EVs as well as the recharging stations in the route planning while serving users in the context of DARP. Also, our problem conforms to the new regulations imposed by governments and municipalities that aim to decrease emissions, by encouraging companies and agencies to utilize EVs and/or other alternative fuel vehicles in their fleet, due to their benefit in terms of environmental impact.

Second, to employ EVs in route planning, the battery recharging strategy becomes an essential aspect of the problem, due to many underlying challenges. For example, one challenge that arises in this respect is the low energy capacity of batteries, which usually cannot satisfy the needs of general transport customers (Fuller, 2016). Another challenge is that the battery may need several hours (e.g., 2-6 hours) to be fully recharged from empty (Agrawal et al., 2016). In the majority of E-VRPs, the charging strategy can be full recharging with a linear charging function in each visit to a recharging station (Schneider et al., 2014; Goeke and Schneider, 2015; Hiermann et al., 2016), or partial recharging with a linear charging function (Felipe et al., 2014; Schiffer and Walther, 2017; Desaulniers et al., 2016), or partial recharging with a nonlinear function (Montoya et al., 2017; Froger et al., 2017a, b).

Fortunately, though, there is a sound alternative recharging mechanism that allows an EV to be recharged faster in only one to two minutes (Mak et al., 2013). This is done by swapping its battery instead of recharging it at a battery-swap station (Hof et al., 2017). Many researchers have recently considered the battery-swapping model in VRPs (see, e.g., Liao et al., 2016; Hof et al., 2017; Xu et al., 2017; and Liu and Wang, 2017). The battery-swap strategy is particularly useful in the context of the DARP due to the user satisfaction constraints that require limiting the user's ride time. In fact, due to the hard temporal constraints in DARP (time windows, ride time and maximum route duration) (Cordeau and Laporte, 2007; and Parragh et al., 2008), which makes it hard to effectively design the planning to satisfy the users requests, it is more effective to use the battery swapping recharging mechanism, since it allows EVs to be recharged very quickly. In addition, the battery-swapping strategy improves the productivity of vehicles and reduces the charging costs (Yang and Sun, 2015). In this paper, we consider intermediate stops for battery swapping of EVs. In fact, today EVs are quickly entering the market, and as a result public recharging stations are increasingly in demand and are becoming more available~~.~~ Thus, rather than full/partial recharging, we can consider that in each visit to any recharging public station, the depleted battery will be replaced by a full one as followed by Li (2013).

Finally, we note that recent studies of EVs with battery-swap feature consider that the new battery is deployed after around 100 miles in a single trip (see, e.g., Adler and Mirchandani, 2014; Liao et al., 2016; Xu et al., 2017). Nevertheless, this assumption might not be realistic, since the service time to deploy the battery depends on the energy consumed by the vehicle during its journey. These factors include engine efficiency, regenerative power, road slope, etc. (Wu et al., 2015). To include these factors, we apply a realistic energy consumption model of Genikomsakis and Mitrentsis (2017), in order to determine the service time when the depleted battery should be replaced by a full one.

To sum up, the problem at hand is so-called the Dial-a-Ride Problem with Electric Vehicles and battery swapping stations (DARP-EV), which can be considered as a combination of the traditional DARP and the

E-VRP. In addition, we consider that different types of users need to be transported. For example, a user may need a stretcher or a wheelchair. Thus, the EVs fleet considered in our work is a heterogeneous fleet; i.e., it consists of vehicles having different capacity resources, such as passenger seats, stretchers and wheelchairs. Hence, our problem belongs to the heterogeneous DARP category, as studied by Parragh (2011), Braekers et al. (2014), Braekers and Kovacs (2016), and Masmoudi et al. (2016, 2017). Using different capacity resources as well as different types of users is considered more complex and more general than the traditional DARP (with homogeneous capacity vehicles and single type of users) (Parragh, 2011).

Since our DARP is a combination of the classical DARP, which is NP-hard (Cordeau and Laporte, 2007), and the E-VRP, which is also NP-hard (Schneider et al., 2014), the DARP-EV is NP-hard. Thus, it is extremely difficult to solve such problem using conventional methods. Specifically, the need for recharging (battery swapping), given the limited availability of recharging stations, makes the planning very challenging computationally. Therefore, due to its complexity, most researchers resort to developing metaheuristic methods to solve medium and large instances of the DARP and its variants (see., e.g., Parragh et al., 2010; Parragh, 2011; Parragh and Schmid, 2013; Muelas et al., 2013, 2015; Marković et al., 2015; Braekers and Kovacs, 2016; Chassaing et al., 2016; Masmoudi et al., 2016, 2017). On the other hand, exact methods (e.g. Branch and Cut) are used only for solving small instances of the DARP and its variants (Cordeau, 2006; Parragh, 2011; Braekers et al., 2014; and Liu et al., 2015).

Moreover, for the E-VRP, exact methods are not able to solve small instances within a fast computation time (Goeke and Schneider, 2015). For example, Schneider et al. (2014) confirm that instances of 20 customers cannot be solved to optimality by commercial solvers in a reasonable processing time. Thus, the majority of studied research has also developed metaheuristics to solve this problem (see, e.g., Schneider et al., 2014; Goeke and Schneider, 2015; Hof et al., 2016; Hiermann et al., 2016; Keskin and Çatay, 2016). Therefore, we decided to develop a metaheuristic approach to solve the DARP-EV. Namely, we propose three Evolutionary Variable Neighborhood Search (EVO-VNS) metaheuristic algorithms that can help obtain efficient solutions with a reasonable computational effort.

The contributions of this work are as follows: *i*) we investigate a practical extension of the DARP as described above; *ii*) we develop three different variants of an effective Evolutionary Variable Neighborhood Search (EVO-VNS) algorithm to solve the DARP-EV; *iii*) extensive numerical experiments are applied to assess the performance of the proposed methods on newly generated instances and on the benchmark instances of the DARP.

The remainder of this paper is organized as follows. An overview of the recent literature is provided in Section 2. Some problem assumptions are presented in Section 3. A formal description of the problem is given in Section 4, whereas Section 5 describes our proposed evolutionary algorithms. Section 6 presents the computational results and the conclusions are stated in Section 7.

2. Literature review

This section provides a brief review on recent and related routing problems. We first review the studies that focus on metaheuristic algorithms for DARPs, and next we focus on the studies in the domain of electric

VRPs. In addition, before we conclude the review, we explain the motivation behind our proposed method by presenting a brief discussion on the use of hybrid methods in the DARP and other variants of the VRP.

2.1. The dial-a-ride problem

For comprehensive reviews on DARPs, interested readers are referred to the studies of Cordeau and Laporte (2007), Doerner and Salazar-Gonzalez (2014) and Molenbruch et al. (2017).

Due to the complexity of the DARP, several metaheuristic approaches are proposed in the literature. These include Variable Neighborhood Search (VNS) (Parragh et al., 2010; Parragh, 2011; Muelas et al., 2013, 2015), Tabu Search (TS) (Cordeau and Laporte, 2003), Adaptive Large Neighborhood Search (ALNS) (Masson et al., 2014), Genetic Algorithm (GA) (Jørgensen et al., 2007), hybrid column generation and Large Neighborhood Search (hybrid LNS) (Parragh and Schmid, 2013), and Deterministic Annealing (DA) algorithm (Braekers et al., 2014).

The Heterogeneous DARP (HDARP) is one of the most studied DARPs and takes into account several types of users and resources (e.g., a patient seat, a wheelchair space and a stretcher) (Wong and Bell, 2006). In the study of Parragh (2011), the HDARP with two types of vehicles and four resources (i.e., stretcher, wheelchair, staff seat, patient seat and accompanying person) are studied. The authors proposed Branch-and-Cut (B&C) and VNS algorithms to solve the HDARP. In another study, Braekers et al. (2014) studied multiple depots and heterogeneous vehicles and users for the HDARP. The authors proposed B&C and DA algorithms. The HDARP with multiple trips, single depot, and configurable vehicle capacity has been studied by Liu et al. (2015). To improve the bounds of their B&C algorithm, the authors introduced two mixed integer programming models. They were able to solve instances with up to 22 requests within a running time of four hours. Zhang et al. (2015) studied the public patient transportation problem derived from Hong Kong hospital authority using a fleet of conventional fuel-operated ambulances. The authors considered the sterilization requirement of the ambulance after returning to the depot and introduced the driver’s lunch break extension. Later, Lim et al. (2016) considered an application in Hong Kong within the context of the multi-trip DARP by including lunch breaks and the presence of an assistant. An efficient heuristic with an ad-hoc component was developed and tested on a real-life dataset. In a recent study, Masmoudi et al. (2017) proposed a hybrid Genetic Algorithm (GA) to solve the HDARP. Their algorithm includes two crossover operators, the first is based on a sequencing strategy of a one-point crossover, while the second is based on a merging strategy for selecting individual genes from parent solutions. They also used an additional local search phase and four mutation operators to enhance the solution quality. Their algorithm outperforms the current state-of-the-art algorithms proposed for both the DARP and the HDARP.

Some recent works address various extensions of the DARP, which consider more realistic concepts (e.g., Marković et al., 2015; Masmoudi et al., 2016; Braekers and Kovacs, 2016) and some dynamic pricing procedures (see, e.g., Sayarshad and Chow, 2015; Amirgholy and Gonzales, 2016).

2.2. The electric vehicle routing problems and their applications

In the literature, different routing problems resulted from applying different types of electric vehicles (such as, electrically-powered, battery-powered and plug-in hybrids) in the logistics operations. Among these problems, there are green vehicle routing problems (Nie and Ghamami, 2013; Wang and Lin, 2013; Felipe et al., 2014), transportation network problems (He et al., 2013; Agrawal et al., 2016; Genikomsakis and Mitrentsis, 2017), routing problems with partial/full recharging strategy (Schneider et al., 2014; Goeke and Schneider, 2015; Hiermann et al., 2016), energy efficient routing of electric vehicles (Eisner et al., 2011; Kobayashi et al., 2011; Siddiqi et al., 2011; Sachenbacher et al., 2011), and electric vehicle shortest path (Liao et al., 2016).

Due to the limited driving range of most EVs types, the necessity of visiting stations for recharging is an important aspect that should be considered while planning the service of users' requests in the field of VRPs. Conrad and Figliozzi (2011) incorporate a mixed fleet of vehicles composed of electric and conventional vehicles in the context of VRP. To recharge the electric vehicles, the authors consider that the vehicles can be recharged any time during traveling, where the number of recharging services needed is estimated by dividing the total distance travelled by the limited driving range of the vehicle. This assumption, however, may not be very realistic in real-world applications. Erdoğan and Miller-Hooks (2012) proposed the green VRP (G-VRP), where the vehicles have a limited driving range, and a limited refueling infrastructure is also considered. They assume that the energy consumption is constant, and that the vehicle can visit a recharging station more than once during the route. However, they do not include a capacity or time windows constraints in their model. Montoya et al. (2015) propose an efficient Multi-Space Sampling Heuristic (MSH) to solve the benchmark instances of the G-VRP of Erdoğan and Miller-Hooks (2012). The proposed approach can find 8 new best solutions.

Other studies that consider refueling in G-VRP can be found in Koç and Karaoglan (2016), Adler and Mirchandani (2016) and Yavuz (2017). A survey paper related to the G-VRP can be found in Bektaş et al. (2016).

Schneider et al. (2014) extend the G-VRP of Erdoğan and Miller-Hooks (2012) by considering EVs as a fleet of vehicles, resulting then in E-VRP. In addition, they consider the traditional VRP constraints, such as the capacity of vehicles and time windows of users in their modelling. They developed a hybrid VNS with TS method to solve the E-VRP. Extensive experiments are applied and show that the proposed VNS with TS provide good results on the benchmark instances of the traditional G-VRP and VRP with Times Windows (VRPTW). Goeke and Schneider (2015) study the E-VRP by considering a mixed fleet of electric and conventional vehicles. They use a realistic energy consumption function by considering the mass, capacity of vehicles and constant speed to determine the time when the EV needs to visit a recharging station, which can be considered a similar problem to our case. The energy consumption function is derived from the function of the fuel consumption proposed by Bektaş and Laporte (2011) for the conventional vehicles. Montoya et al. (2017) and Froger et al. (2017a) study the E-VRP with a nonlinear function to recharge the EVs. The same problem is extended by Forger et al. (2017b) by considering that the number of vehicles that can simultaneously be charged at any recharging station is limited by the available number of chargers. To solve this problem, a metaheuristic composed of two-stages is developed (Iterative Local Search as first stage and Benders decomposition as second stage). Felipe et al. (2014) consider a variant of the E-VRP, where they allow partial recharging and consider different charging technologies. The work of Wen et al. (2016) considers an electric vehicle scheduling problem (E-VSP), where a set of buses with certain start and end locations is considered. A mixed integer programming formulation and ALNS were developed, where the objective was to minimize the total distance using the minimum number of vehicles to cover all scheduled trips. Hiermann et al. (2016) proposed an ALNS method with labeling procedures and embedded with local search to solve the Fleet size and Mix E-VRP with time windows. Keskin and Çatay (2016) developed an ALNS approach to solve the E-VRP with partial recharging. Schiffer and Walther (2017) considered electric vehicles in the context of a location routing problem, where both routing and siting decisions are simultaneously taken into account. In addition to the usual time windows and capacity constraints, different charging options are also considered in their work.

Another emergent problem that uses EVs is the Battery Electric Vehicles (BEVs) routing problem with swapping battery stations, which is another stream of research that is related to our problem. Several constraints, such as vehicle capacity, delivery time windows, limited locations of recharging/swapping battery infrastructure, and a maximum route duration are considered in different studies. For example, Mak et al. (2013) proposed two distributionally robust optimization models for the battery-swap location problem under limited information concerning requests distribution.

In another study, Adler and Mirchandani (2014) examined routing of EVs through a network of battery-swap stations. They also improved a Markov Decision Process (MDP) algorithm using an approximate Dynamic Programming (DP) technique to distribute battery switch loads among the stations, in order to reduce the average delay of each vehicle.

Furthermore, numerous studies aimed for accelerating the adoption of BEVs, by attempting to optimally deploy and strategically allocate budget for the charging infrastructure, in order to help sustain the mass-adoption of BEVs (Hof et al., 2017; Liu and Wang, 2017). Moreover, these studies sought also to treat the routing, touring, fleet deployment or relocation problem of BEVs to incorporate them in city logistics and shared mobility (Liao et al., 2016; Boyacı et al., 2017).

To sum up, it is apparent that this kind of E-VRP, where a limited driving range and the need to visit recharging stations, has received the interest of many researchers in the literature. In addition, some works address various extensions of the E-VRP, which consider other realistic concepts (e.g., Liao et al., 2016; Roberti and Wen, 2016; Desaulniers et al., 2016; Hof et al., 2017). For the interested readers, more details on several variants and new trends in electric VRPs can be found in recent surveys by Martínez-Lao et al. (2017) and Pelletier et al. (2017).

However, based on our literature review, it is also observed that although EVs is widely studied in different VRP variants, it is not yet applied in the domain of DARPs. This has inspired us to apply a fleet of EVs instead of the traditional conventional vehicles (CVs), taking into account the limited driving range and the possible need of recharging as in most E-VRPs. Thus, applying EVs in our DARP-EV distinguishes our work from the widely applied EVs in different E-VRPs variants.

2.3. Hybrid solution methods in DARPs and VRPs

Examining previous solution methods of the DARP, it appears that hybrid population-based metaheuristics for solving this problem are not widely applied, with a few exceptions such as the work of Masmoudi et al. (2017), previously discussed in Section 2.1. Also, Masmoudi et al. (2016) endorse the use of local search methods within a population-based metaheuristic algorithm for solving complex versions of DARPs. In their study, a simple hybridization method is proposed by augmenting the Bees Algorithm (which is a population-based method) with two well-known single-solution based algorithms, namely Demon Algorithm (DA) and Simulated Annealing (SA), in order to enhance the intensification around the elite (i.e., best of the best) solutions. In addition, Chassaing et al. (2016) propose an Evolutionary Local Search (ELS) approach for solving the DARP. ELS is an extension of Iterated Local Search (ILS), where a randomized constructive heuristic is used to generate several copies of the current solution to be used as starting solutions for the ELS. Each of these copies is first modified (mutated), before the ELS performs local search by combining six neighborhood structures, which are controlled by dynamically updated probabilities in order to improve its convergence.

Taking a wider look at hybrid methods, especially those involving evolutionary algorithms, we observe that they have been attempted for solving different combinatorial optimization problems and in particular different variants of the VRP. Among these we mention the Hybrid Evolutionary Algorithm (HEA) of Koç et al. (2015) for solving the heterogeneous fleet vehicle routing problem with time windows. The HEA combines Adaptive Large Neighborhood Search (ALNS) with a population-based search, where what is called an Education procedure is applied to repair an offspring resulting from crossover before inserting it back to the population. In addition, extensive search around elite solutions is performed using ALNS. A similar idea of applying an Education procedure for repairing infeasible solutions is applied in the Hybrid Genetic Search with Advanced Diversity Control (HGSADC) of Vidal et al. (2013), for solving a large class of time-constrained vehicle routing problems. However, they consider population diversity as an objective that should be optimized together with the solution quality. For other hybrid population based methods to solve VRP variants including DARPs, interested readers are referred to the survey paper of Braekers et al.(2016) and Molenbruch et al. (2017).

As can be seen from this review, state-of-the-art approaches largely gain from the incorporation of local search metaheuristics to improve the population.

Regarding the hybridization of VNS with population based methods in the literature, we noticed that very few works apply this technique for variants of the VRP. For example, the work of Jabir et al. (2017) extend an Ant Colony Optimization (ACO), which is used for route construction in the multi-depot green vehicle routing problem, with a local search that uses VNS. Guan and Lin (2016) developed a hybrid VNS with ACO to solve the single row facility layout problem. Xia et al. (2016) proposed an efficient hybrid GA using VNS as an improvement phase to solve the dynamic Integrated Process Planning and Scheduling (IPPS) problem. However, to the best of our knowledge, hybridization of VNS with population based methods has not been applied before to any variant of the DARP. Moreover, it seems that the only hybridization of VNS applied in the DARP is developed by Parragh and Schmid (2013), where they proposed a Large Neighboorhood Search (LNS) with VNS.

Interested readers can find other hybrid evolutionary methods and other types of hybridization in different combinatorial optimization problems in the excellent survey paper of Blum et al. (2011).

As previously mentioned, in our work we develop an evolutionary VNS for solving the DARP-EV. What distinguishes our technique from other hybridization techniques in the literature is that instead of integrating the VNS within a fully-operating population-based method as a local search procedure, which is the classical hybridization mechanism, we attempt to transform the VNS itself to a semi-population based method. This is achieved by injecting VNS with several features that are borrowed from evolutionary-based algorithms, as will be explained in detail in Section 5 of this paper.

3. Problem assumptions

Before we introduce the formal problem description, we present in this section some problem features that distinguish our work from previous works and also some simplifying assumptions that have been incorporated to make the problem more manageable. In most studied energy consumption models (e.g., Erdoğan and Miller-Hooks, 2012; Schneider et al., 2014; Koç and Karaoglan, 2016; Adler and Mirchandani, 2016; Yavuz, 2017), the energy consumption is considered constant. Nevertheless, in recent years, researchers started to consider more realistic energy consumption in routing models that incorporate EVs (e.g., Wu et al., 2015; Goeke and Schneider, 2015; Genikomsakis and Mitrentsis, 2017). This has motivated us to use a realistic energy consumption model, based on the one proposed by Genikomsakis and Mitrentsis (2017). Despite this, we have considered some simplifications in our application of EVs in routing, compared to other EVs applications, due to the complexity of our problem as explained next.

Regarding the travel speed, we assume that it is constant on each arc, i.e., acceleration phases are neglected. In addition, in our EVs, we assume that the gradient is also constant over an arc. Nevertheless, it is possible to integrate acceleration patterns in the topology, for example by adding a path having an intermediate vertex in each arc. Thus, each of the added vertices will mark the variation in gradient and acceleration (Simpson, 2005; Genikomsakis and Mitrentsis, 2017). In our framework, it is also assumed that, for each pair (i, j) of the network, the road angle is positive (+2°) if j is a pickup node, whereas the road angle is negative (-2°) if j is a delivery node. Without loss of generality, this approach is employed in order to take into account the effect of the road network topology on the energy consumption (or regeneration).

We note that the speed could be handled as a decision variable, which can be increased in order to satisfy the time windows, and can be reduced in order to decrease the amount of consumed energy (see, e.g., Bektaş and Laporte, 2011; Demir et al., 2012, 2014) for fuel consumption. However, since the vehicle's speed is strongly affected by traffic conditions, we chose not to consider this modeling, similar to the energy consumption model of Goeke and Schneider (2015), which is used to calculate the energy consumption in the E-VRP with times windows and mixed fleet of conventional and electric vehicles. Since the energy consumption affects the battery level (and consequently the need to recharge at a recharging station), these simplifying assumptions seem to be even more important within the context of electric vehicles (Goeke and Schneider, 2015).

Another simplifying assumption that we consider is that the effect of outside temperature on the capacity of the battery is neglected. In contrast to the applied EVs in the VRPs that assume homogeneous capacity vehicles, we introduce in our work a new fleet of EVs containing different capacity resources to serve different types of users, which can also arise in the distribution of goods within the context of VRP. For example, different resources could be needed to transport different types of goods (e.g. liquids and objects) in the same vehicle. In fact, our EVs differ than those studied in the literature in that we consider the load/unload of users during the calculation of the energy consumption function, since this has an influence on the energy consumption during the route. To the best of our knowledge, only Goeke and Schneider (2015) have considered the impact load/unload of goods on the energy consumption of the EVs. However, the difference between the energy consumption model applied by them and that applied in our case is that they consider the efficiency of the electric machine when operating as a motor or as a generator. In addition, they consider the recuperation energy constant, and they do not consider the power consumption of accessories in their model. This is contrary to our model, where we consider these parameters as variables that depend on many factors, based on the model of Genikomsakis and Mitrentsis (2017), as will be defined later.

Finally, we note that there is a slight difference between our model and the model of Genikomsakis and Mitrentsis (2017), where they consider the speed and acceleration as well as the road gradient as variables, while in our case, as mentioned above, we have modeled these as constant on each traveling arc. Thus, we have slightly modified the original model of Genikomsakis and Mitrentsis (2017), which is described in detail in the next section.

4. Problem description

The DARP-EV can be formally described as follows. We have a complete directed graph $G = (V, A)$, where V is the set of all nodes and $A = \{(i, j): i, j \in V, i \neq j\}$ is the set of arcs connecting each pair of nodes. The set V is further partitioned into two subsets; $N = \{1, \dots, 2n\}$ is the set of pickup and delivery nodes and $F = \{2n + 1, \dots, 2n + f\}$ is the set of Battery Swap Station (BSS) nodes. For n user requests, $P = \{1, \dots, n\}$ and $D = \{n + 1, \dots, 2n\}$ represent the pickup and the delivery nodes, respectively. The nodes 0 and $2n$+1+f denote the same depot, where every route begins at depot 0 and ends at $2n$+1+f. We assume that the depot is also considered as a BSS node. The complete set of nodes can now be represented as $V = N \cup F \cup \{0, 2n + 1 + f\}$. An arc (i, j) in set A has an associated non-negative travel cost c_{ij} equal to the distance d_{ij} ($d_{ij} = c_{ij}$) and a non-negative travel time t_{ij}. Moreover, we assume that the number of stops that the vehicle can make for swapping its battery is not limited. The time window to visit any BSS node is set to $[0, T]$, where T is length of the planning horizon.

The mathematical formulation differentiates between visits to users' nodes on one hand, and visits to BSSs nodes and the depot on the other hand. The difference is that each user node must be visited exactly once, while BSS nodes may be visited multiple times or may not be visited at all. Moreover, the depot node must be visited at the beginning and at the end of each tour, and can also serve, if needed, as a BSS node. To allow a subset of vertices to have multiple visits, while others are visited exactly once, a set of f' dummy vertices, $F' = \{2n + f + 1, \dots, 2n + f + f'\}$ are augmented on the graph G (see, e.g., Erdoğan and Miller-Hooks, 2012; Schneider et al. 2014; Goeke and Schneider, 2015). Each of these nodes accounts for a potential visit to a BSS or depot serving as a BSS. In other words, these copies of station nodes are needed in the model to allow for multiple visits at the original set of nodes (even when done by the same vehicle). Thus, we obtain the graph $G' = (V', A')$, where $V' = V \cup F'$. The technique of introducing dummy vertices was proposed by Bard et al. (1998) to allow for intermediate stops at depots, for the purpose of reloading vehicles with goods during their route for delivery.

We assume that a fleet of EVs $K = \{k_1, \dots, k_k\}$ is available and each vehicle capacity is defined with Q^{rk}, which shows the amount of resource r available on each vehicle k. We also assume that there are four types of resources available in each vehicle: *i*) an accompanying person $Q^{0,k}$, *ii*) a handicapped person's seat $Q^{1,k}$, *iii*) a stretcher $Q^{2,k}$, and *iv*) a wheelchair $Q^{3,k}$. The battery capacity for each vehicle is denoted as H, which is consumed at an energy rate EC in each time (in minutes). Each user request is associated with a time window $[T_i^-, T_i^+]$ and a demand requirement q_i^r for each resource r, where this demand at each delivery is equal to q_{n+i}^r = -q_i^r. For the convenience of the users, we implicitly set a limit on the maximum allowed user ride time L_{max}. Finally, a service time s_i is needed when visiting a pickup or delivery node i ($\forall$ *i* $\in$ *N*), and a swapping battery time s_f when visiting a BSS node $f \in F'$. Since each vehicle is assumed to have only one driver, we use the terms (vehicles and drivers) interchangeably. For each driver, the maximum working time per day is T_{max}.

The energy consumption at each time t (in minutes) when visiting a node i $EC_{(i)}(t)$ can be calculated as discussed in Genikomsakis and Mitrentsis (2017) by the following equation (1).

$$EC_{(i)}(t)[\mathrm{W}] = \begin{cases} \left(FR.n_{gear}.n_{gen}\left(\frac{0.001.P_{out}}{P_{mot}}\right).norm+P_{ac}\right)\frac{\sqrt{RTE}}{60} & \text{if } FR<0 \text{ and } P_{bat}<0 \quad (1.a) \\ \left(FR.n_{gear}.n_{gen}\left(\frac{0.001.P_{out}}{P_{mot}}\right).norm+P_{ac}\right)\left(\frac{1}{60}\right)/\sqrt{RTE} & \text{if } FR<0 \text{ and } P_{bat}>0 \quad (1.b) \\ \left(\frac{FR}{\left(n_{gear}.n_{gen}\left(\frac{0.001.P_{out}}{P_{mot}}\right).norm\right)}+P_{ac}\right)\left(\frac{1}{60}\right)/\sqrt{RTE} & \text{if } FR>0 \quad (1.c) \end{cases} \quad (1)$$

The first part (1.a) of equation (1) describes the condition when the renovation energy surpasses the consumption of the accessories and then the battery reserves this overflow of energy. In the contrary, the second part (1.b) considers the situation where the revived energy is not enough for the consumption of the accessories, and as a result, this energy is taken off from the storage. Finally, the third part (1.c) accounts for the condition in which the energy is not regenerated. Consequently, the vehicle is not empowered and the accessories do not consume the energy which is already taken off from the storage. The tractive power FR equation (2) is calculated as:

$$FR[\mathrm{N}]=(\tfrac{1}{2}.\rho.A_f.C_D.v^2+m(u_j).a + C_f.m.a + m(u_j).g.(\sin(\alpha_{ij}) + C_r.\cos(\alpha_{ij}))).v. \quad (2)$$

The mechanical power P_{out} can be calculated using the following equation (3)

$$P_{out}[\mathrm{W}] = \begin{cases} FR.n_{gear} & \text{if } FR < 0, \\ FR/n_{gear} & \text{if } FR > 0. \end{cases} \quad (3)$$

The input power of the motor P_{in}[W] is depending on the sign of traction power to wheel, which is defined by equation (4).

$$P_{in} = \begin{cases} P_{out}.n_{gen}norm & \text{if } FR < 0, \\ P_{out}/n_{mot}norm & \text{if } FR > 0. \end{cases} \quad (4)$$

Hence, if the P_{in} value is negative ($P_{in} < 0$), EV can recuperate the energy during the regenerative state. In this case P_{in} is then calculated by equation (5)

$$P_{in}[\mathrm{W}] = \begin{cases} P_{out}.n_{gen}.norm & \text{if } FR < 0, \\ P_{out}/(n_{mot}.norm) & \text{if } FR < 0. \end{cases} \quad (5)$$

Depending on the mechanical power P_{out} and the input power of the motor P_{in} values, it is necessary to calculate the efficiency of the electric machine n_{mot} or n_{gen} when operating as motor or as a generator, respectively. In this regard, using the coefficients of Table 1, we use the following piecewise function in equation (6) to calculate the curb-efficiency. The same function is also shown in Figure.1.

$$
eff(x) = \begin{cases} (\varphi_1 x + \varphi_2)\dfrac{1}{x+\varphi_3} & \text{if } 0 \le x < 0.25 \\ \varphi_4 x + \varphi_5 & \text{if } 0.25 \le x < 0.75 \\ \varphi_6 x + \varphi_7 & \text{if } x \ge 0.75 \end{cases} \tag{6}
$$

where x presents the mechanical power P_{out} as a fraction of its rated power value P_{mot}, in other words, x =0.001$|P_{out}|/P_{mot}$.

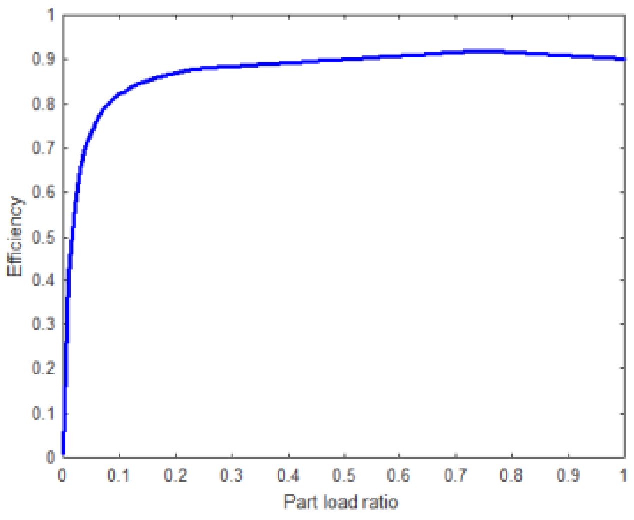

Fig. 1. Piecewise function of the efficiency

Table 1
Coefficients used for motor/generator mode

Coefficient	Motor mode (n_{mot})	Generator mode (n_{gen})
φ_1	0.924300	0.925473
φ_2	0.000127	0.000148
φ_3	0.012730	0.014849
φ_4	0.080000	0.075312
φ_5	0.860000	0.858605
φ_6	-0.073600	-0.062602
φ_7	0.975200	0.971034

Based on the sign of P_{in} in equations (4) and (5), the electric output value of the battery P_{bat} can be calculated using equation (7).

$$P_{bat}=P_{in}+P_{ac}. \tag{7}$$

We note that in Genikomsakis and Mitrentsis (2017) vehicle speeds as well as the acceleration rates are considered as variables. Since vehicle speed is subject to traffic conditions, we did not adopt the variable speed modeling in our study. We assume that vehicle speed is constant in each arc since we do not consider instantaneous acceleration/deceleration phases. However, we have adopted the vehicle mass as a variable on the function of Genikomsakis and Mitrentsis (2017). This is because the mass of the vehicle has an effect on the energy consumption, as explained in Goeke and Schneider (2015). In other words, the consumption depends on the current payload of a user u. Thus, the vehicle mass function $m(u)$ can be defined as $m(u) =$

m_v+$m_u u$, where m_v represents the curb weight and m_u is the weight of the user. In addition, we note that u_j presents the number of users available in the vehicle when arriving at the next user j.

The physical constants and vehicle properties of the specific vehicle "Nissan Leaf model" (Carsales.com.au, 2015) and coefficients are adopted from Genikomsakis and Mitrentsis (2017) and are summarized in Tables 2.

Table 2
Parameters and technical specifications of the vehicle

Notation	Description	Value
g	Gravitational constant (m/s^2)	9.8066
ρ	Air mass density(kg/m^3)	1.25
A_f	Frontal surface area of the vehicle(m^2)	2.19
C_d	Coefficient of aerodynamic drag	0.29
C_f	Mass correction factor	0.05
C_r	Coefficient of rolling resistance	0.008
v	Vehicle speed (km/h)	17
H	Capacity of battery (kW)	40[a]
α	Road angle	2[b]
a	Acceleration (m/s^2)	0
m_v	Vehicle mass + driver (kg)	1633
n_{gear}	Coefficient of gear efficiency	0.97
P_{ac}	Power consumption of accessories(W)	300
P_{mot}	Rate power (kW)	75
$norm$	normalization factor of motor efficiency	0.987
RTE	Round trip efficiency	0.95
m_u	Average weight of user(with wheelchair) in kilograms	70(80)[c]

[a] *Estimated value from Tesla Motors (2013), represents around 100 miles for one trip to be depleted (Lion battery)*
[b] *https://www.engineeringtoolbox.com/slope-degrees-gradient-grade-d_1562.html*
[c] *Average value estimated from Centers for Disease Control and Prevention (2012)*

The DARP-EV consists of planning a set of routes to satisfy a set of transport users by considering the minimization of the total routing costs. A solution to the DARP-EV must satisfy the following conditions: *i*) the pickup node must be visited before its corresponding delivery node, *ii*) the total demand of the route for all visited nodes must not exceed the resource capacity, *iii*) each node must be served within its time window, such that if a vehicle arrives early, it must wait until the beginning of the time window, *iv*) the maximum ride time of each user must not be exceeded, *v*) the same vehicle must visit the pickup and delivery nodes of the same user, *vi*) each vehicle can visit a BSS node for swapping the depleted battery, if the remaining energy level in its battery is not enough to serve the next user, and *vii*) each route should start and end at the same depot and the duration of the route should not exceed the maximum working time.

Figure 2 presents an example of the DARP-EV containing 3 vehicles (routes) and 14 users, where each user i should be picked up from its origin i^+ to its destination i^-, two battery swap stations and the depot, which can also serve as a BSS. The value along each route shows the amount of charge in the battery when the vehicle arrives at the node of each user i and to the BSS node. Vehicle V_1 visits the BSS node F_1 to replace the battery by a full one after servicing the nodes $4^+, 4^-, 7^+, 7^-, 6^+, 6^-, 14^+, 14^-$ in order to be able to service the nodes $5^+, 5^-$ before returning back to the depot. Vehicle V_2 services the nodes $3^+, 11^+, 11^-, 9^+, 9^-, 3^-, 2^+, 2^-, 1^+, 1^-$ and returns to the depot without visiting any F_f nodes. Vehicle V_3 visits F_1 node after servicing the nodes $8^+, 10^+, 8^-$ to continue its travel to serve the nodes $13^+, 10^-, 13^-, 12^+, 12^-$. From Figure 2 it can be observed that a battery swapping station can be visited many times by the same or different vehicles as the F_1 node, and a station may not necessarily be visited at all as the F_2 node.

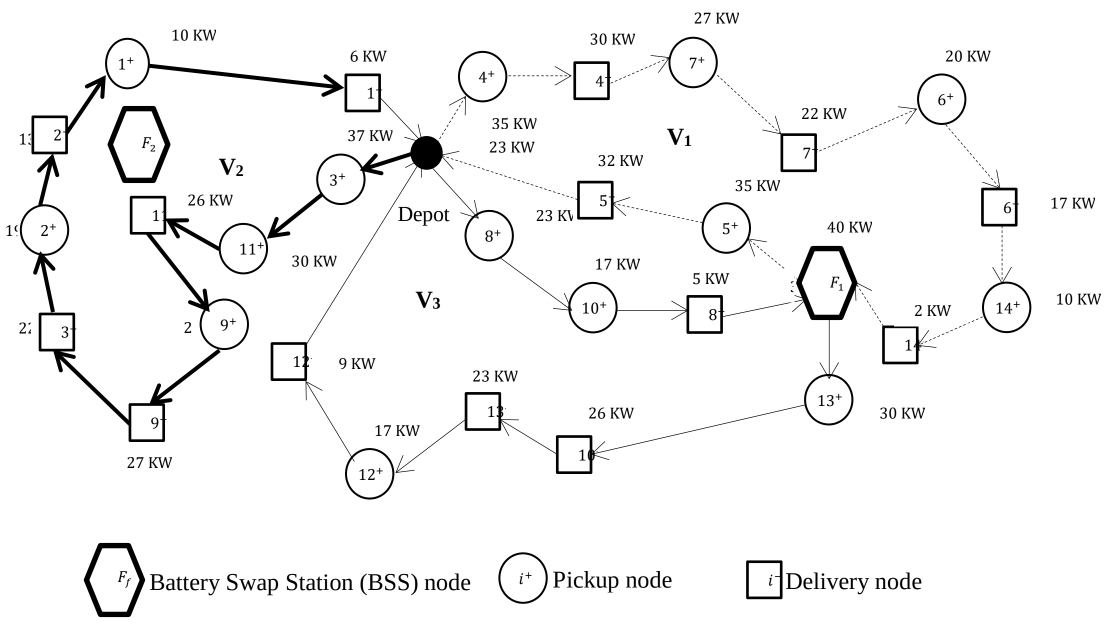

Fig. 2. Illustrative example of the DARP-EV

We now present a Mixed-Integer Non-Linear Program (MINLP) formulation for the DARP-EV that is inspired by the standard DARP formulation as in Parragh (2011) and the G-VRP of Erdoğan and Miller-Hooks (2012), which can be extended to the E-VRP by assuming that the fleet is composed of EVs, as mentioned by Schneider et al.(2014). More specifically, constraints (9)-(21) of our model are already used in the DARP formulation of Parragh (2011), with slight modifications in these constraints by considering the BSS node (and its dummy vertex). Constraints (22)-(24) are the same as in the formulation of Erdoğan and Miller-Hooks (2012). We note that constraints (15), (20), (22) and (23) are not in integer linear form. This is intended to make the problem formulation easier to understand. Since a metaheuristic is used to solve the problem, the integer linear formulation is not needed.

The DARP-EV can be formulated as follows: binary variables x_{ij}^k are equal to 1 if arc (i, j) is included in the solution and 0 otherwise. Continuous variables B_i^k represent the time that vehicle k starts servicing node i. Continuous variables Q_i^{rk} indicate the load of resource r on vehicle k immediately after visiting node i. Continuous variables l_i^k represent the ride time of user $i \in P$ on vehicle k. Continuous variables $z_{bat}^k(i)$ represent the battery charge level of the vehicle when visiting node i.

$$\text{minimize} \sum_{k\in K}\sum_{i\in V'}\sum_{j\in V'} c_{ij}x_{ij}^k \tag{8}$$

subject to

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^k = 1 \qquad \forall i \in P \tag{9}$$

$$\sum_{k\in K}\sum_{j\in N} x_{ji}^k = 1 \qquad \forall i \in D \tag{10}$$

$$\sum_{j\in N} x_{ij}^k = \sum_{j\in N} x_{j,n+i}^k \qquad \forall k \in K, \forall i \in P \tag{11}$$

$$\sum_{j\in V'} x_{0j}^k = 1 \qquad \forall k \in K \tag{12}$$

$$\sum_{i\in V'} x_{i,2n+f'+1}^k = 1 \qquad \forall k \in K \tag{13}$$

$$\sum_{i\in V'} x_{ij}^k = \sum_{i\in V'} x_{ji}^k \qquad \forall k \in K, \forall j \in V' \tag{14}$$

$$x_{ij}^k = 1 \Rightarrow Q_j^{rk} \ge q_j^r + Q_i^{rk} \qquad \forall k \in K, \forall (i,j) \in A, r \in \{0,1,2,3\} \tag{15}$$

$$0 \le Q_i^{rk} \le Q^{rk} \qquad \forall k \in K, \forall i \in N, r \in \{0,1,2,3\} \tag{16}$$

$$Q_0^{rk} = 0 \qquad \forall k \in K, r \in \{0,1,2,3\} \tag{17}$$

$$l_i^k = B_{n+i}^k - (B_i^k + s_i) \qquad \forall k \in K, \forall i \in P \tag{18}$$

$$t_{i,n+i} \le l_i^k \le L_{\max} \qquad \forall k \in K, \forall i \in P \tag{19}$$

$$x_{ij}^k = 1 \Rightarrow B_j^k \ge B_i^k + s_i + t_{ij} \qquad \forall k \in K, \forall (i,j) \in A' \tag{20}$$

$$T_i^- \le B_i^k \le T_i^+ \qquad \forall k \in K, \forall i \in V' \tag{21}$$

$$z_{bat}^k(j) \le z_{bat}^k(i) - ECt_{ij}x_{ij}^k + H(1 - x_{ij}^k) \qquad \forall j \in N, \forall i \in V', i \ne j, \forall k \in K \tag{22}$$

$$z_{bat}^k(j) \ge \min\{ECt_{j0}, EC(t_{jf} + t_{f0})\} \qquad \forall j \in N, \forall f \in F', \forall k \in K \tag{23}$$

$$z_{bat}^k(j) = H \qquad \forall j \in F', \forall k \in K \tag{24}$$

$$B_{2n+f'+1}^k - B_0^k \le T_{\max} \qquad \forall k \in K, \forall f \in F' \tag{25}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in K, \forall (i,j) \in A'. \tag{26}$$

The objective function (8) aims to minimize the total routing costs, including BSS costs. Constraints (9)-(11) ensure serving the pickup and delivery pair by the same vehicle. Constraints (12)-(13) ensure that each vehicle k starts at the origin depot and ends at the corresponding destination depot, while constraints (14) define arc flows. Constraints (15) and (16) set the capacity values. Constraints (17) make sure that a vehicle leaves the depot with an empty load. Constraints (18) define the ride time of each user in each route, which is bounded by constraints (19). These constraints also enforce the precedence relationship between the pickup and delivery nodes. Constraints (20) define the beginning of service at each node. Constraints (21) impose the time windows. Constraints (22) track the fuel level of the vehicle according to the sequence and type of the visited nodes. If i is a customer node and j is visited immediately after i (x_{ij}^k) with vehicle k, the first term in Constraints (22) will ensure that the fuel level is reduced when the vehicle arrives at j according the distance from i to j and the fuel consumption rate. Constraints (23) guarantee that the vehicle will not be stranded due to shortage in fuel by ensuring that after visiting any customer in the route, there is enough fuel remaining to return to the depot either directly or through a BSS. We note that Constraints (23) can be modified to $z_{bat}^k(j) \ge \{ECt_{j0}, ECt_{jf}\}$, since we only need to make sure that the vehicle can either return to the depot or visit a BSS for battery swapping with the remaining battery. Constraints (24) guarantee that the battery is full after visiting a BSS node. Constraints (25) define the time duration of the route of each vehicle, which is strictly limited by T_{max}. Finally, constraints (26) define binary decision variables.

5. Evolutionary variable neighborhood search algorithms for the DARP-EV

This section presents three different variations of our proposed evolutionary Variable Neighborhood Search (EVO-VNS) algorithm to solve the DARP-EV. The VNS algorithm was first introduced by Mladenovic and Hansen (1997) and used widely since then because of its simplicity, effectiveness, and adaptability to solve many VRPs variants (see, e.g., Mladenović et al., 2012; Carrizosa et al., 2013; Polat et al., 2015; Wei et al., 2015; Sarasola et al., 2016; Todosijević et al., 2016) including DARP and its variants (see, e.g., Parragh et al., 2009; Parragh et al., 2010; Parragh, 2011; Schilde et al., 2011, 2014; Muelas et al., 2013; 2015; Parragh et al., 2014; Detti et al., 2016). Caporossi et al. (2016) and Mladenović et al. (2017) provided recent and successful applications of the VNS in different combinatorial optimizations problems.

Contrary to the basic traditional VNS that searches iteratively around a single initial solution, which limits its exploration power of large search spaces, our EVO-VNS, exploits a population of solutions as normally done in population-based metaheuristics. We combine features from famous evolutionary and swarm based techniques, namely the Genetic Algorithm (GA) proposed by Holland (1975), the Shuffled Frog-Leaping Algorithm (SFLA) proposed by Eusuff et al. (2006), and the Bees Algorithm (BA) of Pham et al (2006). This structure is intended to maintain population diversify (Fang and Wang, 2012) and ensure global exploration for our EVO-VNS. In addition, our algorithm restarts at each VNS iteration from a different initial solution. This can enhance its diversification power, allowing it to skip unnecessary iterations around local optima. Moreover, we enhance the VNS by increasing its intensification power around favorable solutions. This is intended to boost the performance of the classical VNS and improve its convergence towards better solutions. Thus, this process of hybridization shapes a new hybrid VNS that includes the main advantages of two or more well-known metaheuristic approaches in a judicious way. Our approach is a novel approach, since we are not aware of any work that utilizes such evolutionary hybrid VNS for solving any variant of the VRP including the DARP.

The main steps of the EVO-VNS are shown in Algorithm 1. The algorithm runs for a number of iterations until no improvement is achieved of the global best solution after five consecutive iterations. First, an effective construction heuristic is used to generate the initial population of size Pop. Then, for a number of iterations, the following steps are performed. In step 1, each solution in Pop is evaluated according to the fitness function, and solutions are sorted according to fitness from the highest to lowest. In step 2, the population Pop is divided into m groups, each one contains (c) solutions (i.e., $Pop = m.c$). In this process, the first solution goes to group 1, second solution goes to group 2, the m^{th} solution goes to group m, and solution m+1 goes back to the first group and so on. This step is similar to the generation of solutions in the SFLA proposed by Eusuff and Lansey (2003).

In step 3, for each group, the following steps are applied. First, s solutions are selected based on the roulette wheel selection mechanism. If the best solution of the current group is not improved in the last three consecutive iterations, a Merge Crossover 1 (MX1) based on the DARP study of Masmoudi et al. (2017) is applied as a perturbation phase between one solution from (s) and one solution from $(c - s)$ to create a new solution. The main feature in the MX1 operator is that the new solution generated after crossover will inherit information from the already improved solution s in the current group n and another highly diversified solution $(c - s)$. This new solution construction mechanism thus enables the algorithm to achieve the required balance between intensification around a good solution in the group n, and exploration of new regions of the search space.

The MX1 crossover step is then followed by the procedure of repairing infeasible solutions of Masmoudi et al. (2017) (respecting the feasibility of a solution is explained in subsection 5.4). The only difference is that we generate a new solution based on our constructive heuristic described in subsection 5.1, in case the solution is still infeasible after all attempts to repair the current solution. In fact, the MX1 operator permits to produce a new solution that inherits good characteristics of both selected solutions (Masmoudi et al., 2017), in order to better diversify the search, while maintaining the feasibility of the solution as much as possible. In order to implement the MX1 crossover, a simple chromosome encoding of the solution is needed, as described in subsection 5.2.6. If crossover is applied, (s) new solutions are then obtained and replace the original s solutions in the current group; otherwise, the original (s) solutions are not changed.

Then, in step 3.2, these (s) solutions are sent to the intensification phase using an effective single-solution based metaheuristic algorithm, namely VNS. We use here three different types of VNS (i.e., VNS$_1$, VNS$_2$ and VNS$_3$). Each s solution improved by the VNS algorithm is then memorized in a list L_m, where m corresponds the index of the current group. In step 4, the best solution x_{best} is selected from the memorized solutions obtained from all L_m lists. If the objective function of x_{best} is smaller than that of x^*_{best} ($f(x_{best}) < f(x^*_{best})$), where x^*_{best} is the global best solution, x_{best} becomes the new global best solution. In step 5, the s new solutions obtained after applying VNS on each group are migrated to the next population. To complete the population Pop, new $(Pop\text{-}s)$ solutions are generated in step 6. These two steps (5 and 6) are based on the well-known Bees Algorithm (BA) metaheuristic of Pham et al (2005). The main advantage of step 5 is that it permits the best solutions to survive to the next EVO-VNS iteration, while the purpose of step 6 is to increase the diversification of the search in the next EVO-VNS iteration as well.

Algorithm 1: The evolutionary variable neighborhood search algorithm

```
Initial population: Generate the initial population $Pop$ using a set of construction heuristics; $f(x^*_{best}) = \infty$;
    Repeat
        Step 1: Sort the solutions in descending order according to their fitness;
        Step 2: Divide the population $Pop$ into $m$ groups, each group contains $c$ solutions;
        Step 3: For each group $m$ DO
            Step 3.1: Select $(s)$ solutions using roulette wheel selection from the $c$
            solutions;
             If no improvement of the best solution of the current group after three
             consecutive iterations Do
                 Perform the crossover operator between one solution from $s$ and one
                 solution from $(c - s)$ of the current group;
                 Replace the $(s)$ solutions with the $(s)$ new solutions in the current
                 group;
             End If
            Step 3.2: Perform VNS on each of the $s$ solutions and memorize the new
            solution in list $L_m$;
        Step 4: Select the best memorized solution $x_{best}$ from the lists $L_m$;
             If $f(x_{best}) < f(x^*_{best})$ Then
                 $x^*_{best} \leftarrow x_{best}$
        Step 5: Insert the $s$ new solutions in the population;
        Step 6: Generate $Pop - s$ new solutions to complete the population $Pop$;
    Until No improvement has been achieved after five consecutive iterations
Return: $x^*_{best}$
```

5.1. Initialization phase

For the initialization phase, we use a modified version of the insertion heuristic proposed by Braekers et al. (2014) to insert the users as in the DARP. In addition, we consider the replacement of the battery by a new full one if it is depleted.

We first initialize a list L with a set of users to be served. Then, the following steps are performed. A user i is randomly selected from the list L and inserted in one of the already existing available vehicles in the best position that respects time windows, ride time and a maximum route duration. If a user i cannot be inserted in the route due to violation of the battery level, the selected user is re-inserted together with a BSS node f. This is done by selecting the nearest f node to the already existing node and inserting it between the previously inserted node of user i-1 and the current user i. This insertion procedure is applied until all users are served.

5.2. Three variants of the variable neighborhood search algorithm

As defined by Mladenovic and Hansen (1997), the VNS algorithm can be broken down into two phases: a deterministic phase, in which a local search converges to a local optimum, and a stochastic phase put in place to escape the local optima. As in the standard VNS, the stochastic part (called shaking phase) of the algorithm consists of generating a new solution x' in a given neighborhood. As for the descent with variable neighborhood, the search uses an initial solution x as a starting point, and uses a set of N_h neighborhoods h = $\{1, \dots, h_{max}\}$. At each iteration, a random solution x' is generated from the current neighborhood N_h. Then, a local search is applied to x' which generates a new solution x''. If this new solution x'' is better than x', an update is performed, and the process resumes with the first neighborhood. Otherwise, the same steps are repeated after selecting the next neighborhood $h+1$. The algorithm returns x_{best} when the number of n_{vns} iterations is reached.

In our study, we propose three different enhanced VNS variants due to many characteristics and features added compared to the traditional VNS in the literature, with the aim to achieve efficient solutions to our problem. We describe these variants in the following subsections.

5.2.1 Variable neighborhood search with roulette-wheel selection VNS$_1$

In this section, we present the first version of our VNS (denoted VNS$_1$), as described in Algorithm 2. First, both the current and best solutions are initialized with the selected solution from the current group of the population, or obtained after crossover (if done). The algorithm runs for n_{vns1} times. At each iteration, a new solution x' is generated from x using the current neighborhood N_h. In this regard, and instead of choosing the route pairs randomly as in the majority of studied VNS approaches in the DARP (see., e.g., Parragh et al., 2009; Parragh et al., 2010; Parragh, 2011; Schilde et al., 2011, 2014; Muelas et al., 2013; 2015), in our VNS$_1$ the routes chosen in all neighborhood structures are selected by the roulette wheel method, for the exclusion of several infeasible exchange operations (Polat et al., 2015; Hof et al., 2017).

The solution x' is improved by our local search strategy, which contains four local search operators {I1, I2, I3 and I4} followed by the Insert Battery-Station (IBS) and Remove Battery-Station (RBS) operators. The last two operators are needed because the solution may become infeasible due to insufficient battery level, after applying neighborhood and local search operators to obtain a new solution x''. To select the new solution x'', we adopt a first improvement strategy, where all possible neighboring solutions are generated using the current local search operator until the first improving solution is found. Otherwise, the next local search operator, in order, is performed. If no improvement is obtained after all local searches, the procedure stops and outputs the current solution.

One of the main characteristics of our VNS$_1$ is that the order of the local search operators (i.e., I1, I2, I3 and I4) is determined based on their performance score at the previous iteration, instead of a random order. For the selection process, we use the roulette wheel selection procedure of the ALNS. We note that in the beginning of the local search procedure, the order of the local search operators is generated randomly. If the new solution x'' is feasible and is better than the current solution, it is accepted. On the other hand, if the new cost is higher than the current cost, the new solution may be accepted subject to a given probability distribution of Cauchy function proposed by Tiwari et al. (2006): $T_i/(T_i^2 + \Delta E^2)$, where T_i is the current temperature and ΔE represents the difference in cost between the current and the adjacent solution. The temperature is decreased with the cooling scheme: $T_i = \delta * T_{i-1}$, where δ is the cooling rate, and i is the iteration number.

Following Tiwari et al. (2006) and Lin and Vincent (2015), Cauchy probability function is efficient and gives more opportunities to escape from local optima than the traditional Boltzmann function of Metropolis et al. (1953), which is the acceptance criterion applied in most VNS algorithms. If the obtained solution is better than the best solution (x_{best}), x_{best} is updated, x'' becomes the current solution x and the procedure resumes with the first neighborhood. Otherwise, the next neighborhood stucture is applied.

Algorithm 2: Variable neighborhood search algorithm with roulette-wheel selection (VNS$_1$)

```
Initialize A set of neighborhood structures $N_h$, where $h = \{1, \dots, h_{max}\}$; $x_{best} = x$ = the current initial solution $s$;
and $T_i = T_{max}$
Repeat
        $h \leftarrow 1$;
        Repeat
                Generate a new solution $x'$ from the $h^{th}$ neighborhood of $x$ ($x' \in N_h(x)$);
                Apply local search procedure on $x'$ to obtain $x''$;
                If $f(x'') \leq f(x)$ or accepted by the acceptance criterion Then
                        $x \leftarrow x''$;
                        $k \leftarrow 1$;
                        If $f(x'') < f(x_{best})$ then
                                $x_{best} \leftarrow x''$;
                        End If
                Else
                        $h \leftarrow h+1$;
                End if
                Update the weights of operators;
         Until $h = h_{max}+1$
         $T_i = \delta * T_{i-1}$;
Until The number of iterations $n_{vns1}$
Return $x_{best}$
```

5.2.2 Variable neighborhood search with selected local search VNS$_2$

The structure of the second VNS (denoted by VNS$_2$) is similar to Algorithm 2. The differences are detailed as follows. The first difference is that, after obtaining a new solution x', n_{loc} iterations is applied on the current x'. At each n_{loc} iteration, only one local search operator from {I1, I2, I3 or I4} is selected based on its performance. Thus, this procedure helps to better exploit the movements of the current selected local search structure. The selected operator is applied on x' followed by the IBS and RBS operators, resulting in a new solution x''. After this step, if the objective function of x'' is better than that of x', x'' replaces x'. After n_{loc} iterations, a new x_{best} is updated if found.

The final difference is in the acceptance function. In VNS$_2$, we use the acceptance function of the Record-to-Record Travel (RRT) of Dueck (1993). Specifically, if the objective function of x'' is less than that of the current value Rec minus the deviation Dev ($Rec - Dev$), where Rec is the objective function value of the best solution observed so far and Dev is equal to 0.01*Rec, the new solution is accepted. During the process of search, the Rec value is updated based on the objective function of x''. The detailed steps of our VNS$_2$ are shown in Algorithm 3.

Algorithm 3: Variable neighborhood search with selected local search (VNS$_2$)

```
Initialize A set of neighborhood structures $N_h$, where $h = \{1, \ldots, h_{max}\}$; $x_{best}$ =$x$ = the current initial solution $s$;
Repeat
        $h \leftarrow 1$;
        Repeat
                Generate a new solution $x'$ from the $h^{th}$ neighborhood of $x$ ($x' \in N_h(x)$);
                $c$=1;
                While ($c \leq n_{loc}$)
                        Perform a local search operator from {I1, I2, I3 or I4} on $x'$ to obtain $x''$;
                        Perform IBS and RBS on $x''$;
                        If $f(x'') \leq f(x')$ or accepted by the acceptance criterion Then
                                $x' \leftarrow x''$;
                End While
                If $f(x') < f(x_{best})$ then
                        $x_{best} \leftarrow x'$;
                        $h \leftarrow 1$;
                Else
                        $h \leftarrow h+1$;
          Update the weights of operators;
          Until $h$=$h_{max}$+1
Until The number of iterations $n_{vns2}$
Return $x_{best}$
```

5.2.3 Variable neighborhood search with descend acceptance VNS$_3$

The final proposed VNS variant is named as VNS$_3$. The following distinguishes VNS$_3$ from the two previous VNS algorithms. First, instead of generating only one point selected randomly in the current neighborhood structure N_h, the procedure generates many random neighboring solutions from the current neighborhood structure. More specifically, for each neighborhood N_h, n_{neig} iterations are applied, such that in each iteration a random neighboring point $x'$of the current solution is generated. The number of iterations n_{neig} is assumed to be equal to the number of vehicles in each instance. Then, the new solution x' is improved by the local search strategy.

As pointed by Wei et al. (2015), this technique permits to better explore the region of the selected neighborhood structure around the current solution. In addition, in VNS$_3$, we only accept a better solution (i.e., we follow a straightforward descend acceptance strategy without applying an acceptance function as done in VNS$_1$ and VNS$_2$). Thus, VNS$_3$ follows the original VNS of Mladenovic and Hansen (1997). Moreover, the selection of routes is also done in a random way in our VNS$_3$. The framework of the proposed VNS$_3$ is shown in Algorithm 4.

Algorithm 4: Variable neighborhood search algorithm with descend acceptance (VNS$_3$)

```
Initialize A set of neighborhood structures $N_h$, where $h = \{1, \ldots, h_{max}\}$; $x_{best}$ =$x$ = the current initial solution $s$;
```

```
Repeat
    $h \leftarrow 1$
    Repeat
        Repeat
            Generate a new solution $x'$ from the $h^{th}$ neighborhood of $x$ ($x' \in N_h(x)$);
            Perform the local search strategy on $x'$ to obtain $x''$;
            If $f(x'') < f(x_{best})$ then
                $x \leftarrow x''$;
                $x_{best} \leftarrow x''$;
            Else if $f(x'') < f(x)$
                $x \leftarrow x''$
        Until $n_{neig}$ iterations is reached;
        $h \leftarrow h+1$;
    Until $h = h_{max}$
Until The number of iterations $n_{vns3}$
Return $x_{best}$
```

5.2.4. Neighborhood search operators

During each step of the VNS algorithms, several well-known neighborhood search operators inspired and adopted from the literature are applied in the phase of shaking. According to Hemmelmayr et al. (2009), the neighborhood move is an important phase and should balance between perturbation and conservation of the good parts in the current solution. In this regard, four effective neighborhood search structures (i.e., N1, N2, N3 and N4) with different movements are developed as described below.

As Mladenović and Hansen (1997) pointed, the order of neighborhood search operators is very critical in VNS. Usually neighborhood structures are chosen such that the size of the neighborhood increases as VNS progresses from one structure to the next. Thus, at the initial stages {N1, N2}, small changes are performed around the incumbent solution. Moving further from the current solution {N3, N4} is only done if the small changes were ineffective. As a result, we follow the following order of neighborhood structures N1, N2, N3 and N4.

Swap (N1): This operator is inspired from Braekers et al. (2014). The swap operator consists of swapping either two requests or two vehicles belonging to two different routes. To apply this operator, we first select one route. Then, two types of swaps are considered: swapping any user request in this route with a user request selected from another route, or swapping a vehicle in this route with another vehicle from a different route.

Remove-sequence (N2): This neighborhood operator is inspired from Parragh et al. (2010) while taking into account the features of the DARP-EV. First, this operator selects two routes. Thereafter, a sequential range y= {1, 2, 3, 4 or 5} (based on the number of users in the route) of successive users from the route is selected randomly. The chosen sequences (including the BSS node f, if found) are then removed from their current route and re-inserted one after another at their best positions in other routes. It is observed in each range that the pickup and delivery of a user must be chosen. When the user is reinserted, the algorithm examines all potential combinations of insertion positions of pickup and delivery nodes in the selected route. If there is no possible feasible insertion, the one that has the lowest cost is chosen as in Muelas et al. (2013) and inserted in any position.

Cross-exchange (N3)-(N4): This neighborhood is proposed by Osman (1993), where b consecutive users are transferred moving from one specific route (route 1) to another different one (route 2). Subsequently, d

consecutive users are carried from route 2 to route 1. The random selection of b is bounded between 2 and 3, and d is chosen at random equivalent to b or b-1. As suggested in Hemmelmayr et al. (2009), this procedure is conducted to obtain more diversification of the search. Furthermore, as long as the segment distance is lengthier, the chance to have an effective swap is minimized. In the course of this study, the segment length is restricted between 2 (**N3**) and 3 (**N4**). Considering its capacity of achieving an effective diversification, this version of neighborhood move is often applied in the perturbation stage (see, e.g., Polacek et al., 2004; Hemmelmayr et al., 2009).

5.2.5. Local search operators

The local search operation is applied in order to intensify the search around the solution obtained from neighborhood structures. Regarding our local search, it is composed of two intra-route operators (Relocate and 4-opt) and two inter-route operators (2-opt* and remove-two-insert-one), which are capable of enhancing the solution in a rapid and successful way. A detailed description of the operators is provided below.

2opt* (I1): This operator is proposed by Potvin and Rousseau (1993). Two arcs from distinct routes are deleted so as to disconnect each route into two segments. Thereafter, each first segment of one particular route is linked with the last segment of the other route, so as to come up with two new routes.

Remove-two-insert-one (I2): This operator is adopted from Xiang et al. (2006). It consists of removing two randomly selected users from a vehicle and then trying to insert them one by one in another vehicle in their best position.

Relocate (I3): This operator is similar to the previous operator but it is applied in the same vehicle. This operator is applied on each user in the vehicle by removing the user and reinserting it in the best possible position. In this case, three types of moves of relocating a user i are considered; the first, consists of relocating only the pickup node of the selected user i. The second is for the delivery node of user i, while the third consists of removing a user i and then inserting the delivery node of this user immediately after its pickup node.

4-opt (I4): This operator is adopted from Braekers et al. (2014), and has shown its effectiveness to improve solutions in the field of DARP. This operator consists of selecting four consecutive arcs to be deleted from only one selected route. In other words, the order of the three successive nodes can change in the route. We note that this operator is applied on each set of four consecutive arcs in the selected route.

The probability of choosing operator d at iteration t, is calculated using the roulette wheel mechanism as in Ropke and Pisinger (2006): $P_d^{t+1} = P_d^t(1- r_p) + r_p\pi_i/\omega_i$, where r_p is the roulette wheel parameter, π_i is the score of the operator i, and ω_i is the number of times that the operator i has been used. The score of an operator is increased by π_1 if the operator finds a new best solution, otherwise, it is enhanced by π_2 if it locates a better solution than the current one. On the other hand, it is increased by π_3 if the current selected operator finds an approved solution that is non-improving. After n_{seq} iterations, the new weights are adjusted using the obtained scores.

After applying neighborhood and local search moves, there is a possibility of having some unnecessary BSS nodes in a solution. In addition, the current solution may require adding other BSS node(s). In order to avoid having any of these situations, two operators are proposed.

Remove Battery-Station (RBS): This operator is intended to examine each pair of nodes (i, j). The BSS node is removed if the level of charge in the battery at node i is sufficient to reach node j.

Insert Battery-Station (IBS): An important factor in the context of EVs, is to determine the latest time at which recharging the vehicle has to take place, in order to prevent it from getting stranded due to lack of energy (Schneider et al., 2014; Goeke and Schneider, 2015). Thus, inspired from Liao et al. (2016), the proposed idea of the IBS operator is that if the remaining charge level in the battery of the vehicle k at node i is not sufficient to directly reach node j, an insertion of a BSS node f is performed. This is done by locating the nearest f node($\forall f \in F$) to the current node j to do a battery swap.

5.2.6. Chromosome encoding

In our EVO-VNS, an MX1 crossover operator is applied leading to the necessity for encoding a solution. A chromosome (solution) is encoded using a sequence of available vehicles v_k, each starts its trip from the depot (denoted by 0) and returns back to the same depot. For each vehicle route, we assume an ordered list of pickup and drop off nodes as well as the visited BSS nodes F_f (if found). To encode the pair of nodes for each user i, h_i^+ and h_i^- represent the pickup and delivery nodes of this user, respectively. So, for example, if a solution has two routes (i.e., two vehicles), eight requests and one BSS node, the chromosome encoding will be as follows: 0,1^+,1^-,4^+,5^+,4^-,5^-,6^+,F_1,2^+,5^-,6^-,2^-,0 for vehicle v_1 and 0,3^+,7^+,8^+,3^-,8^-,7^-,0 for vehicle v_2. Figure 3 shows an example of a solution of these two routes.

V_1	0	1^+	1^-	4^+	5^+	4^-	5^-	6^+	F_1	2^+	5^-	6^-	2^-	0
V_2	0	3^+	7^+	8^+	3^-	8^-	7^-	0						

Fig.3: An example of chromosome encoding

5.3. Generation of new solutions

To complete the population Pop, new $(Pop - s)$ solutions should be generated using a new heuristic. Our heuristic is based on destroy-reinsert users. First, s users are selected from the current solution and put in a list L. Second, a 2-regret insertion heuristic as in Ropke and Pisinger (2006) is applied to re-insert the temporarily deleted users. For each request i in L, Δf_i^p the insertion cost of the request i, is identified in the best route as well as at its best position. At each iteration, the request i^* is chosen to be inserted in its best position according to the following formula $i^* \coloneqq \arg\max_{i \in L}(\sum_p^k \Delta f_i^p - \Delta f_i^1)$. In case there is no possibility to incorporate more users in a route, or otherwise if all requests are inserted, the heuristic stops.

As Pisinger and Ropke (2007) indicated, deleting a large number of users in the removal phase may have a considerable effect on the results. Accordingly, we applied the following technique to select a number of users u. If the number of users in the instance is less than 50 users, u is selected randomly between five and 10. Otherwise, u is chosen randomly between five and 20.

5.4. Evaluation function

In each step of our algorithms, whenever a new solution is generated, it must be evaluated by the evaluation function in terms of feasibility and cost. We evaluate the solution by the following evaluation function based on Cordeau and Laporte (2003) as $f(x) = c(x) + \sum_{r=0}^{3} \alpha q_r(x) + \beta d(x) + \gamma w(x) + \tau a(x) + o(x)$. The term $c(x)$ gives the routing costs of solution x. Variables $q_r(x), d(x)$, $w(x)$, $a(x)$ and $o(x)$ denote violations of the following constraints in order: vehicle load, route duration, time windows, maximum ride time and the battery state of the vehicle. Following Parragh et al. (2010) and Cordeau and Laporte (2003), the first four violations are calculated as follows: $q_r(x) = \sum_{i=1}^{2n}(Q_i^{rk} - Q^{rk})^+$, $d(x) = \sum_{k=1}^{K}(B_{2n+f'+1}^k - B_0^k - T_{max})^+$, $w(x)$=$\sum_{i=1}^{2n}(B_i^k - T_i^+)^+$ and $a(x) = \sum_{i=1}^{n}(L_i^k - L_{max})^+$. Note that these terms are applied only for all $i \in N$ where $x^+ = \{0, x\}, \forall\, k \in K$. Besides the previous constraints, the level of the battery in the vehicle must be respected in the solution. $z_{bat}^k(i)$=$z_{bat}^k(i-1)$-$ECc_{i-1,i}^k$, if $i \in V'$ and $z_{bat}^k(i)$=H, if $i \in F'$. Binary term $o(s)$ is equal to 0 if $z_{bat}^k(i) \geq 0, \forall\, i \in V', \forall\, k \in K$ and 1 otherwise. The penalty parameters $(\alpha, \beta, \gamma \text{ and } \tau)$ are dynamically adjusted during the search as in Cordeau et al. (2001), i.e., if the current solution respects the vehicle load constraint, α is divided by 1+δ, where δ is a uniform number generated randomly between 0 and 0.5; otherwise, α is multiplied by 1+δ. The same penalty procedure is applied to parameters $\beta, \gamma \text{ and } \tau$.

It should be noted that for a solution x to become a new best solution, we must have $q_r(x)$= $d(x)$= $w(x)$= $a(x) = o(x)$=0, for each resource r =0, 1, 2, 3.

In order to evaluate a route, we follow the adapted eight-step evaluation procedure of Parragh et al.(2010) that is proposed by Cordeau and Laporte (2003), as shown in Algorithm 5. The algorithm applies the forward time slack concept of Savelsbergh (1992), proposed for the VRP, after adapting it to the DARP. The forward time slack SL_i for a node $i \in N$ is calculated as:l_j^k

$$SL_i = \min_{i \leq j \leq y} \{\textstyle\sum_{i \leq p \leq j} W_p + (\min\{l_j^k - B_j^k, L_{max} - P_j\})^+\}$$

Where $j \in \{n+1, \ldots, 2n\}$ is the destination of user i, and y is the last node in the route. W_p is the waiting time at node p, and P_j represents the ride time of the user from i to j, given that $j-n$ is visited before i on the route; P_j=0 for all other j.The forward time slack SL_i represents the maximum amount of time that the departure of the vehicle from node i can be delayed, without causing a violation in the time window and maximum ride time constraints.

Algorithm 5: The eight-step evaluation scheme

```
1. Set departure time $D_0^k$:= $e_0$
2. Compute beginning of service ($B_i^k$), departure time ($D_i^k = B_i^k + s_i$), battery level ($z_{bat}^k(i)$); arrival time ($A_i^k$), waiting time ($W_i^k = B_i^k - A_i^k$), and load of vehicle ($Q_i^{rk}$) for each node $i$ along the route
    If some $B_i^k$> $l_i^k$, or $z_{bat}^k(i) < 0$, or $Q_i^{rk}$ > $Q^{rk}$, Go to step 8
3. Compute $SL_0$
4. Set $D_0^k$ := $e_0$+min{$SL_0$, $\sum_{0<p<y} W_p$}
5. Update $A_i^k$, $W_i^k$, $B_i^k$ and $D_i^k$ for ech node on the route
6. Compute $l_i^k$ for each request on the route
    If all $l_i^k \leq L_{max}$ Go to step 8
7. For evry node $j$ that is an origin
    (a) Compute $SL_j$
    (b) Set $W_j^k$:= $W_j^k$+ min{$SL_j$, $\sum_{j<p<y} W_p$}; $B_j^k$:= $D_j^k$+ $t_{ij} + W_j$; $D_j^k$:= $B_j^k$+ $s_j$
    (c) Update $A_i^k$, $W_i^k$, $B_i^k$ and $D_i^k$ for each node that comes after $j$ in the route
    (d) Update $L_i^k$ for each request $i$ whose destination is after $j$
    If all $L_i^k \leq L_{max}$ of requests whose destinations lie after $j$, Go to step 8
```

8. Compute changes in violation and calculate $f(x)$

5. Computational experiments

This section presents the detailed numerical results obtained by our proposed algorithms. All implementations are done using C programming language on a configuration of Intel Core i7-5555U 3.14 GHz and 8 GB RAM.

5.1. Data and experimental setting

To test our algorithms, we use three sets of data: small-, medium- and large-sized instances. All our instances contain heterogeneous user types and are generated similar to the instance generation of Masmoudi et al. (2017). There are three sets (U, E, I), which are modified versions of the instances created by Parragh (2011) and Braekers et al. (2014) for the small- and medium-sized instances, and from Cordeau and Laporte (2003) for the large-sized instances. For the small and medium instances, these instances include up to four vehicles and 96 requests. The time windows of users on small and medium instances are set to 15 minutes. The maximum user ride time (L_{max}) and service time (s_i) at each location are set to 30 minutes and three minutes, respectively.

For the large-sized instances (20 instances), these instances contain up to 144 requests and 13 vehicles. The service time is equal to three minutes for each user, and the transportation time in minutes is assumed to be equal to the Euclidean distance between any locations. The maximum daily working time for each vehicle has a limit of 480 minutes, while the maximum ride time is 90 minutes.

In the first set of instances (R1a-R10a), the time windows range between 15 and 45 minutes. In the second set of instances (R1b-R10b), the time windows are set to be between 30 and 90 minutes. The capacity of the vehicle contains four types of resources: a staff seat, a patient's seat, a stretcher, and a wheelchair place. To generate the instances (U, E, I), Masmoudi et al. (2017) assumed certain probabilities of patients' requesting facilities and companions as shown in Table 3. The adopted and generated instances with their detailed results can be downloaded from the http://www.ddarp-ev-73.webself.net.

Table 3
Probabilities used to generate instances by Masmoudi et al. (2017)

Instance set	Patient request probabilities			Probability for a companion (%)
	Seat (%)	Stretcher (%)	Wheelchair (%)	
U	0.50	0.25	0.25	0.00
E	0.25	0.25	0.50	0.10
I	0.83	0.11	0.06	0.50

We assume that the vehicle capacity resources include one staff seat, six patient seats, one wheelchair place, and one stretcher.

To complete our DARP-EV instance sets regarding the number of BSSs, we calculate the number of stations adaptively according to the number of users in each instance. Precisely, for n users, the number of recharging stations is assumed to be 0.1*$|n|$ as done in Goeke and Schneider (2015). The coordinates of BSS nodes are randomly generated in a specific square area (i.e., $[-10, 10]^2$). We assume that the initial depot is considered also as a BSS node. Thus, the BSS nodes (together with the depot) are set along the route, such that the vehicle departs from the depot with a full-battery charge.

5.2. Parameter settings

Before testing our algorithms, it is essential to do extensive experiments to obtain the best parameter values. Therefore, we have chosen the parameters based on recommendations from the literature or by making preliminary experiments to obtain a good tradeoff between solution quality and computational time. A summary of all parameters used in our algorithms are shown in Table 12 in Appendix A.

Concerning the VNS$_1$ algorithm, the initial temperature value T_{max} was set to 100 and the cooling rate δ equal to 0.99975, as suggested by Ropke and Pisinger (2006) and Demir et al. (2012). For VNS$_2$ and VNS$_3$, we have n_{loc} = n_{neig} = number of vehicles in each instance, respectively. For VNS$_1$, VNS$_2$ and VNS$_3$, the number of iterations is denoted as n_{vns1}, n_{vns2} and n_{vns3}. On the other hand, the overall EVO-VNS algorithm outputs the best solution when no improvement after five consecutive iterations. Since we have additional diversification procedures, our parameters π_1, π_2 and π_3 for VNS$_1$ and VNS$_2$ are set equal to 15, 10 and 5, respectively. The adjustment parameters have been set as $\pi_1 \geq \pi_2 \geq \pi_3$ to reward an operator for good performance, as adopted values from Masmoudi et al. (2016).

Regarding the hybrid variants of VNS (i.e., the three EVO-VNS variants), to obtain the required good parameters of the hybridization, we tune our parameters similar to the procedure followed by Goeke and Schneider (2015) and Masmoudi et al. (2016). We first identified the parameters that we believe have a strong effect on solution quality, namely, the number of groups $\boldsymbol{m}$, the number of solutions in each group $\boldsymbol{c}$, and the number of solutions selected based on roulette wheel selection $\boldsymbol{s}$. Then, we identified for each of these parameters a base setting that has good performance. After this, we tested different settings for each parameter, and then we kept the best setting found among them, while we tuned the rest of the parameters. The order of tuning the parameters is taken randomly. Table 4 and its detailed results in the website shows the different settings tested for each parameter, and the deviation Best%(Avg%) from the best solution found in five runs, using this setting, compared to the result obtained using the best setting for the same parameter. The analysis is done on a set of instances, which contain various levels of heterogeneity of users, and the requests vary from small to large. We highlight the best setting for each parameter in bold.

In more details, the following method is applied to tune $\boldsymbol{m}$, $\boldsymbol{c}$ and $\boldsymbol{s}$. Firstly, we set the size of each group population $\boldsymbol{m}$. Secondly, for each $\boldsymbol{m}$ value, we assess the effectiveness of different combinations of the pair ($\boldsymbol{c}$, $\boldsymbol{s}$). In this regard, we note that several diversification mechanisms and components of the different evolutionary algorithms that we have incorporated in our VNS variants will need to run many times during an iteration, which is computationally expensive. Thus, we have chosen a small population size ($\boldsymbol{Pop}$), where the tested number of groups $\boldsymbol{m}$ is equal to 2, 3 and 4, the tested number of solutions in each group $\boldsymbol{c}$ is equal to 4, 5, 6, 7 and 8, and the tested number of solutions selected based on roulette wheel selection $\boldsymbol{s}$ is equal to 2 and 3.

Table 4
Identification of the best parameter setting for the hybrid EVO-VNS

m	2						**3**						4					
(*c*, *s*)	(5;2)	(6;2)	(4;2)	(6;3)	(7;3)	(8;3)	(4;2)	(5;2)	**(6;2)**	(6;3)	(7;3)	(8;3)	(4;2)	(5;2)	(6;2)	(7;2)	(8;2)	(6;3)
Best	733.84	733.43	733.19	732.98	733.04	732.74	732.33	732.33	**732.33**	732.33	732.30	732.33	732.32	732.32	732.32	732.32	732.25	732.24
Best%	0.22	0.16	0.13	0.10	0.11	0.07	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00
Avg	739.14	737.46	736.28	735.77	735.63	735.16	734.97	734.00	**733.46**	733.40	733.40	733.49	733.28	733.34	733.12	732.91	733.06	732.32
Avg%	0.22	0.16	0.13	0.10	0.11	0.07	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00

CPU (min)	4.92	4.33	3.35	3.20	3.63	3.74	5.35	5.68	**5.27**	7.46	8.04	8.60	9.68	10.42	12.13	12.25	13.67	14.79

As shown in Table 4 and its detailed results in the website, the parameter values of m, c and s considerably affect the solution quality. As indicated by the results, the quality of average and best solutions shows a slight improvement when using m=4 with differents values of c and s, and also require an additional computational time. The best parameters are chosen based on obtaining a high quality solution in a short CPU time. Thus, the following parameter values were finally selected, m=3, c=6 and s=2 for all our algorithms, in order make a fair comparison to evaluate their performance.

Finally, we note that the number of iterations of each evolutionary algorithm is only set to 1,000 iterations (i.e., n_{vns1}=n_{vns2}=n_{vns3}=1,000). This reduction was intended to reduce the computational time of our evolutionary algorithms.

5.3. *Computational analysis*

This section presents the detailed results obtained by our algorithms after testing them not only on the generated instances of the DARP-EV, but also on the benchmark DARP instances of Masmoudi et al. (2017).

5.3.1. Results on the DARP instances of Masmoudi et al. (2017)

To further prove the efficiency of our evolutionary algorithms, we used the benchmark DARP instances with heterogeneous users of Masmoudi et al. (2017). If we assume a fleet of conventional vehicles, with each vehicle having full tank capacity instead of EVs, the DARP-EV can be easily transformed to the DARP.

In order to compare with the hybrid Genetic Algorithm (GA) of Masmoudi et al. (2017), each algorithm (including the hybrid GA), was run on every instance five times, as also done in Masmoudi et al. (2017). For each table in this subsection, column “BKS” represents the best known solutions. The column “Best%” (“Avg%”) indicates the percentage of deviation from the optimal solution (best known solution) in small-medium (large) instances, and the computation time in minutes is symbolized by “CPU”. The detailed results of these tables are shown in the website. We note that, the negative percent deviations in EVO-VNS$_1$, EVO-VNS$_2$ and EVO-VNS$_3$ algorithms indicate an improvement in solution with respect to the best value found by the hybrid GA. Tables 5 and 6 show the results obtained for the small-medium instances and the large instances, respectively.

It should be noted that we cannot compare the performance of the algorithms with respect to the computational time with that reported in Masmoudi et al. (2017) for the sake of a fair comparison with the previously published method. This is because a different machine has been used to run our algorithms than that used for the hybrid GA of Masmoudi et al. (2017). In addition, estimating the speed factor of the configuration applied in Masmoudi et al. (2017) as well as that of our machine is not possible by using Dongarra (2014) table, since no relevant information is reported in Dongarra (2014) and in Linpack (2016). In addition, as mentioned in Masmoudi et al. (2017), the computational power of MFlops and the speed factor of the configuration applied for the hybrid GA are not known. Thus, in Tables 5 and 6 the computational time is reported only for the record, and is not intended for an accurate comparison with the previously published methods. In general, though, our algorithm is run within a reasonable computational time.

Table 5

Comparison of the hybrid GA and our algorithms on small-medium instances

Inst.	BKS[*]	Hybrid GA			EVO-VNS$_1$			EVO-VNS$_2$			EVO-VNS$_3$		
		Best %	Avg %	CPU (min)	Best %	Avg %	CPU (min)	Best %	Avg %	CPU (min)	Best %	Avg %	CPU (min)
a2-16	331.16[*]	0.00	0.00	0.23	0.00	0.00	0.34	0.00	0.00	0.35	0.00	0.00	0.37
a2-20	347.03[*]	0.00	0.00	0.47	0.00	0.00	0.85	0.00	0.00	0.88	0.00	0.00	0.99
a2-24	450.25[*]	0.00	0.00	0.38	0.00	0.00	0.65	0.00	0.00	0.68	0.00	0.00	0.75
a3-18	300.63[*]	0.00	0.00	0.23	0.00	0.00	0.39	0.00	0.00	0.41	0.00	0.00	0.45
a3-24	344.91[*]	0.00	0.00	0.35	0.00	0.00	0.52	0.00	0.00	0.54	0.00	0.00	0.57
a3-30	500.58[*]	0.00	0.00	0.41	0.00	0.00	0.72	0.00	0.00	0.77	0.00	0.00	0.91
a3-36	583.19[*]	0.00	0.00	0.57	0.00	0.00	1.13	0.00	0.00	1.19	0.00	0.00	1.40
a4-16	285.99[*]	0.00	0.00	0.20	0.00	0.00	0.33	0.00	0.00	0.34	0.00	0.00	0.39
a4-24	383.84[*]	0.00	0.00	0.29	0.00	0.00	0.50	0.00	0.00	0.53	0.00	0.00	0.59
a4-32	500.24[*]	0.00	0.00	0.55	0.00	0.00	0.91	0.00	0.00	0.96	0.00	0.00	1.06
a4-40	580.42[*]	0.00	0.00	0.58	0.00	0.00	1.02	0.00	0.00	1.07	0.00	0.00	1.21
a4-48	670.52[*]	0.00	0.00	0.81	0.00	0.00	1.35	0.00	0.00	1.41	0.00	0.00	1.62
a5-40	500.06[*]	0.00	0.00	0.52	0.00	0.00	0.99	0.00	0.00	1.05	0.00	0.00	1.24
a5-50	693.77[*]	0.00	0.00	0.67	0.00	0.00	1.15	0.00	0.00	1.21	0.00	0.00	1.37
a5-60	828.90[*]	0.00	0.00	0.97	0.00	0.00	1.61	0.00	0.00	1.68	0.00	0.00	1.87
a6-48	614.36[*]	0.00	0.00	0.70	0.00	0.00	1.48	0.00	0.00	1.56	0.00	0.00	1.82
a6-60	847.58[*]	0.00	0.06	1.00	0.00	0.06	1.55	0.00	0.06	1.59	0.00	0.08	1.70
a6-72	949.17[*]	0.00	0.03	1.33	0.00	0.00	2.25	0.00	0.00	2.31	0.00	0.00	2.46
a7-56	740.63[*]	0.00	0.05	0.79	0.00	0.00	1.55	0.00	0.00	1.66	0.00	0.05	1.97
a7-70	946.32[*]	0.00	0.08	1.12	0.00	0.05	1.87	0.00	0.06	1.96	0.00	0.07	2.19
a7-84	1092.90	0.00	0.09	1.37	0.00	0.08	2.71	0.00	0.08	2.87	0.00	0.09	3.33
a8-64	762.81	0.00	0.03	0.93	0.00	0.13	1.56	0.00	0.15	1.62	0.00	0.12	1.77
a8-80	982.71	0.00	0.06	1.38	0.00	0.02	2.50	0.00	0.03	2.63	0.00	0.09	2.96
a8-96	1265.36	0.00	0.05	1.51	0.00	0.08	2.62	0.00	0.13	2.70	0.00	0.13	2.88
Avg	***645.97***	***0.00***	***0.05***	***0.72***	***0.00***	***0.02***	***1.27***	***0.00***	***0.02***	***1.33***	***0.00***	***0.03***	***1.49***

[*] *Optimal solutions provided by Braekers et al. (2014) with Branch and Cut (B&C) algorithm*
[b] *Results of Masmoudi et al.(2017), programmed in C and executed on 4 GHz Intel laptop with 1.86 GB RAM.*

Table 6
Comparison of the hybrid GA and our algorithms on large instances

Inst.	BKS[a]	Hybrid GA[b]			EVO-VNS$_1$			EVO-VNS$_2$			EVO-VNS$_3$		
		Best%	Avg%	CPU (min)	Best%	Avg%	CPU (min)	Best%	Avg%	CPU (min)	Best%	Avg%	CPU (min)
R1a	195.97	0.00	0.00	0.44	0.00	0.00	0.71	0.00	0.00	0.61	0.00	0.00	0.78
R2a	336.34	0.00	0.00	0.85	0.00	0.00	1.18	0.00	0.00	1.32	0.00	0.00	1.47
R3a	586.18	0.00	0.63	0.94	0.00	0.17	1.40	0.00	0.07	1.44	0.00	0.19	1.84
R4a	640.03	0.00	0.40	1.48	-0.16	0.36	2.21	-0.16	0.32	2.60	-0.16	0.50	2.59
R5a	714.83	0.00	0.51	1.82	-0.24	0.26	2.77	-0.20	0.28	2.94	-0.24	0.15	3.72
R6a	883.02	0.00	0.52	2.40	-0.10	0.02	4.16	-0.05	0.46	3.83	-0.08	0.09	4.48
R7a	312.05	0.00	0.29	0.47	-0.35	0.35	4.79	-0.35	0.24	5.74	-0.35	0.14	5.90
R8a	553.82	0.00	0.44	0.81	0.04	0.47	5.24	0.07	0.31	5.52	0.10	0.34	5.55
R9a	746.23	0.00	0.35	1.62	-0.21	0.30	7.33	-0.25	0.40	7.89	0.10	0.25	7.30
R10a	963.08	0.00	0.64	2.29	0.09	0.15	10.74	0.11	0.25	10.77	0.11	0.19	11.62
R1b	190.39	0.00	0.00	0.57	0.00	0.00	0.80	0.00	0.00	0.91	0.00	0.00	1.11
R2b	312.92	0.00	0.38	1.03	0.00	0.00	1.63	0.00	0.23	1.75	0.00	0.00	1.94
R3b	551.95	0.00	0.22	1.36	0.00	0.22	2.06	0.00	0.24	2.29	0.03	0.23	3.91
R4b	606.08	0.00	0.73	2.00	-0.13	0.40	2.95	-0.08	0.07	5.46	-0.05	0.41	3.41
R5b	641.84	0.00	0.05	2.93	-0.21	0.04	4.42	-0.21	0.04	5.70	-0.21	0.02	4.65
R6b	832.53	0.00	0.46	3.66	0.03	0.40	5.51	0.03	0.28	6.50	0.06	0.27	7.18
R7b	276.52	0.00	0.00	0.79	-0.13	0.03	1.21	-0.13	0.12	0.81	-0.13	0.16	1.21
R8b	530.56	0.00	0.32	1.53	-0.11	0.23	4.24	-0.11	0.13	4.41	-0.11	0.20	5.49
R9b	699.06	0.00	0.59	2.84	-0.13	0.19	6.54	-0.06	0.26	6.28	-0.02	0.33	6.91
R10b	902.17	0.00	0.53	3.11	0.12	0.22	11.71	0.15	0.38	11.11	0.11	0.73	12.61
Avg	***573.78***	***0.00***	***0.35***	***1.65***	***-0.07***	***0.19***	***4.08***	***-0.06***	***0.20***	***4.39***	***-0.04***	***0.21***	***4.68***

[a] *Best known solutions provided by Masmoudi et al.(2017)*
[b] *Results of Masmoudi et al.(2017), programmed in C and executed on 4 GHz Intel laptop with 1.86 GB RAM.*

Table 5 shows that our algorithms match the optimal solutions computed by the B&C of Braekers et al. (2014) for the small-medium instances. Looking at the average gaps of five runs, our EVO-VNS$_1$, EVO-

VNS$_2$ and EVO-VNS$_3$ algorithms obtain 0.02%, 0.02%, and 0.03% , respectively, compared to 0.05% of the hybrid GA. More importantly, Table 6 clearly shows that our algorithms are competitive with the hybrid GA method of Masmoudi et al. (2017) on large instances in terms of solution quality. Our algorithms improve the results by 0.07%, 0.06% and 0.04% on average. This points out to the steadiness of our evolutionary algorithms in terms of locating high quality solutions in most of the runs. In terms of the average deviation of the average results (calculated for five runs) from the best solutions of the hybrid GA, our algorithms achieved 0.19%, 0.20% and 0.21%, compared to 0.35% achieved by the hybrid GA. Overall, these results confirm that our proposed algorithms are competitive compared to the hybrid GA of Masmoudi et al. (2017) in terms of solution quality. This, however, comes at the expense of computational time, which is in fact more than twice that of the Hybrid GA of Masmoudi et al. (2017). This may be due to the new components that we added to our VNS, in order to enhance its exploration and exploitation powers, which has undoubtedly increased the overall computational time.

5.3.2. Results on our DARP-EV instances

In this section, we present the results on our generated instances of the DARP-EV. In each table in this section, columns “Best” and “Avg” present the best and average solution values, respectively. The column “%” following each of the “Best” (“Avg”) columns provides the percentage of deviation from the best solution value “BS” obtained by any of the three algorithms for a given instance. For the column “CPU”, it presents the average CPU time in minutes. Each instance is calculated five times by applying each algorithm. The results of our suggested methods on the small-medium and large instances are indicated in Tables 7 and 8, respectively.

Table 7
Comparison of our three algorithms on small and medium instances

Instance	BS	EVO-VNS$_1$					EVO-VNS$_2$					EVO-VNS$_3$				
		Best	%	Avg	%	CPU (min)	Best	%	Avg	%	CPU (min)	Best	%	Avg	%	CPU (min)
$\bar{U}$	668.51	668.74	0.02	668.89	0.03	2.52	668.96	0.05	669.57	0.12	2.60	669.09	0.06	669.66	0.12	3.04
$\bar{E}$	684.36	684.64	0.02	684.77	0.03	2.18	684.83	0.05	685.34	0.10	2.53	684.99	0.06	685.28	0.09	3.02
$\bar{I}$	668.15	668.38	0.02	668.69	0.05	2.68	668.50	0.03	668.94	0.08	3.31	668.84	0.07	669.33	0.13	3.60
$\overline{UEI}$	***673.67***	***673.92***	***0.02***	***674.12***	***0.04***	***2.46***	***674.10***	***0.05***	***674.62***	***0.10***	***2.81***	***674.30***	***0.06***	***674.76***	***0.11***	***3.22***

As seen from Table 7, there is a slight difference in our algorithms in terms of finding best solutions for most of the instances. In fact, the EVO-VNS$_1$ algorithm obtains the best solutions at least one time for 66 instances. On the other hand, EVO-VNS$_2$ obtains the best result only for 62 instances, while EVO-VNS$_3$ finds the best solutions for 57 instances. The three algorithms were capable of obtaining the same best solutions for 51 out of 72 instances. Regarding the average, the EVO-VNS$_1$, EVO-VNS$_2$ and EVO-VNS$_3$ algorithms have a gap equal to 0.02%, 0.05% and 0.06%, respectively, from the best solution for all instances. This articulates that the diversification and intensification mechanisms applied in our algorithms have considerable contribution in improving the solution quality.

Table 8
Comparison of our three algorithms on large-sized instances

Instance	BS	EVO-VNS$_1$					EVO-VNS$_2$					EVO-VNS$_3$				
		Best	%	Avg	%	CPU (min)	Best	%	Avg	%	CPU (min)	Best	%	Avg	%	CPU (min)

$\bar{U}$	605.76	606.08	0.04	608.59	0.40	9.00	606.20	0.08	610.08	0.63	10.16	606.59	0.11	610.14	0.63	10.62
$\bar{E}$	616.95	617.59	0.09	620.29	0.47	8.09	617.80	0.13	621.20	0.63	9.07	617.69	0.11	621.17	0.63	9.47
$\bar{I}$	610.85	611.51	0.08	614.32	0.50	10.58	611.48	0.08	614.89	0.55	11.97	611.31	0.08	615.04	0.63	12.52
$\overline{UEI}$	***611.18***	***611.73***	***0.07***	***614.40***	***0.45***	***9.22***	***611.82***	***0.10***	***615.39***	***0.60***	***10.40***	***611.86***	***0.10***	***615.45***	***0.63***	***10.87***

The results shown in Table 8 clearly indicate that our algorithms are able to gain good solution quality. EVO-VNS$_1$ obtains the best solutions for 49 instances, while EVO-VNS$_2$ can find the best solutions for 45 instances, and EVO-VNS$_3$ for 43 instances. Generally, in 28 cases, all methods are effective in obtaining the same best solutions. Taking the average values over five runs for each algorithm, the average of the best result deviates from the best solution by 0.07% for EVO-VNS$_1$, 0.10% for EVO-VNS$_2$ and 0.10% for EVO-VNS$_3$. On the other hand, the average gap percent is equal to 0.45% for EVO-VNS$_1$, 0.60% for EVO-VNS$_2$ and 0.63% for EVO-VNS$_3$. Generally, all three algorithms perform well and are able to obtain good-quality solutions for large instances.

To evaluate the effectiveness of hybridizing our VNS algorithms with different new components (population-based), we compared our evolutionary algorithms (EVO-VNS$_1$, EVO-VNS$_2$ and EVO-VNS$_3$) with non-hybrid variants of the same algorithms (i.e., VNS$_1$, VNS$_2$ and VNS$_3$).

Table 9
The contribution of our evolutionary algorithms compared to the standard algorithms

Instance (Data)	VNS$_1$					VNS$_2$					VNS$_3$				
	Best	%	Avg	%	CPU	Best	%	Avg	%	CPU	Best	%	Avg	%	CPU
Type-U (Small-Medium)	670.36	0.20	672.27	0.44	1.59	670.43	0.25	673.50	0.65	1.64	670.89	0.33	674.98	0.82	1.93
Type-E (Small-Medium)	686.42	0.25	689.26	0.63	1.37	686.76	0.28	689.18	0.58	1.60	686.61	0.30	690.07	0.70	1.91
Type-I (Small-Medium)	669.66	0.17	672.61	0.55	1.70	669.89	0.21	672.75	0.58	2.10	670.86	0.34	673.13	0.59	2.28
Average	***675.48***	***0.21***	***678.05***	***0.54***	***1.55***	***675.69***	***0.25***	***678.48***	***0.61***	***1.78***	***676.12***	***0.32***	***679.40***	***0.70***	***2.04***
Type-U (Large)	610.05	0.61	613.26	1.16	6.02	610.11	0.67	614.60	1.34	6.80	611.71	0.86	614.88	1.43	7.09
Type-E (Large)	620.84	0.55	624.27	1.06	5.38	621.10	0.61	625.25	1.22	6.03	623.20	0.90	626.27	1.42	6.29
Type-I (Large)	615.49	0.67	618.01	1.08	7.06	615.26	0.65	618.90	1.19	7.99	617.61	0.94	620.01	1.40	8.35
Average	***615.46***	***0.61***	***618.51***	***1.10***	***6.15***	***615.49***	***0.64***	***619.58***	***1.25***	***6.94***	***617.50***	***0.90***	***620.39***	***1.42***	***7.24***

Table 9 compares the average of five runs and the best results for all instances of each data set, using each algorithm. Columns "Best" ("Avg") report the best (average) solution values of our implemented standard algorithms (i.e., VNS$_1$, VNS$_2$ and VNS$_3$). Columns "%" presents the percentage of deviation from the best (Avg) solutions obtained by our VNS$_1$, VNS$_2$ and VNS$_3$, compared to the EVO-VNS$_1$, EVO-VNS$_2$ and EVO-VNS$_3$.

The results in Table 9 clearly indicate that our evolutionary algorithms outperform the standard algorithms in terms of best solution value and average solution quality. In terms of average value of five runs for the small-medium instances, VNS$_3$, for example, has obtained an average value of 679.40 compared to 674.76 obtained by EVO-VNS$_3$, with an average gap difference equal to 0.70%. For the larger instances, the use of our evolutionary strategies achieves remarkable improvement in comparison with the standard methods in terms of best solution and average solution value over five runs. This confirms that our evolutionary algorithms are more stable and more efficient than the standard algorithms. In addition, the detailed results of Table 9 show the ability of the evolutionary algorithms to obtain good solutions in most of the runs, in comparison with the standard algorithms. Moreover, in large-sized instances, the basic VNS$_1$, VNS$_2$ and VNS$_3$ can only find 29, 22 and 16 best solutions, compared to 49, 45 and 43 found by the EVO-VNS$_1$, EVO-VNS$_2$ and EVO-VNS$_3$ algorithms.

Finally, a comparative study is shown in Table 13 in Appendix B to assess the impact of using the population phase, with and without the crossover MX1 operator and other components in our EVO-VNS algorithms.

5.4. Energy consumption function versus constant energy consumption

In this section, we evaluate the impact of applying the realistic energy consumption function adopted in our algorithms. We have tested two strategies: Realistic Consumption (RC) function of Genikomsakis and Mitrentsis (2017) that is the case of our current study, and Constant Consumption (CC) which is equal to 380 W/mile (EPA) applied also using the same vehicle model described in Table 2. We chose to test our EVO-VNS$_1$ using both these two strategies. Table 9 displays the results of the comparison. In Table 10 (and its detailed results in the website), the Column "Nb-stat available" presents the number of BSSs available in each instance and the Column "Nb-Bat-swap" presents the number of battery-swaps performed by the vehicles in each instance. Each instance is solved five times by applying each algorithm.

Table 10
Importance of the realistic energy consumption function in our algorithms

Inst.	BS	EVO-VNS$_1$ with RC				EVO-VNS$_1$ with CC			
		Best	Best%	Avg	Avg%	Best	Best%	Avg	Avg%
U	668.51	668.74	0.02	668.89	0.03	669.98	0.24	670.95	0.39
E	684.36	684.65	0.02	684.77	0.03	685.51	0.15	686.35	0.26
I	668.15	668.38	0.02	668.69	0.04	669.13	0.15	670.36	0.32
UEI	***673.67***	***673.92***	***0.02***	***674.12***	***0.03***	***674.87***	***0.18***	***675.89***	***0.32***

From Table 10, we can see that using the RC strategy is better than applying the CC strategy with an average gap equal to 0.02%(0.03%) compared to 0.18%(0.32%), respectively. In addition, from the detailed results of this table in our website, we can observe that in some instances, the results obtained through the application of the RC strategy in terms of the number of battery swaps are better than the best results realized with the use of the CC strategy (as indicated in bold). For the rest of the instances, we observe that, in both RC and CC strategies, our algorithms were capable of obtaining the same results. As we anticipated before, our algorithms with the use of realistic consumption energy function have performed well. Moreover, using the RC strategy is more efficient than the CC strategy, as also confirmed by De Gennaro et al. (2015).

5.4. The effect on battery capacity

In this subsection, we analyze the effect of using different battery capacity H value in terms of the objective function as well as on visiting the BSSs. In this experiment, we use EVO-VNS$_1$ as an example. In Table 11, three different battery capacity values H=35, 30 and 25 are tested against the original H=40 value, using our EVO-VNS$_1$, with the limitation restraints of route duration and the number of available vehicles. The experiment's results are applied to all instance types for each data set. The column "Best%" ("Avg%") indicates the percentage of deviation from the best solution (column "BS") obtained by any algorithm for a given instance for H=40. The detailed results are given in our website. In addition, in the detailed results of this table, we show the number of available BSSs in each instance in the column denoted by "Nb-Ava-BSS", the columns “Nb-Veh” and “Nb-Cl” present the number of available vehicles and the number of users in each instance, respectively, while, the column “Nb-Bat-swap” presents the number of visited BSSs.

Table 11
Impact of battery capacity

Instance (Data)	BS	*H*=40			*H*=35			*H*=30			*H*=25		
		Best%	Avg%	CPU (min)	Best%	Avg%	CPU (min)	Best%	Avg%	CPU (min)	Best%	Avg%	CPU (min)
Type-U (Small-Medium)	668.51	0.02	0.03	2.52	0.65	0.67	2.56	1.29	1.33	2.40	2.81	2.91	2.56
Type-E (Small-Medium)	684.36	0.02	0.03	2.18	0.80	0.82	2.20	2.03	2.07	1.98	3.99	4.10	2.02
Type-I (Small-Medium)	668.15	0.02	0.05	2.68	1.04	1.07	2.72	1.93	1.99	2.55	3.51	3.61	2.79
Average	***673.67***	***0.02***	***0.04***	***2.46***	***0.83***	***0.86***	***2.49***	***1.75***	***1.79***	***2.31***	***3.44***	***3.54***	***2.46***
Type-U (Large)	605.76	0.04	0.40	9.00	0.21	0.70	9.11	0.28	0.77	9.21	1.07	1.56	9.91
Type-E (Large)	616.95	0.09	0.47	8.09	0.25	0.85	8.18	0.34	0.94	8.30	2.25	2.85	8.94
Type-I (Large)	610.85	0.08	0.50	10.58	0.46	1.07	10.73	0.87	1.49	10.86	1.95	2.56	11.91
Average	***611.18***	***0.07***	***0.45***	***9.22***	***0.31***	***0.87***	***9.34***	***0.50***	***1.06***	***9.46***	***1.76***	***2.32***	***10.25***

From Table 11 and its detailed results in the website, we can see that changing the battery capacity value influences the objective function. The average gap for the best run (five runs) is equal to 0.83%(0. 86%), when using H=35, 1.75%(1.79%), when using H=30, and 3.44%(3.54%) when using H=25 for the small-medium size instances. This increase in the objective function value is related to the additional distances and costs caused by visiting the BSSs. In fact, the number of visits to the BSSs increases with each reduction of H. Specifically, the number of BSS visits varies between 0 and 4 using H=40, which presents our case, between 0 and 5 using H=35, between 1 and 5 using H=30, and between 1 and 8 for H=25. Moreover, using H=30 leads to infeasible solutions for 5 out of 72 instances, while for H =25, 19 infeasible solutions are obtained. For the large instances, we can see also that the objective function value increased with each reduction of the battery capacity.

We also observed that the small-medium instances that are characterized with a short time window interval tend to include many visits to BSSs in many instances and for different battery capacities. This is expected, since the vehicles need to travel longer distances in order to satisfy the tight time windows constraint, i.e., they cover larger areas. On the other hand, instances that are characterized with a large time window usually have no visits for BSSs using H=40, while they have only one or two visits when using H=35, H=30 and H=25.

6. Conclusions

This paper presents a practical version of the dial-a-ride problem and investigates electric vehicles with battery swapping stations (DARP-EV). The use of electric vehicles has been already observed in practice, especially in the domain of healthcare services. Hence, there is a need for developing new models and solution techniques considering electric vehicle technologies. This paper provides a new formulation that allows the recharging of an electric vehicle by swapping its depleted battery with a full one in each visit to a battery swapping station, in order to continue its working day and fulfill all users' requirements. We proposed three Evolutionary Variable Neighborhood Search (EVO-VNS) algorithms for solving the DARP-EV by introducing effective (population-based) construction and diversification mechanisms and advanced local search operators. After conducting thorough sensitivity analysis and parameter tuning, our algorithms were intensively put into experimentation on generated data sets with different sizes (small, medium and large). As demonstrated by the experimental results, our algorithms obtain high quality solutions during moderate processing times. In addition, the results demonstrate that our proposed algorithms, which combine features of evolutionary metaheuristics with VNS, have more advantages in their performance than our standalone VNS method. Furthermore, high quality solutions were obtained by our proposed algorithms in comparison with a recent hybrid GA algorithm for the DARP.

Finally, we highlight some limitations that are considered in our work, which constitute various attractive avenues for future investigation. First, the accuracy of the energy consumption estimation presented in our work depends on the resolution of the road network representation. Applying the energy consumption function for all nodes has implications on the accuracy of the estimated energy consumption. The current modelling approach is intended to exemplify the application of an EV energy consumption function in the DARP-EV formulation. However, the accuracy of the energy consumption model can be improved by considering that the trip between two nodes consists of smaller segments, each one with its own road angle and possible speed profile. In addition, to make the EVs model more realistic, the EVs can consider the combination of the brake power, battery loss, temperature, drive loss, acceleration, deceleration, as well as the consideration of the effect of the payload. For more details, interested readers are referred to Simpson (2005). The implementation of these approaches, though, is out of the scope of the current paper and it is suggested in the conclusions as possible directions for future work. In addition, from a methodological perspective, developing exact solution methods, such as Branch-and-Cut for solving moderate size instances of the DARP-EV, and designing efficient metaheuristic techniques for solving rich variants of the problem are interesting research directions. For example, considering realistic energy consumption as well as other relevant constraints to solve practical applications, such as the Shared-Taxi VRP and the Share-a-Ride Problem (SARP). Finally, considering a real data set in the field of DARP-EV is also one of the promising perspectives of this work.

Acknowledgements

Thanks are due to Professor Jiuh-Biing Sheu and to the anonymous referees for their valuable comments and suggestions which helped to improve the quality of this paper.

References

Adler, J. D., Mirchandani, P. B., 2014. Online routing and battery reservations for electric vehicles with swappable batteries. *Transportation Research Part B: Methodological*, *70*, 285-302.

Adler, J. D., Mirchandani, P. B., 2016. The vehicle scheduling problem for fleets with alternative-fuel vehicles. *Transportation Science*, *51*(2), 441-456.

Agrawal, S., Zheng, H., Peeta, S., Kumar, A., 2016. Routing aspects of electric vehicle drivers and their effects on network performance. *Transportation Research Part D: Transport and Environment*, *46*, 246-266.

Americans with disabilities act., 2009.URL. https://www.ada.gov/pubs/adastatute08.htm. Last accessed on 01/06/2017.

Amirgholy, M., Gonzales, E. J., 2016. Demand responsive transit systems with time-dependent demand: user equilibrium, system optimum, and management strategy. *Transportation Research Part B: Methodological*, *92*, 234-252.

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

Bard, J. F., Huang, L., Dror, M., Jaillet, P. 1998. A branch and cut algorithm for the VRP with satellite facilities. *IIE transactions*, *30*(9), 821-834.

Beaudry, A., Laporte, G., Melo, T., Nickel, S., 2010. Dynamic transportation of patients in hospitals. *OR spectrum*, *32*(1), 77-107.

Bektaş, T., Laporte, G., 2011. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8), 1232-1250.

Bektaş, T., Demir, E., Laporte, G., 2016. Green vehicle routing. In *Green Transportation Logistics* (pp. 243-265). Springer International Publishing.

Blum, C., Puchinger, J., Raidl, G. R., Roli, A., 2011. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, *11*(6), 4135-4151.

Boyacı, B., Zografos, K. G., Geroliminis, N., 2017. An integrated optimization-simulation framework for vehicle and personnel relocations of electric carsharing systems with reservations. *Transportation Research Part B: Methodological*, *95*, 214-237.

Braekers, K., Caris, A., Janssens, G. K., 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, *67*, 166-186.

Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I. 2016. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, *99*, 300-313.

Braekers, K., Kovacs, A. A., 2016. A multi-period dial-a-ride problem with driver consistency. *Transportation Research Part B: Methodological*, *94*, 355-377.

Carrizosa, E., Mladenović, N., Todosijević, R., 2013. Variable neighborhood search for minimum sum-of-squares clustering on networks. *European Journal of Operational Research*, *230*(2), 356-363.

Carsales.com.au, 2015. New 2014 Nissan LEAF ZE0 Auto Pricing and Specifications. Available in http://www.carsales.com.au/new-cars/details/Nissan-LEAF-2014/SHRM-AD-393592/?sdmvc=1?intref=sr-bncis-model-stock. Last accessed 16.12.15.

Caporossi, G., Hansen, P., Mladenović, N. 2016. Variable Neighborhood Search. In *Metaheuristics* (pp. 77-98). Springer International Publishing.

Centers for Disease Control and Prevention, 2012. Available in https://www.cdc.gov/nchs/data/series/sr_11/sr11_252.pdf.

Chassaing, M., Duhamel, C., Lacomme, P., 2016. An ELS-based approach with dynamic probabilities management in local search for the dial-a-ride problem. Engineering Applications of Artificial Intelligence 48, 119–133.

Conrad, R. G., Figliozzi, M. A., 2011. The recharging vehicle routing problem. In *IIE Annual Conference. Proceedings*(p. 1). Institute of Industrial and Systems Engineers (IISE).

Cordeau, J. F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, *54*(3), 573-586.

Cordeau, J. F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, *37*(6), 579-594.

Cordeau, J. F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. *Annals of operations research*, *153*(1), 29-46.

Cordeau, J. F., Laporte, G., Mercier, A., 2001. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, *52*(8), 928-936.

De Gennaro, M., Paffumi, E., Martini, G., Manfredi, U., Vianelli, S., Ortenzi, F., Genovese, A., 2015. Experimental test campaign on a battery electric vehicle: laboratory test results (Part 1). *SAE International Journal of Alternative Powertrains*, *4*(2015-01-1167), 100-114.

Demir, E., Bektaş, T., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, *223*(2), 346-359.

Demir, E., Huang, Y., Scholts, S., Van Woensel, T., 2015. A selected review on the negative externalities of the freight transportation: Modeling and pricing. *Transportation Research Part E: Logistics and Transportation Review*, *77*, 95-114.

Desaulniers, G., Errico, F., Irnich, S., Schneider, M., 2016. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, *64*(6), 1388-1405.

Detti, P., P., Papalini, F., de Lara, G. Z. M., 2017. A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega*, *70*, 1-14

Doerner, K., Salazar-Gonzalez, J., 2014. In: Toth, P. and Vigo, D. (Eds). Vehicle Routing: Problems, Methods, and Applications, Second Edition. MOSSIAM Series on Optimation, *Society for Industrial and Applied Mathematics*.

Dongarra, J., 2014. Performance of Various Computers Using Standard Linear Equations Software, (Linpack Benchmark Technical Report, CS-89-85). University of Tennessee, Computer Science Department.

Dueck, G., 1993. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, *104*(1), 86-92.

Eisner, J., Funke, S., Storandt, S., 2011,. Optimal Route Planning for Electric Vehicles in Large Networks. In *AAAI*(pp. 1108-1113).

EPA, 2016. The EPA 10 gallon per minute fuel dispensing limit U. E. P. agency. 40 CFR 80.22.

Erdoğan, S., Miller-Hooks, E., 2012. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review 48*, 1, 100–114.

Eusuff, M. M., Lansey, K. E., 2003. Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources planning and management*, *129*(3), 210-225.

Eusuff, M., Lansey, K., Pasha, F., 2006. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering optimization*, *38*(2), 129-154.

Fang, C., Wang, L., 2012. An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. *Computers & Operations Research*, *39*(5), 890-901.

Felipe, Á., Ortuño, M. T., Righini, G., Tirado, G., 2014. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, *71*, 111-128.

Froger, A., Mendoza, J.E., Jabali, O., Laporte, G., 2017a. A Matheuristic for the Electric Vehicle Routing Problem with Capacitated Charging Stations. Technical Report CIRRELT -2017-31, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport.

Froger, A., Mendoza, J.E., Jabali, O., Laporte, G., 2017b. New formulations for the electric vehicle routing problem with nonlinear charging functions. Technical Report CIRRELT -2017-30, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport.

Fuller, M., 2016. Wireless charging in California: Range, recharge, and vehicle electrification. *Transportation Research Part C: Emerging Technologies*, *67*, 343-356.

Genikomsakis, K. N., Mitrentsis, G., 2017. A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications. *Transportation Research Part D: Transport and Environment*, *50*, 98-118.

Guan, J., Lin, G. 2016. Hybridizing variable neighborhood search with ant colony optimization for solving the single row facility layout problem. *European Journal of Operational Research*, *248*(3), 899-909.

Guo, P., Cheng, W., Wang, Y., 2017. Hybrid evolutionary algorithm with extreme machine learning fitness function evaluation for two-stage capacitated facility location problems. *Expert Systems with Applications*, *71*, 57-68.

Goeke, D., Schneider, M., 2015. Routing a mixed fleet of electric and conventional vehicles. European Journal of Operational Research 245, 1, 81 – 99.

He, F., Wu, D., Yin, Y., Guan, Y., 2013. Optimal deployment of public charging stations for plug-in hybrid electric vehicles. *Transportation Research Part B: Methodological*, *47*, 87-101.

Hemmelmayr, V. C., Doerner, K. F., Hartl, R. F., 2009. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, *195*(3), 791-802.

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*(3), 995-1018.

Hof, J., Schneider, M., Goeke, D., 2017. Solving the battery swap station location-routing problem with capacitated electric vehicles using an AVNS algorithm for vehicle-routing problems with intermediate stops. *Transportation Research Part B: Methodological*, *97*, 102-112.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Jabir, E., Panicker, V. V., Sridharan, R., 2017. Design and development of a hybrid ant colony-variable neighbourhood search algorithm for a multi-depot green vehicle routing problem. *Transportation Research Part D: Transport and Environment*, *57*, 422-457.

Jorgensen, R. M., Larsen, J., Bergvinsdottir, K. B., 2007. Solving the dial-a-ride problem using genetic algorithms. *Journal of the operational research society*, *58*(10), 1321-1331.

Juan, A. A., Mendez, C. A., Faulin, J., de Armas, J., Grasman, S. E., 2016. Electric vehicles in logistics and transportation: a survey on emerging environmental, strategic, and operational challenges. *Energies*, *9*(2), 86.

Keskin,M., Catay, B., 2016. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies 65*, 111 – 127.

Kundakcı, N., Kulak, O., 2016. Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. *Computers & Industrial Engineering*, *96*, 31-51.

Kobayashi, Y., Kiyama, N., Aoshima, H., Kashiyama, M., 2011. A route search method for electric vehicles in consideration of range and locations of charging stations. In *Intelligent Vehicles Symposium (IV), 2011 IEEE* (pp. 920-925). IEEE.

Koç, Ç., Bektaş, T., Jabali, O., Laporte, G., 2015. A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows.*Computers & Operations Research*, 64, 11-27.

Koç, Ç., Karaoglan, I., 2016. The green vehicle routing problem: A heuristic based exact solution approach. *Applied Soft Computing*, 39, 154-164.

Kok, C., Mendoza, J.E., Jabali, O., Laporte, G., 2017. The electric vehicle routing problem with shared charging stations. Technical report, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport.

Li, J. Q. 2013. Transit bus scheduling with limited energy. *Transportation Science*, *48*(4), 521-539.

Liao, C. S., Lu, S. H., Shen, Z. J. M., 2016. The electric vehicle touring problem. *Transportation Research Part B: Methodological*, *86*, 163-180.

Lim, A., Zhang, Z., Qin, H., 2016. Pickup and delivery service with manpower planning in hong kong public hospitals. *Transportation Science*, *51*(2), 688-705.

Lin, S. W., Vincent, F. Y., 2015. A simulated annealing heuristic for the multiconstraint team orienteering problem with multiple time windows. *Applied Soft Computing*, *37*, 632-642.

Linpack, 2016. Available on http://www.roylongbottom.org.uk/

Liu, H., Wang, D. Z., 2017. Locating multiple types of charging facilities for battery electric vehicles. Forthcoming in *Transportation Research Part B: Methodological*.

Liu, M., Luo, Z., Lim, A., 2015. A branch-and-cut algorithm for a realistic dial-a-ride problem. *Transportation Research Part B: Methodological*, *81*, 267-288.

Mak, H. Y., Rong, Y., Shen, Z. J. M., 2013. Infrastructure planning for electric vehicles with battery swapping. *Management Science*, *59*(7), 1557-1575.

Markov, I., Varone, S., Bierlaire, M., 2016. Integrating a heterogeneous fixed fleet and a flexible assignment of destination depots in the waste collection VRP with intermediate facilities. *Transportation Research Part B: Methodological, 84*, 256-273.

Marković, N., Nair, R., Schonfeld, P., Miller-Hooks, E., Mohebbi, M., 2015. Optimizing dial-a-ride services in Maryland: Benefits of computerized routing and scheduling. *Transportation Research Part C: Emerging Technologies, 55*, 156-165.

Martínez-Lao, J., Montoya, F. G., Montoya, M. G., Manzano-Agugliaro, F., 2017. Electric vehicles in Spain: An overview of charging systems. Forthcoming in *Renewable and Sustainable Energy Reviews*.

Masmoudi, M. A., Hosny, M., Braekers, K., Dammak, A., 2016. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, *96*, 60-80.

Masmoudi, M. A., Braekers, K., Masmoudi, M., Dammak, A., 2017. A hybrid Genetic Algorithm for the Heterogeneous Dial-A-Ride Problem. *Computers & Operations Research*, *81*, 1-13.

Masson, R., Lehuédé, F., & Péton, O., 2014. The dial-a-ride problem with transfers. *Computers & Operations Research*, *41*, 12-23.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E., 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*, *21*(6), 1087-1092.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097-1100.

Mladenović, N., Urošević, D., Ilić, A., 2012. A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. *European Journal of Operational Research*, *220*(1), 270-285.

Mladenović, N., Sifaleras, A., Sörensen, K., 2017. Editorial to the Special Cluster on Variable Neighborhood Search, Variants and Recent Applications. *International Transactions in Operational Research*, *24*(3), 507-508.

Molenbruch, Y., Braekers, K., Caris, A., 2017. Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, 1-31.

Montoya, A., Guéret, C., Mendoza, J. E. Villegas, J. G., 2015. A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies 70*, 113 – 128.

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

Muelas, S., LaTorre, A., Peña, J. M., 2013. A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city. *Expert Systems with Applications*, *40*(14), 5516-5531.

Muelas, S., LaTorre, A., Peña, J. M., 2015. A distributed VNS algorithm for optimizing dial-a-ride problems in large-scale scenarios. *Transportation Research Part C: Emerging Technologies*, *54*, 110-130.

Muter, I., Cordeau, J. F., Laporte, G., 2014. A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transportation Science*, *48*(3), 425-441.

Nie, Y. M., Ghamami, M., 2013. A corridor-centric approach to planning electric vehicle charging infrastructure. *Transportation Research Part B: Methodological, 57,* 172-190.

Osman, I. H., 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, *41*(4), 421-451.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, *58*(1), 21-51.

Parragh, S. N., Doerner, K. F., Hartl, R. F., Gandibleux, X., 2009. A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks*, *54*(4), 227-242.

Parragh, S. N., Doerner, K. F., Hartl, R. F., 2010. Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, *37*(6), 1129-1138.

Parragh, S. N., 2011. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, *19*(5), 912-930.

Parragh, S. N., Schmid, V., 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, *40*(1), 490-497.

Parragh, S. N., Pinho de Sousa, J., Almada-Lobo, B., 2014. The dial-a-ride problem with split requests and profits. *Transportation Science*, *49*(2), 311-334.

Pelletier, S., Jabali, O., Laporte, G., Veneroni, M., 2017. Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models. Forthcoming in *Transportation Research Part B: Methodological*.

Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M., 2005. The bees algorithm. Technical note. *Manufacturing Engineering Centre, Cardiff University, UK*, 1-57.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. *Computers & Operations Research*, *34*(8), 2403-2435.

Polacek, M., Hartl, R. F., Doerner, K., Reimann, M., 2004. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics*, *10*(6), 613-627.

Polat, O., Kalayci, C. B., Kulak, O., Günther, H. O., 2015. A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit. *European Journal of Operational Research*, *242*(2), 369-382.

Potvin, J. Y., Rousseau, J. M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, *66*(3), 331-340.

Roberti, R., Wen, M., 2016. The electric traveling salesman problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, *89*, 32-52.
Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455-472.
Sachenbacher, M., Leucker, M., Artmeier, A., & Haselmayr, J. (2011, August). Efficient Energy-Optimal Routing for Electric Vehicles. In *AAAI* (pp. 1402-1407).
Sarasola, B., Doerner, K. F., Schmid, V., Alba, E., 2016. Variable neighborhood search for the stochastic and dynamic vehicle routing problem. *Annals of Operations Research*, *236*(2), 425-461.
Savelsbergh, M. W., 1992. The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, *4*(2), 146-154.
Sayarshad, H. R., Chow, J. Y., 2015. A scalable non-myopic dynamic dial-a-ride and pricing problem. *Transportation Research Part B: Methodological*, *81*, 539-554.
Schiffer, M., Walther, G., 2017. The electric location routing problem with time windows and partial recharging. *European Journal of Operational Research 260*, 3, 995–1013.
Schilde, M., Doerner, K.F., Hartl, R.F., 2011. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & Operations Research*, 38(12), pp 1719–1730.
Schilde, M., Doerner, K. F., Hartl, R. F., 2014. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European journal of operational research*, *238*(1), 18-30.
Schneider, M., Stenger, A., Goeke, D., 2014. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*,*48*(4), 500-520.
Siddiqi, U. F., Shiraishi, Y., Sait, S. M., 2011, November. Multi-constrained route optimization for electric vehicles (EVs) using particle swarm optimization (PSO). In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on* (pp. 391-396). IEEE.
Simpson, A. G., 2005. Parametric modelling of energy consumption in road vehicles.
Tesla Motors Inc., 2013. Available from URL: https://www.tesla.com/videos/battery-swap-event. Last accessed on 17/01/2016.
The Engineering Toolbox. URL. http://www.engineeringtoolbox.com/rolling-friction-resistance-d_1303.html.
Tiwari, M. K., Kumar, S., Shankar, R., 2006. Solving part-type selection and operation allocation problems in an FMS: An approach using constraints-based fast simulated annealing algorithm. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *36*(6), 1170-1184.
Todosijević, R., Benmansour, R., Hanafi, S., Mladenović, N., Artiba, A., 2016. Nested general variable neighborhood search for the periodic maintenance problem. *European Journal of Operational Research*, *252*(2), 385-396.
Travesset-Baro, O., Rosas-Casals, M., Jover, E. 2015. Transport energy consumption in mountainous roads. A comparative case study for internal combustion engines and electric vehicles in Andorra. *Transportation Research Part D: Transport and Environment*, *34*, 16-26.
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(1), 475-489.
Wang, I. L., Wang, Y., Lin, P. C., 2016. Optimal recharging strategies for electric vehicle fleets with duration constraints. *Transportation Research Part C: Emerging Technologies*, *69*, 242-254.
Wang, Y. W., Lin, C. C., 2013. Locating multiple types of recharging stations for battery-powered electric vehicle transport. *Transportation Research Part E: Logistics and Transportation Review*, *58*, 76-87.
Wei, L., Zhang, Z., Zhang, D., Lim, A., 2015. A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, *243*(3), 798-814.
Wen, M., Linde, E., Ropke, S., Mirchandani, P., Larsen, A., 2016. An adaptive large neighborhood search heuristic for the Electric Vehicle Scheduling Problem. *Computers & Operations Research*, *76*, 73-83.
Wong, K. I., Bell, M. G., 2006. Solution of the Dial-a-Ride Problem with multi-dimensional capacity constraints. *International Transactions in Operational Research*, *13*(3), 195-208.
Wu, X., Freese, D., Cabrera, A., Kitch, W. A., 2015. Electric vehicles' energy consumption measurement and estimation. *Transportation Research Part D: Transport and Environment*, *34*, 52-67.
Xia, H., Li, X., Gao, L. 2016. A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling. *Computers & Industrial Engineering*, *102*, 99-112.
Xiang, Z., Chu, C., Chen, H., 2006. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European journal of operational research*, *174*(2), 1117-1139.
Xu, M., Meng, Q., Liu, K., Yamamoto, T., 2017. Joint charging mode and location choice model for battery electric vehicle users. Forthcoming in *Transportation Research Part B: Methodological*.
Yang, J., Sun, H., 2015. Battery swap station location-routing problem with capacitated electric vehicles. *Computers & Operations Research*, *55*, 217-232.
Yavuz, M., 2017. An iterated beam search algorithm for the green vehicle routing problem. *Networks*, 69(3), 317-328.
Yu, M., Zhang, Y., Chen, K., Zhang, D., 2015. Integration of process planning and scheduling using a hybrid GA/PSO algorithm. *The International Journal of Advanced Manufacturing Technology*, *78*(1-4), 583-592.
Zhang, Z., Liu, M., Lim, A., 2015. A memetic algorithm for the patient transportation problem. *Omega*, *54*, 60-71.

Appendix A: Parameters of our standard algorithms

The parameters used in our standard algorithms are shown in Table 12.

Table 12
Parameters used in the standard algorithms

Algorithm	Description of parameters	Best value
VNS$_1$	Number of iterations (n_{vns1})	No improvement of the best solution after five consecutive iterations
	Roulette wheel parameter (r_p)	0.70
	Score of a global better solution (π_1)	15
	Score of a better solution (π_2)	10
	Score of a worse solution (π_3)	5
	Initial temperature (T_{max})	100
	Cooling rate (δ)	0.99975
VNS$_2$	Number of iterations (n_{vns2})	No improvement of the best solution after five consecutive iterations
	Roulette wheel parameter (r_p)	0.70
	Score of a global better solution (π_1)	15
	Score of a better solution (π_2)	10
	Score of a worse solution (π_3)	5
	Deviation value (Dev)	0.01* current global Record
	Number of iterations to apply local search (n_{loc})	Number of vehicles in the instance
VNS$_3$	Number of iterations (n_{vns3})	No improvement of the best solution after five consecutive iterations
	Number of iterations to apply neighborhood (n_{neig})	Number of vehicles in the instance
EVO-VNS$_1$	Number of groups (m)	3
	Number of solutions in each group (c)	6
	Number of solutions to be improved by VNS variant (s)	2
	Stopping criterion of the EVO-VNS$_1$ (n_{Hvns1})	No improvement of the best solution after five consecutive iterations
	Number of iterations of VNS$_1$ (n_{vns1})	1,000
	Roulette wheel parameter (r_p)	0.70
	Score of a global better solution (π_1)	15
	Score of a better solution (π_2)	10
	Score of a worse solution (π_3)	5
	Initial temperature (T_{max})	100
	Cooling rate (δ)	0.99975
EVO-VNS$_2$	Number of groups (m)	3
	Number of solutions in each group (c)	6
	Number of solutions to be improved by VNS variant (s)	2
	Stopping criterion of the EVO-VNS$_2$ (n_{Hvns2})	No improvement of the best solution after five consecutive iterations
	Number of iterations of VNS$_2$ (n_{vns2})	1,000
	Roulette wheel parameter (r_p)	0.70
	Score of a global better solution (π_1)	15
	Score of a better solution (π_2)	10
	Score of a worse solution (π_3)	5
	Deviation value (Dev)	0.01* current global Record
	Number of iterations to apply local search (n_{loc})	Number of vehicles in the instance
EVO-VNS$_3$	Number of groups (m)	3
	Number of solutions in each group (c)	6
	Number of solutions to be improved by VNS variant (s)	2
	Stopping criterion of the EVO-VNS$_3$ (n_{Hvns3})	No improvement of the best solution after five consecutive iterations
	Number of iterations of VNS$_3$ (n_{vns3})	1,000
	Number of iterations to apply neighborhood (n_{neig})	Number of vehicles in the instance

Appendix B: Effect of the different algorithmic components on the EVO-VNS

We investigate here the effect of integrating our different components (i.e., our population phase based on the SFLA and the BA as well as using the MX1 operator) on our standalone (enhanced)VNS to diversify the search and improve the quality of solutions. For this purpose, we use VNS$_1$, as an example, where some combinations are compared based on our standalone(enhanced) VNS$_1$ by adding in each time different component(s). The results of this comparison are shown in Table 13, where we use the benchmark instances of Masmoudi et al. (2017). Combination “1” represents our standalone (enhanced) VNS$_1$. Combination “2” represents the combination of adding the population phase based on the SFLA and the component adopted from the BA (i.e., step 2 and steps 5 and 6, respectively, of Algorithm 1) to the standalone VNS$_1$. In other words, this combination represents our EVO-VNS1 without the MX1 operator. The last combination “3” is the combination of adding to the standalone(enhanced) VNS$_1$ both the population phase and the MX1 operator, which reflects our complete EVO-VNS$_1$ with MX1 operator shown in Algorithm 1.

In Table 13, the column “Best”(“Avg”), presents the best (Avg) solutions provided by the hybrid GA of Masmoudi et al. (2017). The columns “Best%” (“Avg%”) indicate the percentage of deviation from the best(Avg) results obtained by the hybrid GA of Masmoudi et al. (2017), respectively. The obtained results (best and average) values of this table are shown in our website.

Table 13
Effect of different components

Inst.	Hybrid GA			Combination 1			Combination 2			Combination 3		
	Best	Avg	CPU (min)	Best%	Avg%	CPU (min)	Best%	Avg%	CPU (min)	Best%	Avg%	CPU (min)
R1a	195.97	195.97	0.44	0.00	0.00	1.13	0.00	0.00	0.95	0.00	0.00	0.71
R2a	336.34	336.34	0.85	0.00	0.00	1.87	0.00	0.00	1.56	0.00	0.00	1.18
R3a	586.18	589.86	0.94	0.00	0.14	2.22	0.00	**-0.39**	1.84	0.00	**-0.46**	1.40
R4a	640.03	642.56	1.48	0.40	0.27	3.48	0.00	0.00	2.86	**-0.16**	**-0.04**	2.21
R5a	714.83	718.51	1.82	0.00	0.23	4.39	**-0.02**	0.10	3.64	**-0.24**	**-0.26**	2.77
R6a	883.02	887.65	2.40	0.82	0.75	6.55	0.13	**-0.33**	5.40	**-0.10**	**-0.50**	4.16
R7a	312.05	312.96	0.47	0.69	1.00	7.58	0.15	0.43	6.29	**-0.35**	0.05	4.79
R8a	553.82	556.23	0.81	0.90	0.94	8.30	0.31	0.20	6.89	0.04	0.03	5.24
R9a	746.23	748.87	1.62	1.12	1.12	11.63	0.18	0.07	9.68	**-0.21**	**-0.05**	7.33
R10a	963.08	969.22	2.29	1.13	1.17	16.99	0.39	0.61	14.12	0.09	**-0.48**	10.74
R1b	190.39	190.39	0.57	0.00	0.00	1.26	0.00	0.00	1.05	0.00	0.00	0.80
R2b	312.92	314.12	1.03	0.00	0.36	2.58	0.00	**-0.38**	2.14	0.00	**-0.38**	1.63
R3b	551.95	553.15	1.36	0.00	0.69	3.26	0.00	0.19	2.70	0.00	0.01	2.06
R4b	606.08	610.48	2.00	0.58	0.30	4.66	0.16	**-0.14**	3.86	**-0.13**	**-0.32**	2.95
R5b	641.84	642.15	2.93	0.69	0.77	6.99	-0.13	0.13	5.80	**-0.21**	**-0.01**	4.42
R6b	832.53	836.32	3.66	0.86	0.68	8.70	0.23	0.27	7.18	**0.03**	**-0.06**	5.51
R7b	276.52	276.52	0.79	0.90	1.27	1.92	0.00	0.34	1.60	**-0.13**	0.03	1.21
R8b	530.56	532.28	1.53	0.73	0.91	6.74	0.23	0.43	5.64	**-0.11**	**-0.09**	4.24
R9b	699.06	703.15	2.84	1.22	0.99	10.31	0.42	0.06	8.52	**-0.13**	**-0.40**	6.54
R10b	902.17	906.91	3.11	1.28	1.55	18.47	0.51	0.83	15.27	0.12	**-0.30**	11.71
Avg	***573.78***	***576.18***	***1.65***	***0.57***	***0.66***	***6.45***	***0.13***	***0.12***	***5.35***	***-0.07***	***-0.16***	***4.08***

As seen in Table 13, we observe that using only our standalone(enhanced) VNS$_1$ (combination “1”) cannot obtain good results compared the best (average) solutions of the hybrid GA of Masmoudi et al. (2017), with a positive deviation gap equal to 0.57%(0.66%). A considerable improvement is obtained by adding our evolutionary phase to the VNS$_1$(combination “2”), where in some instances a negative deviation gap is obtained compared to the best and average results of the hybrid GA. Thus, combination “2” shows clearly that using a population phase based on SFLA as well as using the BA technique of diversification contribute positively to the quality of solutions and clearly outperform the standalone(enhanced)VNS$_1$. This can be attributed to their added value in terms of balancing between exploration and exploitation. Comparing combination “2” and combination “3”, we observe positive percent deviation values of the results obtained by combination “2” compared to combination “3”, relative to the hybrid GA. Therefore, it is evident that applying this strategy alone is still unable to keep away from convergence to local optima during the evolutionary process. In fact, the elimination of our MX1 from the evolutionary process made the performance of combination “2” unsuccessful, compared the results obtained by the hybrid GA, as well as to our proposed EVO-VNS$_1$ with MX1 (combination “3”). In conclusion, applying all SFLA, BA and MX1 components to our standalone VNS$_1$ is indeed the most effective combination, compared to the other combinations. Our EVO-VNS$_1$ can obtain good results compared to the hybrid GA, although with a slight improvement of 0.07%(0.16%) in terms of best(average) results of the hybrid GA.