energies

MDPI

Article

Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery: Mathematical Modeling and Adaptive Large Neighborhood Search Heuristic Method

Wei Xu [1,†], Chenghao Zhang [2,†], Ming Cheng [2,*] and Yucheng Huang [2,*]

[1] Department of Engineering, Applied Technology College of Soochow University, Kunshan 215325, China
[2] School of Rail Transportation, Soochow University, Suzhou 215137, China
* Correspondence: mcheng1@suda.edu.cn (M.C.); ychuang@suda.edu.cn (Y.H.)
† These authors contributed equally to this work.

Abstract: Electric vehicles (EVs) are a promising option to reduce air pollution and shipping costs, especially in urban areas. To provide scientific guidance for the growing number of logistics companies using EVs, we investigated an electric-vehicle-routing problem with simultaneous pickup and delivery that also considers non-linear charging and load-dependent discharging (EVRPSPD-NL-LD). The objective was to minimize the total number of EVs and the total working time, including travel time, charging time, waiting time, and service time. We formulated the problem as a mixed integer linear program (MILP), and small-size problems could be solved to optimality in an acceptable amount of time using the commercial solver IBM ILOG CPLEX Optimization Studio (CPLEX). In view of the fact that the problem is NP-hard, an adaptive large neighborhood search (ALNS) metaheuristic method was proposed to solve large-size problems. Meanwhile, new operators and a time priority approach were developed to provide options for different scenarios. The results of our computational investigation and sensitivity analysis showed that the proposed methods are effective and efficient for modified benchmark instances.

Keywords: electric-vehicle-routing problem; mixed integer linear problem; non-linear charging; load-dependent discharging; adaptive large neighborhood search; energy saving

check for updates

Citation: Xu, W.; Zhang, C.; Cheng, M.; Huang, Y. Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery: Mathematical Modeling and Adaptive Large Neighborhood Search Heuristic Method. *Energies* **2022**, *15*, 9222. https://doi.org/10.3390/en15239222

Academic Editor: Charisios Achillas

Received: 18 October 2022
Accepted: 2 December 2022
Published: 5 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the past decade, due to the increasing energy crisis and the pressure of environmental protection, electric vehicles (EVs) have become one of the main directions of future automotive development for zero carbon emissions, higher driving experience, and lower cost [1]. The *Emissions Gap Report* [2] also notes that economic recovery after the COVID-19 crisis should be used as an opportunity to promote a low-carbon transition in the transportation sector. Currently, increasingly more logistics and distribution companies are adopting EVs for cargo distribution. With the tightening of environmental protection policies, regulations have been introduced to avoid traditional fossil fuel vehicles for distribution in the urban areas of some cities. However, the large-scale application of EVs is still hampered by technical and cost limitations, such as a long battery-charging time, limited driving range, lack of charging infrastructure, and high charging cost [3]. These problems also bring challenges to vehicle scheduling and route planning. Therefore, it is critical to establish an excellent scheduling policy, including charging plans, which can not only reduce the total travel time but also lower the distribution cost.

Reverse logistics is receiving increasingly more attention from manufacturing and distribution companies, which are increasingly faced with the reverse flow of finished goods or raw materials while distributing goods to customers. Thus, a practical variant of the electric-vehicle-routing problem (EVRP) is that customers may require both pickup and delivery services simultaneously. The problem derives from the traditional vehicle-routing

problem with simultaneous pickup and delivery (VRPSPD), which was first introduced by Min [4]. Compared to only pickup or delivery, the business model of simultaneous pickup and delivery can find numerous applications in the express service industry, manufacturing industry, and retail industry.

EVs are troubled not only by the limitations of the charging infrastructure and battery capacity but also by the accurate prediction of power consumption while driving. The vast majority of the literature assumes that the power consumption of EVs depends only on the distance traveled [5,6]. However, the energy consumption is affected not only by the driving distance but also by several other parameters, such as the gross vehicle weight, acceleration, and braking and road attributes. Therefore, to ensure that EVs are able to pick up and deliver goods from/to customers, it is necessary to take these factors into account to estimate energy consumption and to make reasonable visits to CSs and specify charging schedules.

In this paper, we address a variant of a practical electric-vehicle-routing problem that considers customers requiring both delivery and pickup services simultaneously, non-linear charging (NL), and load-dependent discharging (LD). The objective is to minimize (i) the number of vehicles and (ii) the total working time.

The novel contributions of this paper can be summarized as follows:

(1) A new mixed integer linear programming model for EVRPSPD-NL-LD is presented, in which (i) each customer has both pickup and delivery requests with time windows, (ii) the non-linear charging function is considered instead of a constant charging rate, and (iii) the power of EVs is discharged depending on not only the travel distance but also the weight of the current load.

(2) We developed efficient metaheuristics, i.e., an ALNS with and without new operators, and a time priority method to solve EVRPSPD-NL-LD.

The remainder of the paper is organized as follows: a literature review is presented in Section 2. Section 3 describes the problem and formulates a mathematical programming model. The proposed ALNS heuristic methods are designed in Section 4. Section 5 provides the computational investigation and discusses the results, including sensitivity analysis. Finally, in Section 6, conclusions and future research directions are presented.

2. Literature Review

2.1. Vehicle-Routing Problem with Simultaneous Pickup and Delivery

Parragh et al. [7] categorized general pickup and delivery problems into two problem classes, of which the simultaneous pickup and delivery problem we discuss in this paper is highlighted in Figure 1.

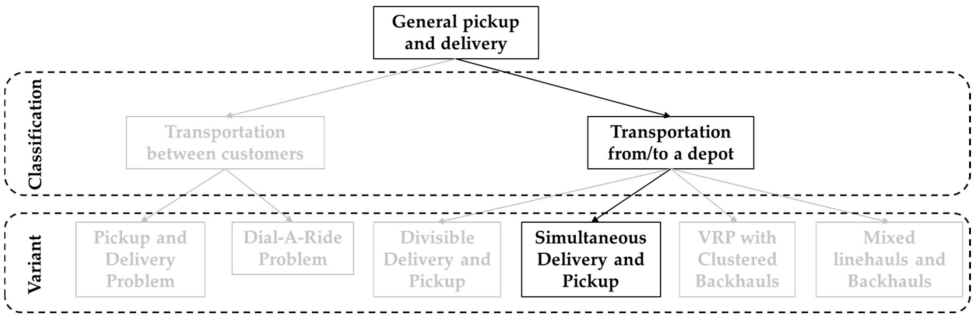


Figure 1. Pickup and delivery problems distinguished by Parragh et al. [7].

The earliest vehicle-routing problem with simultaneous pickup and delivery model was proposed by Min (1989) [4]. The prototype of his research came from the library book distribution problem in Franklin County, Ohio. The author showed that a large amount of travel time can be saved through his simultaneous delivery model and algorithm from experimental results.

Zachariadis and Kiranoudis [8] designed a local search meta-heuristic solution method and tested it on 18 existing large-scale benchmark instances. They concluded that the algorithm is robust and effective, improving previous solutions to most of the best-known test problems. Olgun et al. [9] established a green-vehicle-routing problem model for simultaneous pickup and delivery (G-VRPSPD) and studied the impact of a green objective function on the total fuel consumption cost. They proposed an HH-ILS heuristic algorithm and conducted sensitivity analysis. The results revealed that the green objective function has a significant impact on the total fuel consumption cost. Yu et al. [10] incorporated parcel lockers into the vehicle-routing problem with simultaneous pickup and delivery. They developed two simulated annealing (SA) algorithms to solve this problem, and the experimental results on a modified example showed that the algorithm is effective in solving VRPSPD-PL. The application of VRPSPD and its variants in industries is also presented in [11–13].

2.2. Electric-Vehicle-Routing Problem with Simultaneous Pickup and Delivery

Goeke [14] considered pickup and delivery by EVs with a paired one-to-one service mode and stochastic battery depletion. He developed a granular tabu search (GTS) method and carried out numerical experiments on small and large cases, respectively. The experimental results proved that GTS is competitive on the benchmark instances of pickup and delivery problems with time windows. Soysal et al. [15] proposed a one-to-one pickup and delivery problem focusing on stochastic battery depletion and presented an approximated linear model for the problem. Yilmaz and Kalayci [16] studied the electric-vehicle-routing problem with simultaneous pickup and delivery. After testing several variable neighborhood search variants, they simplified the variable neighborhood algorithm and found a good solution in a reasonable time.

2.3. Elective-Vehicle-Routing Problem—Non-Linear Charging and/or Load-Dependent Discharging

Generally, the charging function is non-linear due to the changes in battery temperature, terminal voltage, current, etc. [17–21]. Montoya et al. [3] extended EVRP models to consider a non-linear charging function. They called this problem the electric vehicle routing problem with non-linear charging (EVRP-NL). They proposed a hybrid meta-heuristic algorithm and showed that ignoring non-linear charging may lead to solutions that are infeasible or too expensive. Since then, more scholars have started research on non-linear charging functions [5,22–25].

However, most scholars have ignored the complexity of the battery depletion process. In reality, load-dependent discharging is an ideal choice that is close to reality. Shao et al. [26] and Kancharla and Ramadurai [27] considered load-dependent discharging in the energy consumption equation for EVs. Zhang et al. [28] also presented the energy consumption of EVs in their model. They proposed a meta-heuristic algorithm based on an ant colony algorithm. The effectiveness of the proposed algorithm was evaluated through a large number of numerical experiments on the newly generated examples. Basso et al. [29] introduced a total-mass-dependent method to estimate the energy cost. From the numerical test of the road network in Gothenburg, Sweden, they presented that the time and energy estimates are more accurate than those found using existing methods. Soon, Kancharla, and Ramadurai [22] considered both NL and LD for the EVRP for the first time. They concluded that LD has a significant impact on the solution obtained. Although they considered both NL and LD, they did not consider simultaneous pickup and delivery.

2.4. Solution Methodology

Like other VRPs and their variants, EVRPSPD is an NP-hard combinatorial optimization problem. Thus, to efficiently deal with large-size EVRPSPD instances that arise in practice, most scholars focus on heuristic and metaheuristic methods that can produce high-quality solutions within limited computational times. Several popular methods of dealing with the EVRP and its variants are summarized and compared in Table 1.

Table 1. Popular algorithm/method to solve the EVRP and its variants.

Literature	Problem	Solution Methodology
Schneider et al. (2014) [30]	EVRP with time windows (EVRPTW)	Tabu search
Keskin and Çatay (2016) [31]	EVRPTW with partial recharge (EVRPTW-PR)	Adaptive large neighborhood search algorithm
Shao et al. (2017) [26]	Green-vehicle-routing problem with simultaneous pickup and delivery (G-VRPSPD)	Hyper-genetic algorithm
Kancharla and Ramadurai (2018) [27]	EVRP with load-dependent discharging (EVRP-LD)	Adaptive large neighborhood search algorithm
Verma (2018) [32]	EVRP with time windows, recharging stations and battery-swapping stations (EVRP-RS-BS-SS)	Genetic algorithm
Zhang et al. (2018) [28]	EVRP with recharging stations (EVRP-RS)	Ant colony
Koç et al. (2019) [33]	EVRP with shared charging stations	Adaptive large neighborhood search algorithm
Basso et al. (2019) [29]	EVRP with energy consumption	Linear programming solver, Bellman–Ford algorithm
Goeke (2019) [14]	Pickup and delivery problem with time windows (EV-PDPTW)	Granular tabu search
Kancharla and Ramadurai (2020) [22]	EVRP with non-linear charging and load-dependent discharging (EVRP-NL-LD)	Adaptive large neighborhood search algorithm
Löffler et al. (2020) [17]	EVRP with a single recharge (EVRP-SR)	Tabu search
Olgun et al. (2021) [9]	Green-vehicle-routing problem with simultaneous pickup and delivery (G-VRPSPD)	Iterative local search and variable neighborhood descent heuristics
Lee (2021) [23]	EVRP with load-dependent energy consumption (EVRP-LD)	Branch and price
Zang et al. (2022) [34]	EVRPTW with non-linear battery depreciation	Column generation
Kyriakakis et al. (2022) [35]	Capacitated-vehicle-routing problem with time windows (CVRPTW)	Hybrid tabu search and variable neighborhood descent algorithm
Yilmaz and Kalayci (2022) [16]	EVRP with simultaneous pickup and delivery (EVRPSPD)	Variable neighborhood search

In general, no scholars have integrated non-linear charging and load-dependent discharging with EVRPSPD so far. In EVRPSPD, EVs have both pickup and delivery requests, so the power consumption of EVs is more difficult to predict and the linear assumptions of discharging may produce large errors, which makes it necessary to consider NL and LD simultaneously.

3. Mathematical Programming Model

3.1. Modeling of Battery-Charging Functions

We used the piecewise linear approximation method to fit the non-linear characteristics of the charging process, which has also been used in [3,5,23–25]. Montoya et al. [3] proposed three types of CSs (slow, moderate, fast) and corresponding parameters. Figure 2 shows the piecewise linear approximation results of the battery-charging functions for a moderate facility, in which the solid line represents the original data and the dashed line represents the data after linear approximation. Linear approximation helps to construct linear programming models. Theoretically, if the curve is divided into infinite straight lines, the straight line and the curve completely overlap. Considering accuracy, computability, and curve slope variation, we divided the curve into three segments based on the percentage of power charged: the first 60% ($a_0 - a_1$), the middle 30% ($a_1 - a_2$), and the last 10% ($a_2 - a_3$). c_b and a_b represent the charging time and the power level for the breakpoint $b \in B$, respectively, where $B = \left\{0, \ldots, \overline{b}\right\}$ is the set of breakpoints of the piecewise linear approximation. The slope relationship of each part of the segment is that $\frac{a_1}{c_1} > \frac{a_2 - a_1}{c_2 - c_1} > \frac{a_3 - a_2}{c_3 - c_2}$.

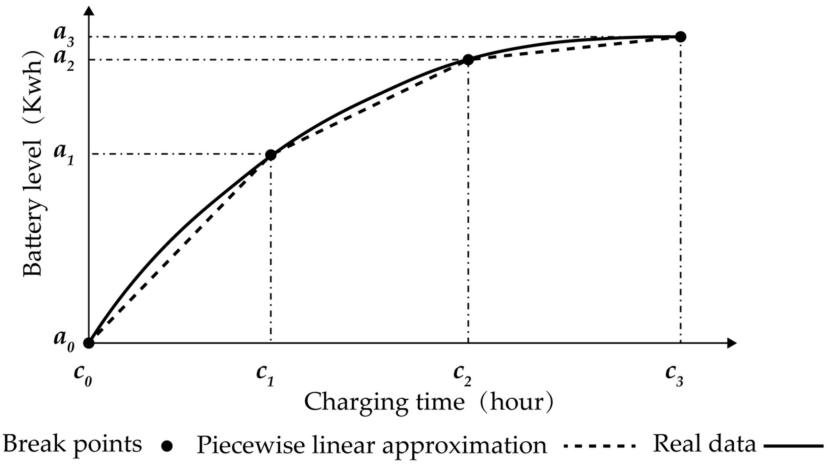


Figure 2. Piecewise linear approximation for the charging function.

3.2. Estimation of Power Required

The energy estimation model we used was derived from the comprehensive modal emission model (CMEM) [26,27,36]. The output engine power P (kW) of the model is as follows in Equation (1).

$$P = \frac{\left(Ma + Mgsin\theta + MgC_r cos\theta + 0.5C_d \rho A v^2\right)v}{1000\epsilon} \quad (1)$$

where the parameters of the formulation are described in Table 2.

Table 2. Parameters in the CMEM model.

Parameter	Description	Parameter	Description
v	Speed (m/s)	ρ	Air density (kg/m^3)
a	Acceleration (m/s^2)	A	Frontal surface area (m^2)
M	Gross vehicle weight (kg)	C_d	Coefficient of aerodynamic drag
g	Gravitational constant (m/s^2)	C_r	Coefficient of rolling resistance
θ	Longitudinal gradient of road	ϵ	Vehicle drive train efficiency

We assumed that all EVs were driving at a constant speed and $\theta = 0$. In addition, the acceleration start and deceleration stop of EVs were not taken into account in this paper. Hence, Equation (1) can be simplified to Equation (2).

$$P = \frac{(MgC_r + 0.5C_d\rho Av^2)v}{1000\epsilon} \tag{2}$$

Equation (2) is rewritten as a linear model of load in Equation (3).

$$P = \Phi_1 + \Phi_2 M \tag{3}$$

where $\Phi_1 = \frac{0.5C_d\rho Av^3}{1000\epsilon}$ is a constant and $\Phi_2 = \frac{gC_rv}{1000\epsilon}$ is the coefficient of weight, M.

In this paper, we used $\rho = 1.2041$, $C_d = 0.48$, $A = 2.3301$, $v = 40$, $\epsilon = 0.89$, $g = 9.81$, and $C_r = 0.01$, which are the same as presented in [22].

The values of Φ_1 and Φ_2 can be obtained by the data mentioned before. The only variable in Equation (3), the gross vehicle weight (M), is the sum of the vehicle weight (m) and the vehicle load weight (u).

3.3. Problem Description

A distribution center (depot) in a given area serves a given number of customers, each of whom may have both pickup and delivery requests. Each customer's pickup and delivery requests may have the same or different time windows. A certain number of EVs departs from the distribution center to visit customer nodes to fulfill the corresponding pickup and delivery requests of customers and returns to the distribution center. A certain amount of service time is required at each customer node.

In addition, throughout the journey, when the power of an EV runs low, it needs to visit a CS and spend a certain amount of time for charging. The charging time and the power charged satisfy a non-linear curve relationship. We assumed that the power consumption rate is non-constant during the journey and the power consumption mainly depends on the distance traveled and the weight of goods on board, which can be calculated using Equation (3).

We defined this problem as an electric-vehicle-routing problem with simultaneous pickup and delivery, considering non-linear charging and load-dependent discharging (EVRPSPD-NL-LD).

To better describe the problem, here is an example for further explanation. In the example, there are 1 distribution center (depot), 5 customers, and 2 CSs. The parameters of customers are shown in Table 3. Figure 3 provide a sample route and the optimal route, respectively, while the detailed route information is shown in Tables 4 and 5.

Table 3. The respective time window and cargo request data of the example.

Customer	1	2	3	4	5
Time window for pickup	[9.8, 10.3]	[9.4, 10.3]	[12.9, 13.5]	[4.0, 4.8]	[11.8, 12.6]
Time window for delivery	[7.6, 8.3]	[6.9, 7.9]	[5.8, 6.7]	[6.2, 6.8]	[1.5, 2.3]
Pickup (kg)	224	83	193	148	143
Delivery (kg)	219	101	136	141	208

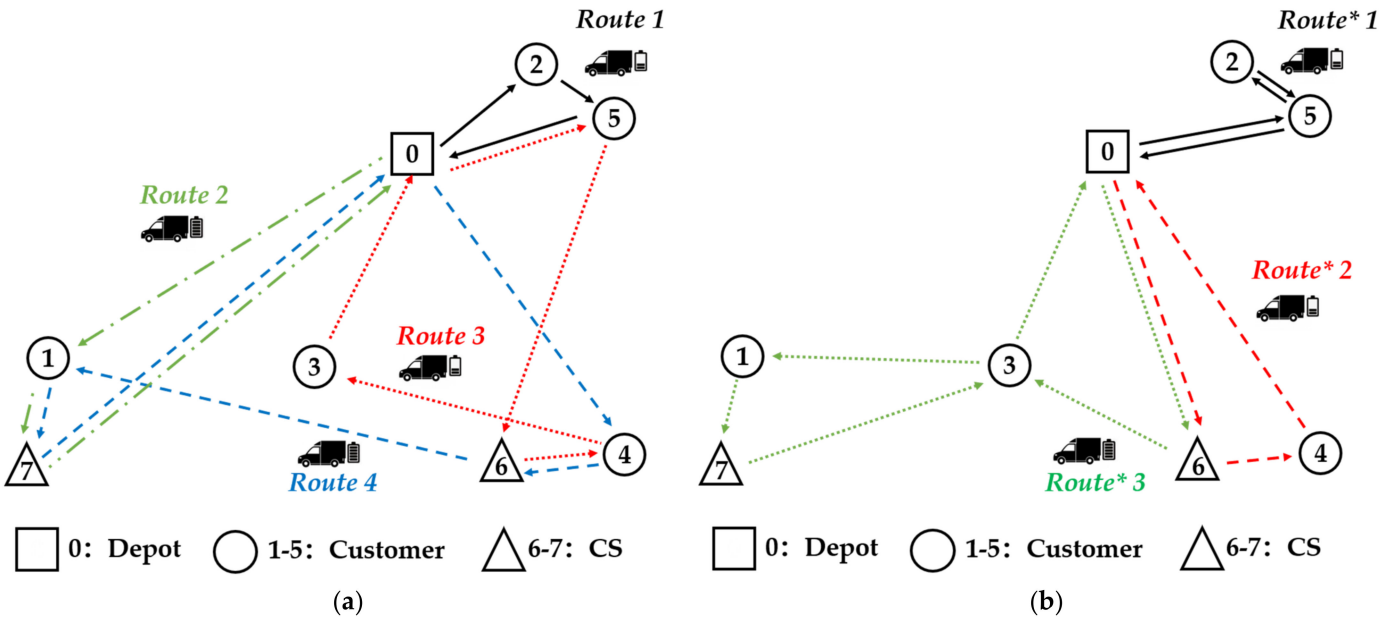


Figure 3. Sample route of (**a**) a feasible solution and (**b**) an optimal solution.

Table 4. Detailed route information of Figure 3a.

Vehicle	Route
Vehicle 1	0—2 (pickup and delivery)—5 (pickup)—0
Vehicle 2	0—1 (delivery)—7 (charging)—0
Vehicle 3	0—5 (delivery)—6 (charging)—4 (delivery)—3 (pickup and delivery)—0
Vehicle 4	0—4 (pickup)—6 (charging)—1 (pickup)—7 (charging)—0

Table 5. Detailed route information of Figure 3b.

Vehicle	Route
Vehicle 1	0—5 (delivery)—2 (pickup and delivery)—5 (pickup)—0
Vehicle 2	0—6 (charging)—4 (pickup and delivery)—0
Vehicle 3	0—6 (charging)—3 (delivery)—1 (pickup and delivery)—7 (charging)—3 (pickup)—0

3.4. Mathematical Formulation

We formulated EVRPSPD-NL-LD as a mixed integer linear program (MILP). The assumptions of the problem are as follows:

(1) A request, whether for pickup or delivery, is less than the capacity of an EV and can only be served once by one EV.
(2) The speed of the homogeneous fleet of EVs is fixed.
(3) EVs are allowed to arrive earlier than the time window, which means the EV will wait at the customer until the time window is met, but EVs cannot arrive later than the time window.
(4) Each EV can only be used once under the limited cruising duration.
(5) The CSs are homogeneous and can be visited multiple times.
(6) Each EV must depart from the depot and eventually return to the depot.

Let n denote the total number of customers. Because customers may have two requests, let $V = \{1, 2, \ldots, n, n+1, \ldots, 2n\}$ denote the set of requests (i.e., request i represents the pickup for customer i and request $n+i$ represent the delivery for customer i). Let $P = \{1, 2, \cdots, n\}$ and $D = \{n+1, \cdots, 2n\}$ denote the request of pickup and delivery,

respectively. Obviously, $V = P \cup D$. Let 0 and $N+1$ denote the depot. Let F and F' denote the set of CSs and their duplications. The nodes defined in this paper consist of depot, customers, and charging stations.

Thus, EVRPSPD-NL-LD can be defined on a complete directed graph $G = \left(V'_{0,N+1}, A\right)$, with the set of arcs $A = \left\{(i,j) \middle| i,j \in V'_{0,N+1}, i \neq j\right\}$. With each arc and node, parameters are summarized in Table 6.

Table 6. Parameters in the EVRPSPD-NL-LD model.

Parameter	Description
n	Total number of customers
V	Set of requests, $V = \{1, 2, \cdots, n, n+1, \cdots, 2n\}$
P	Set of pickup vertices, $P = \{1, 2, \cdots, n\}$
D	Set of delivery vertices, $D = \{n+1, \cdots, 2n\}$
F	Set of charging stations, $F = \{f_1, f_2, \cdots\}$
F'	Set of charging stations and their copies, $F' = \{f_1, f_2, \cdots, f'_1, f'_2, \cdots\}$
$0, N+1$	Depot
$V\prime$	$= V \cup F'$
V'_0	$= V \cup F' \cup \{0\}$
V'_{N+1}	$= V \cup F' \cup \{N+1\}$
$V'_{0,N+1}$	$= V \cup F' \cup \{0\} \cup \{N+1\}$
A	Set of arc (i,j), $i,j \in V'_{0,N+1}, i \neq j$
d_{ij}	Distance between vertices i and j, $i,j \in V'_{0,N+1}, i \neq j$ (km)
t_{ij}	Travel time between vertices i and j, $i,j \in V'_{0,N+1}, i \neq j$ (h)
p_i	Request of pickup for customer i, $i \in P$ (kg)
d_i	Request of delivery for customer $i-n$, $i \in D$ (kg)
R_i	$R_i \in \{p_1, \ldots, p_n, d_{n+1}, \ldots d_{2n}\}, i \in V$
C	Vehicle capacity (kg)
Q	Battery capacity (kwh)
B	Set of the breakpoints in charging function, $B = \left\{0, 1, \ldots, \bar{b}\right\}$
c_b	Time at breakpoint b, $b \in B$ (h)
a_b	Battery level at breakpoint b (kwh), $b \in B$
$[e_i, l_i]$	Time window of vertex i, $i \in V'_{0,N+1}$, where $e_0 = 0$ and l_0 is the maximum duration of process
s_i	Service time for vertex i, where $s_0 = s_N = 0, i \in V_{0,N+1}$ (h)
K	Set of vehicles, $K = \{k_1, k_2, \cdots\}$
Φ_1	Consumption rate with zero load
Φ_2	When the load increases by 1 kg, the consumption rate increases Φ_2
m	Weight of vehicle (kg)
M_c	A sufficiently large coefficient

We introduced variable x_{ij} to construct routes, route variable v_i to track the route, and variable q_{ij} to track the available capacity of a specific EV through the duration. The variables are summarized in Table 7.

Table 7. Variables in the EVRPSPD-NL-LD model.

Variable	Description
x_{ij}	=1, if arc (i,j) is traveled, otherwise 0
u_i	Load of EV when departing from node $i, i \in V'$
U_i	Load of EV when departing from depot to route i, $i \in V'$
q_{ij}	$= R_i$, if node i is in route j, $i,j \in V'$
τ_i	Arrival time of EV at node i, $i \in V'_{0,N+1}$
Tw_i	Waiting time before arriving at node $i, i \in V'$

Table 7. *Cont.*

Variable	Description
v_i	Remaining battery capacity on arrival at node i, $i \in V_{0,N+1}$
y_i/Y_i	Remaining battery capacity on arrival/departure at charging station i
o_i/O_i	Time required to charge to y_i/Y_i
θ_i	Charging time at charging station i
z_{ib}/w_{ib}	Auxiliary binary variable used to determine whether the arrival/departure power level is between breakpoint b and breakpoint $b-1$ in the piecewise linear approximation at CS i, $i \in F'$
α_{ib}/λ_{ib}	Auxiliary variable to enable the expression of $(y_i, o_i)/(Y_i, O_i)$ as a convex combination of the breakpoints $\{a_b, c_b\}$

The **EVRPSPD-NL-LD** model that we developed is as follows.
Minimize:

$$\sum_{j \in V'} M_c x_{0j} + \sum_{i \in V_0'} \sum_{j \in V_{N+1}', i \neq j} t_{ij} x_{ij} + \sum_{i \in V'} T w_i + \sum_{j \in F'} \theta_i + \sum_{j \in V'} s_i \tag{4}$$

Subject to:
Generating routes:

$$\sum_{j \in V_{N+1}', i \neq j} x_{ij} = 1, \forall i \in V \tag{5}$$

$$\sum_{j \in V_{N+1}', i \neq j} x_{ij} \leq 1, \forall i \in F' \tag{6}$$

$$\sum_{j \in V_{N+1}', i \neq j} x_{ij} - \sum_{j \in V_0', i \neq j} x_{ji} = 0, \forall i \in V' \tag{7}$$

$$r_j \geq j x_{0j}, \forall j \in V' \tag{8}$$

$$r_j \leq j x_{0j} - n\left(x_{0j} - 1\right), \forall j \in V' \tag{9}$$

$$r_j \geq r_i + n\left(x_{ij} - 1\right), \forall i, j \in V' \tag{10}$$

$$r_j \leq r_i + n\left(1 - x_{ij}\right), \forall i, j \in V' \tag{11}$$

Tracking store of battery:

$$v_0 = Q \tag{12}$$

$$v_0 - v_i - [\Phi_1(U_i + m) + \Phi_2] t_{0i} \geq (-Q - [\Phi_1(C + m) + \Phi_2] t_{0i})(1 - x_{0i}), \forall i \in V_{N+1}' \tag{13}$$

$$v_i - v_j - \left[\Phi_1\left(u_j + m\right) + \Phi_2\right] t_{ij} \geq \left(-Q - [\Phi_1(C + m) + \Phi_2] t_{ij}\right)\left(1 - x_{ij}\right), \forall i \in V, \forall j \in V_{N+1}', i \neq j \tag{14}$$

$$Y_i - v_j - \left[\Phi_1\left(u_j + m\right) + \Phi_2\right] t_{ij} \geq \left(-Q - [\Phi_1(C + m) + \Phi_2] t_{ij}\right)\left(1 - x_{ij}\right), \forall i \in F', \forall j \in V_{N+1}', i \neq j \tag{15}$$

$$y_i = v_i, \forall i \in F' \tag{16}$$

$$y_i = \sum_{b \in B} \alpha_{ib} a_b, \forall i \in F' \tag{17}$$

$$o_i = \sum_{b \in B} \alpha_{ib} c_b, \forall i \in F' \tag{18}$$

$$\sum_{b \in B} \alpha_{ib} = \sum_{b \in B} z_{ib}, \forall i \in F' \tag{19}$$

$$\sum_{b \in B} z_{jb} = \sum_{i \in V_0', i \neq j} x_{ij}, \forall j \in F' \tag{20}$$

$$\alpha_{i0} \le z_{i1}, \forall i \in F' \tag{21}$$

$$\alpha_{ib} \le z_{ib} + z_{ib+1}, \forall i \in F', \forall b \in B \backslash \left\{0, \overline{b}\right\} \tag{22}$$

$$\alpha_{i\overline{b}} \le z_{i\overline{b}}, \forall i \in F' \tag{23}$$

$$Y_i = \sum_{b \in B} \lambda_{ib} a_b, \forall i \in F' \tag{24}$$

$$O_i = \sum_{b \in B} \lambda_{ib} c_b, \forall i \in F' \tag{25}$$

$$\sum_{b \in B} \lambda_{ib} = \sum_{b \in B} w_{ib}, \forall i \in F' \tag{26}$$

$$\sum_{b \in B} w_{jb} = \sum_{i \in V_0', i \ne j} x_{ij}, \forall j \in F' \tag{27}$$

$$\lambda_{i0} \le w_{i1}, \forall i \in F' \tag{28}$$

$$\lambda_{ib} \le w_{ib} + w_{ib+1}, \forall i \in F', \forall b \in B \backslash \left\{0, \overline{b}\right\} \tag{29}$$

$$\lambda_{i\overline{b}} \le w_{i\overline{b}}, \forall i \in F' \tag{30}$$

$$\theta_i = O_i - o_i, \forall i \in F' \tag{31}$$

Tracking arrival time:

$$\tau_0 = 0 \tag{32}$$

$$e_i \le \tau_i \le l_i, \forall i \in V' \tag{33}$$

$$e_0 \le \tau_i + t_{iN+1} + s_i \le l_0, \forall i \in V' \tag{34}$$

$$\left(-2l_0 - t_{ij} - s_i\right)\left(1 - x_{ij}\right) \le \tau_j - \tau_i - t_{ij} - s_i - Tw_j \le \left(l_0 - t_{ij} - s_i\right)\left(1 - x_{ij}\right), \forall i \in V_0, \forall j \in V', i \ne j \tag{35}$$

$$\left(-2l_0 - t_{ij} - c_{\overline{b}}\right)\left(1 - x_{ij}\right) \le \tau_j - \tau_i - t_{ij} - \theta_i - Tw_j \le \left(l_0 - t_{ij}\right)\left(1 - x_{ij}\right), \forall i \in F', \forall j \in V', i \ne j \tag{36}$$

Tracking load on vehicle:

$$\left(-C - p_j\right)\left(1 - x_{ij}\right) \le u_j - u_i - p_j, \forall i \in V', j \in P, i \ne j \tag{37}$$

$$\left(-C - d_j\right)\left(1 - x_{ij}\right) \le u_i - u_j - d_j \le \left(C - d_j\right)\left(1 - x_{ij}\right), \forall i \in V', j \in D, i \ne j \tag{38}$$

$$-C\left(1 - x_{ij}\right) \le u_j - u_i, \forall i \in V', j \in F', i \ne j \tag{39}$$

$$(-C - p_i)(1 - x_{0i}) \le u_i - U_i - p_i, \forall i \in P \tag{40}$$

$$(-C - d_i)(1 - x_{0i}) \le U_i - u_i - d_i \le (C - d_i)(1 - x_{0i}), \forall i \in D \tag{41}$$

$$-C(1 - x_{0i}) \le u_i - U_i, \forall i \in F' \tag{42}$$

$$R_i - q_{ij} \le R_i |r_i - j|, \forall i, j \in V' \tag{43}$$

$$U_i = \sum_{j \in D} q_{ji}, \forall i \in V' \tag{44}$$

Domain of variables:

$$\begin{gathered} x_{ij} \in \{0,1\}, \forall (i,j) \in A, i \neq j \\ z_{ib} \in \{0,1\},\ w_{ib} \in \{0,1\}, \forall i \in F', b \in B \\ 0 \leq \Delta_{ij}^{+}, 0 \leq \Delta_{ij}^{-}, 0 \leq q_{ij} \leq R_i, \forall i \in V', j \in V' \\ 0 \leq U_i \leq C, \forall i \in V' \\ 0 \leq u_i \leq C, 0 \leq \tau_i \leq l_0, \forall i \in V_0' \\ 0 \leq y_i \leq Y_i \leq Q,\ 0 \leq o_i \leq O_i \leq c_{\overline{b}}, \forall i \in F' \\ 0 \leq v_i \leq Q, \forall i \in V_{0,N+1}' \\ 0 \leq \lambda_{ib} \leq 1, 0 \leq \alpha_{ib} \leq 1, \forall i \in F', b \in B \\ 0 \leq \theta_i \leq c_{\overline{b}}, \forall i \in F' \end{gathered} \tag{45}$$

The objective of minimizing the total number of EVs used and the total time spent, including travel time, charging time, waiting time, and service time, is defined in Equation (4). Constraint (5) enforces the connectivity of node visits for each EV, and Constraint (6) handles the connectivity of visits to CSs for each EV. Constraint (7) guarantees that each EV should leave the depot. Constraints (8)–(11) determine the routes of the requests. Constraint (12) indicates that the EVs are fully charged before departing from the depot. Constraints (13)–(15) ensure that the battery power never falls below 0 under the consideration of load-dependent consumption. Constraints (16)–(30) enforce the power level and charging time of EVs based on the piecewise linear approximation. Constraint (31) captures the time spent at any CS. We assumed that each EV departs the depot at $t = 0$, which is defined in Constraint (32). Constraint (33) guarantees that the arrival time is within the time window. Constraint (34) ensures that each route is within the maximum duration. Constraints (35) and (36) track the arrival time at nodes and prevent the subtours of each EV. Constraints (37)–(42) ensure that the load of each EV does not exceed its limit. Constraints (43) and (44) ensure that EVs fulfill the requirement from customers when leaving the depot. Constraints (45) depict the domains of the variables.

Note that the model contains a large positive number M_c. The upper bound of M_c can be calculated as follows:

$$M_c = \sum_{i \in V} l_i \tag{46}$$

The linear relaxation of formulations can be strengthened by using valid inequalities. We developed the valid inequalities derived from Koç and Karaoglan [37] to improve our lower bounds, which is shown in Equation (47).

$$\sum_{j \in V'} x_{0j} \geq veh_{min} \tag{47}$$

where veh_{min} denotes the lower bound of the number of EVs used.

Proposition 1. *If we remove a request from a route in a feasible solution, the new route will still be feasible.*

Proof. Obviously, the new route will satisfy the constraint of load and power level if we remove a request. If an EV arrives at the customer earlier than the time window, it will wait there until the customer is available. □

Proposition 2. *Given any feasible route, the time spent at CSs can be determined with the objective of minimizing the total working time.*

Proof. There are two cases when an EV arrives at a customer. □

Case 1. Inside the time window. Figure 4 shows a partial route when an EV arrives at a customer inside the time window, and no waiting time is needed to satisfy the customer's request, where CS and CS' denote the charging station, τ_i denotes the arrival time, t_{ij} denotes the travel time, and s_i denotes the service time. Assume that the power level of the EV arriving at the CS is Q_y and the power required to visit all requests between CS and CS' is Q_c. If $Q_y \geq Q_c$, it is unnecessary to charge the EV at the CS. If $Q_y < Q_c$, the EV must be charged at least $Q_c - Q_y$ at the CS. If the power level leaving the CS is $Q_c + \Delta Q$ (i.e., the power level is ΔQ when the EV arrives at the CS', $\Delta Q \geq 0$), then the power level leaving the CS' is Q_c' (the EV must charge at least $Q_c' - \Delta Q$ at the CS'). Figure 2 defines the relationship between power level and charging time. Thus, the total charging time can be calculated by the power level arriving and leaving the CS. Assume that the piecewise function is $y = f(x)$, where x denotes the time and y denotes the battery level. The slope relationship of each part of the segment is $\frac{a_1}{c_1} > \frac{a_2-a_1}{c_2-c_1} > \frac{a_3-a_2}{c_3-c_2}$. The inverse function of $y = f(x)$ is $x = f^{-1}(y)$, where the slope relationship of each part of the segment is $\frac{c_1}{a_1} < \frac{c_2-c_1}{a_2-a_1} < \frac{c_3-c_2}{a_3-a_2}$ in Figure 2. Thus, the total charging time T_c can be expressed as $f^{-1}(Q_c + \Delta Q) - f^{-1}(Q_y) + f^{-1}(Q_c') - f^{-1}(\Delta Q)$. We can get $\frac{dT_c}{d\Delta Q} = f^{-1\prime}(Q_c + \Delta Q) - f^{-1\prime}(\Delta Q)$. Due to $\frac{c_1}{a_1} < \frac{c_2-c_1}{a_2-a_1} < \frac{c_3-c_2}{a_3-a_2}$, when power levels $Q_c + \Delta Q$ and ΔQ are in the same segment, $\frac{dT_c}{d\Delta Q} = 0$; otherwise, $\frac{dT_c}{d\Delta Q} > 0$. So, no matter how ΔQ and Q_c change, $\frac{dT_c}{d\Delta Q} \geq 0$. This means that when ΔQ increases, T_c does not decrease. Therefore, we can conclude that when $\Delta Q = 0$, T_c takes the minimum value. Thus, EVs should run out of power when they arrive at CSs, and spend the least charging time to ensure enough power to arrive at the next CS.

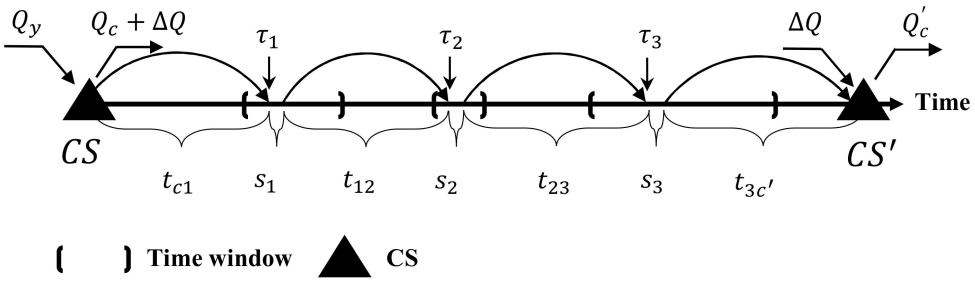


Figure 4. Example of case 1.

Case 2. Outside the time window. In this case, Figure 5a shows a partial route when an EV arrives at a customer outside the time window. When the EV leaves the CS, the power level is Q_c. When the EV arrives at Customer 1, it has to wait for Tw_1 in order to satisfy the time window requirement, and similarly for Request 3. Since Tw_1 and Tw_3 will not affect the time to reach CS', it is a good choice to make use of the waiting time to charge at the CS (see Figure 5b). We postponed the departure time in the CS by compressing Tw_1 and Tw_2 and obtained an extra t^* for charging. Using t^* for charging will reduce the charging time in CS' or subsequent stations and compress the total time spent.

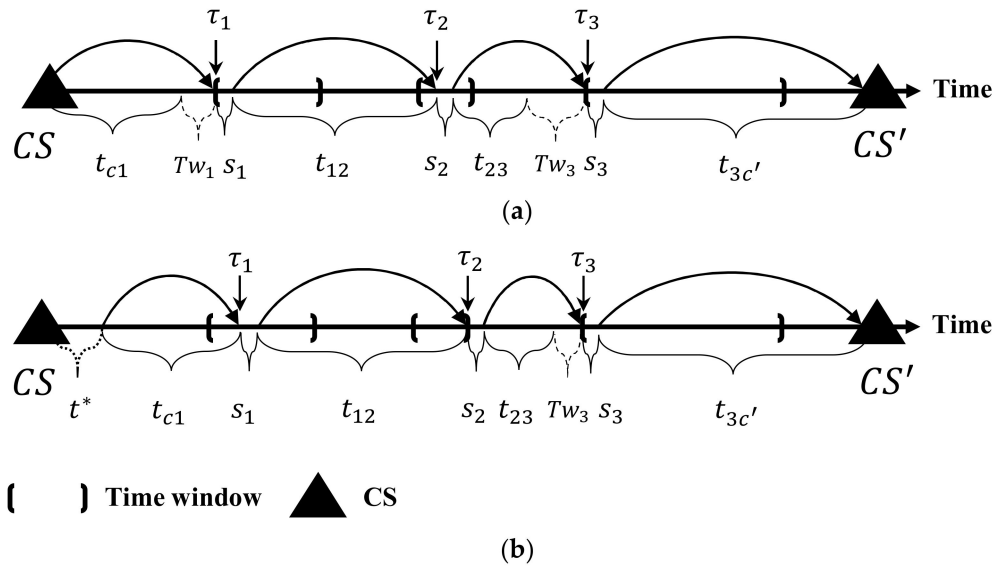


Figure 5. Example of case 2. (**a**) Not using waiting time and (**b**) using waiting time.

4. Solution Method

As a VRPSPD generalization, EVRPSPD is a NP-hard combinatorial optimization problem. Therefore, to effectively deal with the large-scale EVRPSPD instances that arise in practice, it is important to consider heuristic and meta-heuristic methods that can produce high-quality solutions in a limited computational time. We proposed an ALNS heuristic method to solve EVRPSPD-NL-LD. The ALNS was first introduced by Ropke and Pisinger [38] and improved by Keskin and Çatay [31] for solving EVRPs.

4.1. ALNS Approach

4.1.1. Initial Solution

The initial solution was generated by iteratively inserting feasible nodes. In each iteration, we selected a request from a customer and made an insertion to the current route with the least time spent. For the route obtained in every insertion, all constraints in Section 3 must be guaranteed to be satisfied. The feasibility check process of the solution is given in Pseudocode 1. If the route is infeasible due to the battery power level, we performed the repair operation. When no request from a customer could be inserted into the current route due to load or time window constraints, or a repair process could not be performed, the current route was completed. These steps were repeated to construct new routes until all requests from customers were met. The procedure of the initial solution construction is given in Pseudocode 2, and we provide an example of the process in Figure 6.

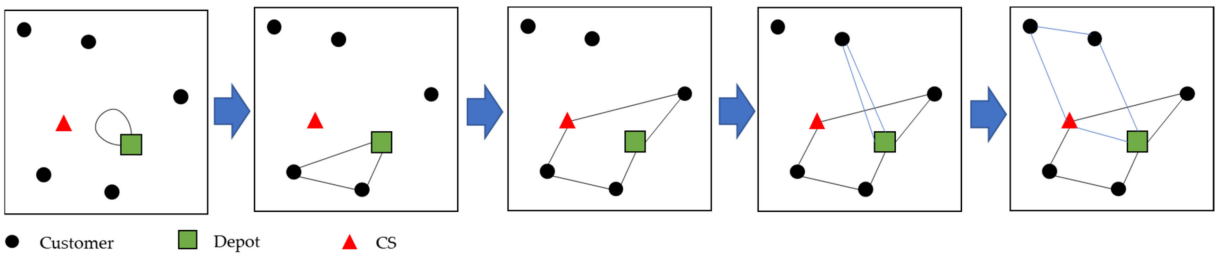


Figure 6. Example of initial solution generation.

Pseudocode 1: Check and Repair (*Route*)

```
If Route satisfies the constraints of load then
    If Route satisfies the constraints of battery level then
        If Route satisfies the constraints of time window then
            Return True and f(Route)
        Else
            Return False
    Else
        Repair Route
        If Route repaired satisfies the constraints of time window then
            Return True and f(Route repaired)
        Else
            Return False
Else
    Return False
```

Note: f(*Route*) is a function that can calculate the time spent for a given route.

Pseudocode 2: Initial Solution Generation

```
Initial CurrentRoute and BestTotalTime
While V is not empty do
For EachRequest in V do
For EachPosition in position where it can be inserted in CurrentRoute then
Insert EachRequest at EachPosition and obtain TempRoute
Check TempRoute in Pseudocode 1
If TempRoute is feasible and f(TempRoute) < BestTotalTime then
BestRoute ← TempRoute
BestTotalTime ← f(TempRoute)
If no request can be inserted to CurrentRoute then
InitialSolution append CurrentRoute
Add another initialized Route
CurrentRoute ← Route
Else
Remove request inserted in BestRoute from V
CurrentRoute ← BestRoute
```

4.1.2. ALNS Procedure

The ALNS heuristic method we proposed includes four classes of operations: Request Removal (*RR*), Request Insertion (*RI*), Station Removal (*SR*), and Station Insertion (*SI*). After the initial solution is obtained, the ALNS improves the solution iteratively within a limited number of iterations. In each iteration, the current feasible solution is destroyed by removing certain nodes, which consist of requests, CSs, or both, by using operator(s) in *RR* or *SR*. Next, the new solution after the nodes removed is repaired using operator(s) in *RI* or *SI* by inserting removed requests and necessary CSs to the destroyed routes in order to obtain new feasible solutions. Several removal and insertion operations are dynamically and adaptively applied depending on the performance of previous iteration cycles. The simulated annealing (SA) approach used in this paper is referenced from Xu [39]. The procedure of the ALNS heuristic method is provided in Figure 7.

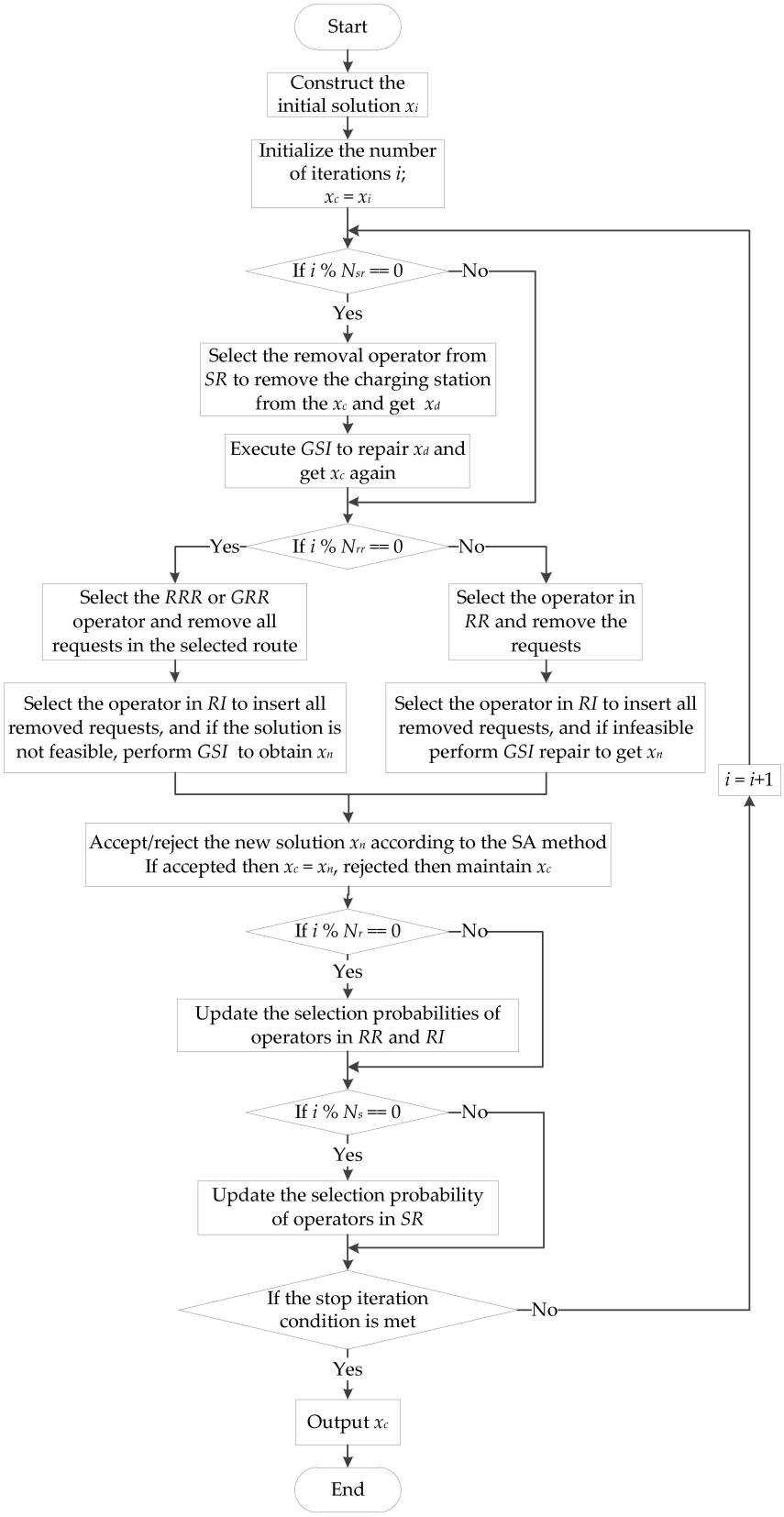


Figure 7. Flowchart of the ALNS.

4.2. Operators

The role of removal and insertion operators mentioned in Section 4.1.2 is to destroy current solutions and construct new solutions. The operators selected in this paper are summarized in Table 8. The operators in bold are new or improved operators developed in this paper.

Table 8. Description of removal and insertion operators.

Operation	Node	Operator	Description
Removal	Request (*RR*)	*Random*	The operator randomly chooses requests and removes them from the routes.
		Worst-Distance	The operator removes the request that increases the largest total distance.
		Worst-time	The operator removes the request with the highest difference between the arrival time of the EV and the earliest allowed arrival time.
		Shaw	The operator removes the request with high similarity.
		Proximity	The operator removes the request with high distance similarity.
		Time-based	The operator removes the request with a high similarity of arrival time.
		Demand	The operator removes the request with a high similarity of cargo weight.
		Zone	The operator removes all requests in a random area.
		Waiting time	The operator removes the request that most increased the waiting time.
	Station (*SR*)	*Random*	The operator randomly removes the CSs from the routes.
		Worst-Distance	The operator removes the CS that most increased the total distance.
		Worst-Charge Usage	The operator removes the CSs where the EV arrives with the maximum battery level.
		Full Charge	The operator removes the CSs where the EV is fully charged.
	Route (*RR′*)	*Random route removal (RRR)*	The operator randomly chooses a route and removes all the nodes visited in that route.
		Greedy route removal (GRR)	The routes are sorted in non-decreasing order of the number of customers serviced, and the routes ranked first are removed.
Insertion	Request (*RI*)	*Greedy*	The operator selects the best insertion position with the minimum total time.
		Regret-k	The operator calculates the difference between the total working time of the first and the *k*th-best insertion of the request and inserts the request with the largest difference to its optimal position.
		Waiting time	The operator inserts the request into the position with the least waiting time.
	Station (*SI*)	***Greedy station insertion (GSI)***	The operator insert a CS between the given starting position and the negative power position and chooses the position that requires the least time.

(1) *Shaw Removal Operator*: Φ_{ij} is changed as follows: if request i and request j belong to the same customer, $\Phi_{ij} = \phi_1 d_{ij} + \phi_2 |e_i - e_j| + \phi_3 l_{ij}$; otherwise, $\Phi_{ij} = \phi_1 d_{ij} + \phi_2 |e_i - e_j| + \phi_3 l_{ij} + \phi_4 |R_i - R_j|$. If request $i \in P$, $R_i = p_i$; otherwise, $R_i = -d_{n+i}$. Where ϕ_1, ϕ_2, ϕ_3, and ϕ_4 are the weights, $l_{ij} = -1$ if request i and j are in the same route, and 1 otherwise.

(2) *Waiting Time Removal Operator*: The waiting time cost at each request is calculated by the following equations:

$$cost_i = \begin{cases} e_i - \tau_{i-1} - t_{i,i-1} - s_{i-1}, i-1 \epsilon V \\ e_i - \tau_{i-1} - t_{i,i-1} - \theta_{i-1}, i-1 \epsilon F \prime \end{cases} \quad (48)$$

where $cost_i$ represents the cost of removing request i. The operator removes the request with the $\lfloor r\lambda^{\kappa} \rfloor$th largest cost where $\lambda \in [0, 1]$ is a random number and $\kappa \geq 1$ is a random parameter introduced to avoid repeated selection of the same request.

(3) *Waiting Time Insertion Operator*: The operator traverses all possible insertion positions of the requirement one by one and selects the position with the least increased waiting time as the insertion position to construct a new solution. Equation (48) calculates the increased waiting time:

$$\Delta t = f(x_n) - f(x_c) - T_n - S_n - C_n + T_c + S_c + C_c \quad (49)$$

where T_n, S_n, and C_n represent the travel time, service time, and charging time of the new route, respectively, and T_c, S_c, and C_c represent the travel time, service time, and charging time of the current route, respectively. Note that two or more CSs are likely to be removed in the *SR* operation. Thus, there is great potential that a destroyed solution will be feasible by inserting multi-stations instead of only one station. We proposed the multi-insertion of a station to repair the solution in a relatively efficient way, while in the literature, only one or two CSs could be accessed in a route [30,31].

(4) *Greedy Station Insertion Operator*: We demonstrated that when the route is determined, the charge of the EV is determined in Proposition 2. The *GSI* is provided in Pseudocode 3.

Pseudocode 3: Greedy Station Insertion (*Route*, *Rank*)

```
Global FeasibleRoute
Find position j where the battery level is negative and first appeared in Route
For i in range (Rank, j) do
    For CS in CSs do
        Get TempRoute by inserting CS into position i in Route
        If TempRoute is feasible then
            FeasibleRoute append TempRoute
        Else if TempRoute is still infeasible with battery level then
            Run Greedy Station Insertion (TempRoute, i) in given maximum
            recursive depth
Select the Route with least time spent among FeasibleRoute
```

Notes: Because the maximum number of removed stations in *SR* does not exceed 4, the maximum recursive depth we adopted is no more than 4. The parameter of *rank* is generally assigned 1 when it is called for the first time.

5. Numerical Experiments

The ALNS designed in this paper was compiled using Python 3.7.1 provided by Spyder 4.1 in Anaconda. The MILP solver was IBM ILOG CPLEX Optimization Studio V12.10. All numerical tests were executed on Inter Processor i7-7700 @ 3.60 GHz and 16 GB memory.

5.1. Data Preparation

The dataset we used derives from the real data of EVRP-NL presented in Montoya et al. [3]. We selected 50 instances that were divided into four subsets with 10, 20, 40, and 80 customers' requests, respectively. We defined 10 and 20 requests as small-size instances and 40 and 80 requests as large-size instances. The instances in each subset differed by the number of available CSs (low and high availability), the type of charging technology (slow, moderate, and fast), and the distribution modes of customer nodes (random distribution, cluster distribution, and combination of the two), which were distinguished by the letters R, C, and RC, respectively. In all instances, the EVs had a battery power capacity of 16 kWh, and the maximum time in a route was 10 h.

To prepare the data for our EVRPSPD-NL-LD problem, we first determined the customers' pickup and delivery requests, which were generated by a random uniform distribution between [50, 250]. The maximum load capacity of an EV was 600, which is the same

as that mentioned by Kancharla and Ramadurai [22]. Given that each customer may be served twice at most for pickup and deliver requests, the maximum duration was increased by 50% compared to the previous 10 h mentioned in [3]. The time window $[e_i, l_i], i \in V$ was set according to two criteria: (1) there is at least one EV that can fulfill the request in the time window and (2) the duration of time window t_d matches a random uniform distribution between [0.5, 1]. The new instances were named after the customer distribution rules, the number of requests, and the number of CSs. Take R01r10s2 as an example, where R01 represents example 01 with random distribution, r10 represents 10 requests, and s2 represents 2 CSs. An example of dataset R01r10s2 is shown in Table 9.

Table 9. An example of dataset R01r10s2.

Type	Coordinate (km)	Request of Pickup (kg)	Request of Delivery (kg)	Time Window of Pickup (h)	Time Window of Delivery (h)	Service Time (h)
Depot	(0, 0)	0	0	[0, **15**]	[0, **15**]	0
Customer	(12.44, 47.61)	**84**	**57**	**[2.1, 2.67]**	**[4.59, 5.44]**	0.16
Customer	(108.2, 93.64)	**65**	**153**	**[12.64, 13.17]**	**[6.31, 6.87]**	0.16
Customer	(58.51, 24.22)	**180**	**249**	**[11.07, 11.9]**	**[8.85, 9.49]**	0.16
Customer	(116.26, 13.76)	**135**	**147**	**[3.79, 4.3]**	**[4.11, 4.77]**	0.16
Customer	(113.56, 85.73)	**233**	**213**	**[5.52, 6.22]**	**[9.33, 9.91]**	0.16
Customer	(5.09, 69.91)	**60**	**111**	**[1.76, 2.27]**	**[6.37, 7.29]**	0.16
Customer	(89.27, 58.94)	**85**	**165**	**[2.98, 3.89]**	**[2.46, 3.09]**	0.16
Customer	(78.54, 57.16)	**87**	**236**	**[4.51, 5.22]**	**[1.93, 2.91]**	0.16
Customer	(111.56, 51.2)	**238**	**234**	**[4.77, 5.33]**	**[2.34, 3.34]**	0.16
Customer	(11.05, 72.52)	**167**	**106**	**[10.12, 10.81]**	**[13.3, 14.17]**	0.16
Station	(93.39, 10.8)	0	0	[0, **15**]	[0, **15**]	0
Station	(8.9, 10.98)	0	0	[0, **15**]	[0, **15**]	0

Notes: The numbers in bold were added or modified compared to the instances presented by Montoya et al. [3].

The parameters of the linear piecewise approximation function shown in Figure 2 were $[c_0, c_1, c_2, c_3] = [0,\ 0.62,\ 0.77,\ 1.01]$ and $[a_0, a_1, a_2, a_3] = [0,\ 13.6,\ 15.2,\ 16]$, respectively. According to the description in Section 3.2, $\Phi_1 = 1.03809$ and $\Phi_2 = 0.0012248$.

5.2. Comparison of CPLEX and the ALNS Heuristic Method

In Table 10, we provide the results for 10-request and 20-request instances. For CPLEX and our ALNS heuristic method, we provided the CPU time in seconds in the "Time (s)" and "$\text{Time}_{\text{Avg.}}$ (s)" column, respectively. Since the CPU time might be large to solve the instances, we limited the CPU time of CPLEX to 1200 s for 10-request instances and 3600 s for 20-request instances. We reported the optimal solution or the solution at the end of the time limit for the number of vehicles in the "k" column and the total time spent of all EVs in the "f_{best}" column. The "$f_{Avg.}$" column denotes the average objective value (total time spent of all EVs) in 10 runs, while the Δ_{fb} column denotes the gap to the objective found by CPLEX and Δ_{fa} denotes the gap between f_{best} and $f_{Avg.}$ in the ALNS.

Table 10. Results obtained by CPLEX and ALNS on 10-request and 20-request instances.

Instance	CPLEX			ALNS					
	k	f_{best} (I)	Time (s)	k	f_{best} (II)	$\text{Time}_{\text{Avg.}}$ (s)	$\Delta_{fb}=\frac{\text{II}-\text{I}}{\text{II}}$ (%)	$f_{Avg.}$ (III)	$\Delta_{fa}=\frac{\text{III}-\text{II}}{\text{III}}$ (%)
R01r10s2	3	**38.95**	323	3	**38.95**	15	0	38.95	0
R02r10s2	3	**38.35**	1200	3	**38.35**	18	0	38.35	0
R01r10s3	3	33.7	1200	3	**33.4**	89	−0.9%	33.4	0
R02r10s3	4	**52.48**	1200	4	**52.48**	31	0	52.48	0

Table 10. *Cont.*

Instance	CPLEX			ALNS					
	k	f_{best} (I)	Time (s)	k	f_{best} (II)	Time$_{Avg.}$ (s)	$\Delta_{fb}=\frac{II-I}{II}$ (%)	$f_{Avg.}$ (III)	$\Delta_{fa}=\frac{III-II}{III}$ (%)
C01r10s2	2	25.71	1200	2	**25.56**	20	0	25.56	0
C02r10s2	4	**44.12**	1200	4	**44.12**	36	0	44.12	0
C03r10s2	2	**22.8**	1200	2	**22.8**	17	0	22.8	0
C04r10s2	3	33.53	1200	3	**33.13**	35	−1.2%	33.13	0
C05r10s2	3	**36.37**	651	3	**36.37**	31	0	36.37	0
C01r10s3	3	**25.17**	1200	2	**25.17**	17	0	25.17	0
C02r10s3	3	**31.71**	1200	3	**31.71**	81	0	31.71	0
C03r10s3	2	**23.8**	1200	2	**23.8**	42	0	23.8	0
C04r10s3	3	30.8	1200	3	**30.77**	83	−0.1%	30.77	0
C05r10s3	3	**41.63**	840	3	**41.63**	78	0	41.63	0
RC01r10s3	2	**22.15**	1200	2	**22.15**	19	0	22.15	0
RC02r10s3	2	**24.43**	1200	2	**24.43**	65	0	24.43	0
R01r20s2	6	**58.63**	3600	6	**58.63**	93	0	59.5	1.5%
R02r20s2	4	**47.27**	3600	4	**47.27**	76	0	48.57	2.7%
R01r20s3	5	**59.9**	3600	5	**59.9**	207	0	61.45	2.5%
R02r20s3	5	46.93	3600	4	**43.48**	225	−7.9%	45.54	4.5%
C01r20s2	4	44.85	3600	4	**42.24**	60	−5.9%	43.16	2.1%
C02r20s2	6	70.6	3600	4	**51.13**	72	−38.1%	55.68	8.2%
C03r20s2	5	55.37	3600	5	**48.5**	82	−14.2%	49.94	2.9%
C04r20s2	4	**42.78**	3600	4	**42.78**	61	0	46.67	8.3%
C01r20s3	6	61.43	3600	5	**55.04**	108	−11.6%	56.3	2.2%
C02r20s3	4	41.61	3600	4	**40.04**	123	−3.9%	42.9	6.7%
C03r20s3	4	45.8	3600	4	**45.65**	91	−0.3%	46.59	2.0%
C04r20s3	4	48.55	3600	4	**47.77**	140	−1.6%	48.6	1.7%
RC01r20s3	4	**50.68**	3600	4	**50.68**	46	0	52.1	2.7%
RC02r20s3	3	38.05	3600	3	**37.31**	145	−2.0%	37.71	1.1%
Avg.	-	-	-	-	-	73.53	−2.92%	-	1.6%

The results showed that CPLEX can only solve 3 of 20 instances to optimality within the time limit, while our ALNS method outperformed the direct solution of CPLEX in 13 of 20 instances, and the same results were obtained for the remaining 7 instances. The average CPU time of the ALNS was 73.53 s, while CPLEX could not solve 17 of 20 instances within the time limit. The average gap of the objective value was around −2.92%, which is shown in the "Δ_{fb}" column. The average gap between f_{best} and $f_{Avg.}$ shown in the "Δ_{fa}" column was 1.6%, which demonstrates the stability of the ALNS to solve small-size instances. Although the solution result of the ALNS in a small-size problems was basically the same as that of CPLEX, we still recommend CPLEX because the solution result of CPLEX can be guaranteed to be the optimal solution.

5.3. ALNS with/without New Operators and the Time Priority Approach

The new operators we proposed include *waiting time removal*, *waiting time insertion*, improved *shaw removal*, and *greedy station insertion* operators, which are presented in Section 4.2. To reduce the CPU time of the ALNS for practical situations, we developed a time priority approach, in which the upper limit of the number of removal requests was reduced to 20% and the total number of iterations decreased to 50%.

We compare the CPU time and solutions by CPLEX and the ALNS with/without new operators for 40-request instances in Table 11. Table 12 provides the results of the ALNS with/without new operators and the time priority approach for 80-request instances. The "$k_{Avg.}$" column denotes the average number of EVs used in 10 runs in Tables 11 and 12, while the "Δ_{ft}"column denotes the gap of CPU time between the ALNS with new operators and the time priority approach in Table 12. Better results are highlighted in the tables.

Table 11. Results obtained by CPLEX and the ALNS with/without new operators on 40-request instances.

Instance	CPLEX			ALNS										
				Without New Operators					With New Operators					
	k	f_{best}	Time (s)	k	$k_{Avg.}$	f_{best}(I)	$f_{Avg.}$	$\text{Time}_{\text{Avg.}}$ (s)	k	$k_{Avg.}$	f_{best}(II)	$\Delta_{fb}=\frac{\text{II}-\text{I}}{\text{II}}$(%)	$f_{Avg.}$(III)	$\Delta_{fa}=\frac{\text{III}-\text{II}}{\text{III}}$(%)	$\text{Time}_{\text{Avg.}}$ (s)
R01r40s3	N/A	N/A	3600	9	9.2	**99.43**	106.1	157	9	**9.1**	99.71	0.3%	105.78	5.7%	141
R02r40s3	N/A	N/A	3600	9	9.6	101.19	107.28	221	9	**9.5**	**99.85**	−1.3%	108.82	8.2%	171
R01r40s4	N/A	N/A	3600	8	8.8	104.45	109.85	455	8	**8.4**	**102.27**	−2.1%	106.85	4.3%	401
R02r40s4	N/A	N/A	3600	7	7.8	**86.05**	94.69	161	7	**7.5**	86.66	0.7%	92.45	6.3%	255
C01r40s3	N/A	N/A	3600	8	8.3	91.56	96.89	101	8	**8.1**	**90.91**	−0.7%	96.26	5.6%	103
C02r40s3	N/A	N/A	3600	7	8	88.8	94.23	169	7	**7.9**	**88.72**	−0.1%	93.65	5.2%	151
C03r40s3	N/A	N/A	3600	8	8.2	91.5	95.98	114	**7**	**8.1**	**86.33**	−6.0%	93.05	7.2%	113
C03r40s3	19	205.37	3600	7	7.3	87.53	91.2	126	7	**7.3**	**87.52**	−0.0%	90.64	3.4%	170
C05r40s3	N/A	N/A	3600	8	**8**	80.37	84.26	101	8	**8**	**79.76**	−0.8%	83	3.9%	109
C06r40s3	N/A	N/A	3600	8	8.7	**92.35**	100.24	54	8	**8.4**	92.89	0.6%	98.23	5.4%	78
Avg.	-	-	-	-	-	-	-	165.9	-	-	-	−0.9%	-	5.5%	169.2

Table 12. Results of ALNS with/without new operators, and time priority approach on 80-request instances.

Instance	ALNS																	
	Without New Operators					With New Operators						Time Priority Approach						
	k	$k_{Avg.}$	f_{best}(I)	$f_{Avg.}$	$\text{Time}_{\text{Avg.}}$ (s)	k	$k_{Avg.}$	f_{best}(II)	$\Delta_{fb}=\frac{\text{II}-\text{I}}{\text{II}}$(%)	$f_{Avg.}$(III)	$\Delta_{fa}=\frac{\text{III}-\text{II}}{\text{III}}$(%)	$\text{Time}_{\text{Avg.}}$ (s) (IV)	k	$k_{Avg.}$	f_{best}(V)	$\Delta_{fb}=\frac{\text{V}-\text{II}}{\text{V}}$(%)	$f_{Avg.}$	$\text{Time}_{\text{Avg.}}$ (s) (VI)	$\Delta_{ft}=\frac{\text{IV}-\text{VI}}{\text{IV}}$(%)
R01r80s5	16	16.9	193	197.18	294	16	**16.5**	**186.75**	−3.3%	196.18	4.8%	324	16	17.5	194.15	3.8%	202.82	33	89.8%
R02r80s5	15	15.3	173.27	181.37	793	**14**	15.1	**173.71**	0.3%	180.77	3.9%	713	14	**15**	178.91	2.9%	187.32	47	93.4%
R03r80s5	16	**16.4**	183.5	191.58	439	16	16.5	**178.46**	−2.8%	190.22	6.1%	561	16	16.9	189.71	5.9%	196.2	38	93.2%
R04r80s5	16	16.2	188.66	195.12	465	16	**16.2**	**184.94**	−2.0%	195.25	5.2%	536	16	16.4	189.66	2.5%	193.91	45	91.6%
R01r80s8	14	**14.6**	165.24	172.06	1291	14	14.7	**164.46**	−0.5%	171.63	4.2%	1031	15	15.5	174.69	5.9%	182.65	137	86.7%
R02r80s8	14	15.3	172.99	173.96	677	14	**15.1**	**162.8**	−6.3%	171.79	5.2%	856	15	15.2	168.23	3.2%	174.66	97	88.6%
R03r80s8	16	17	184.24	194.01	582	16	**16.6**	**177.02**	−4.1%	189.45	6.6%	652	16	17	184.06	3.8%	195.56	83	87.2%
R04r80s8	16	**16.2**	189.19	195.67	735	16	**16.2**	**188.56**	−0.3%	193.57	2.6%	664	16	16.8	196.68	4.1%	202.11	89	86.6%
RC01r80s5	17	17.9	208.87	221.28	317	17	**17.4**	**206.96**	−0.9%	220.06	6.0%	418	17	17.8	213.33	3.0%	225.92	63	84.9%
RC02r80s5	15	15.9	**163.78**	172.62	351	15	**15.2**	163.89	0.1%	167.22	2.0%	336	15	15.6	173.38	5.5%	178.34	27	92.0%
Avg.	-	-	-	-	594	-	-	-	−2.0%	-	4.7%	609.1	-	-	-	4.1%	-	65.9	89.4%

As can be seen from Table 11, CPLEX could not solve the problems to optimality in all 10 instances within the specified time limit of 3600 s, while the average CPU time of the ALNS with and without new operators was 169.2 and 165.9 s, respectively.

Tables 11 and 12 show that the ALNS with new operators achieves better solutions of $k_{Avg.}$ for all 10 40-request instances and 7 of 10 80-request instances. Meanwhile, the ALNS with new operators solved better $f_{Avg.}$ for 7 of 10 40-request instances and 9 of 10 80-request instances. The average gap in $f_{Avg.}$ was -0.9% for 40-request instances and -2.0% for 80-request instances, which demonstrates that the new operators have good results for large-size instances on average. The average gap between f_{best} and $f_{Avg.}$ was 5.5% on 40-request instances and 4.7% on 80-request instances.

Table 12 also shows a significant improvement in CPU time with our time priority approach. The average CPU time to solve 80-request instances for the time priority approach was only 65.9 s compared with 594 s for the ALNS without new operators and 609.1 s for the ALNS with new operators. The average gap of $f_{Avg.}$ between the ALNS with new operators and the time priority approach was 4.1%, which is not significant. Therefore, if the problem needs to be solved in a short amount of time, the time priority approach may be a good choice.

Figure 8 depicts the analysis of solution results of the ALNS with and without new operators and the time priority approach for 40- and 80-request instances, respectively. It is obvious that the ALNS with new operators performed better for $f_{Avg.}$ and f_{best} for both 40- and 80-request instances. For the 80-request instances, the time priority approach performed the worst but it reduced the CPU time by around 90%.

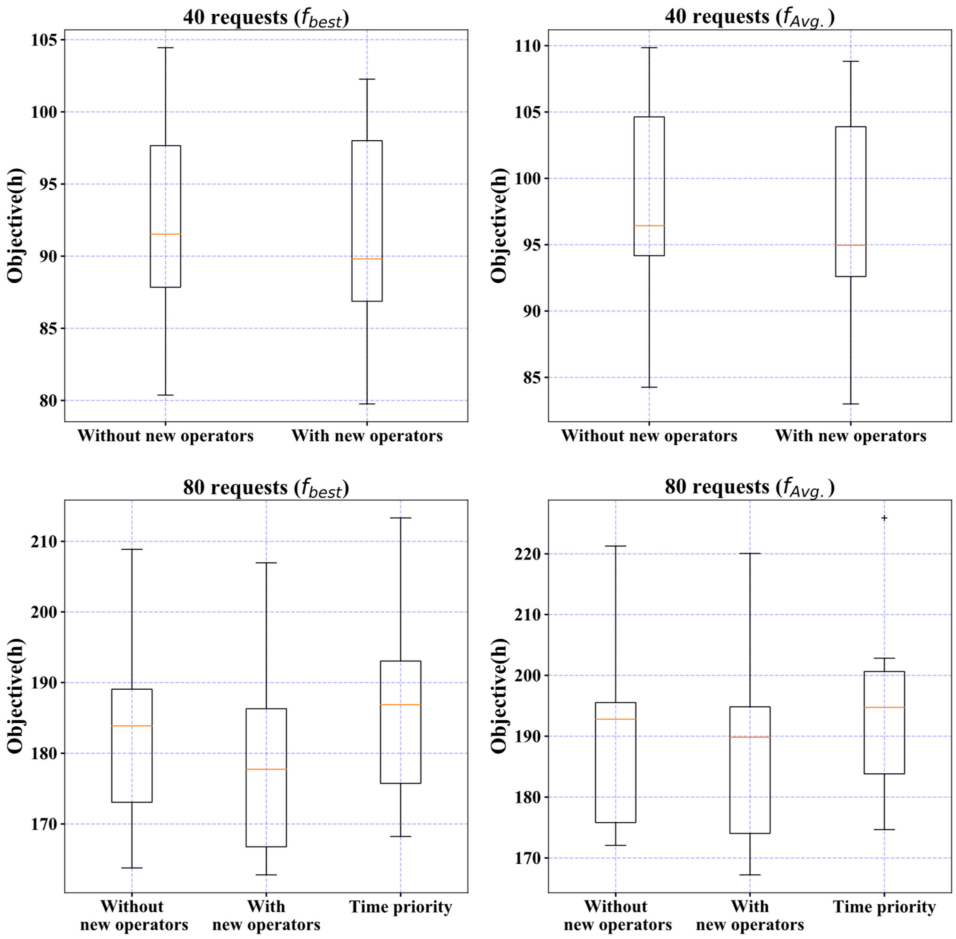


Figure 8. Comparison of solution results of the ALNS with/without new operators and the time priority approach.

Figure 9 shows the metric analysis of the solution results for the different requests. The metrics include the total number of visits to charging stations, the total waiting time, and the total charging time. As the size of the problem increased, the number of visits to charging stations and the total charging time grew exponentially, while the total waiting time grew linearly. As the number of customers increased, the number of time windows increased accordingly. EVs may be fully used without increasing much waiting time when the routes are optimized with the time windows reasonably arranged; however, an increase in power consumption and a corresponding increase in the number of visits to charging stations will increase much faster.

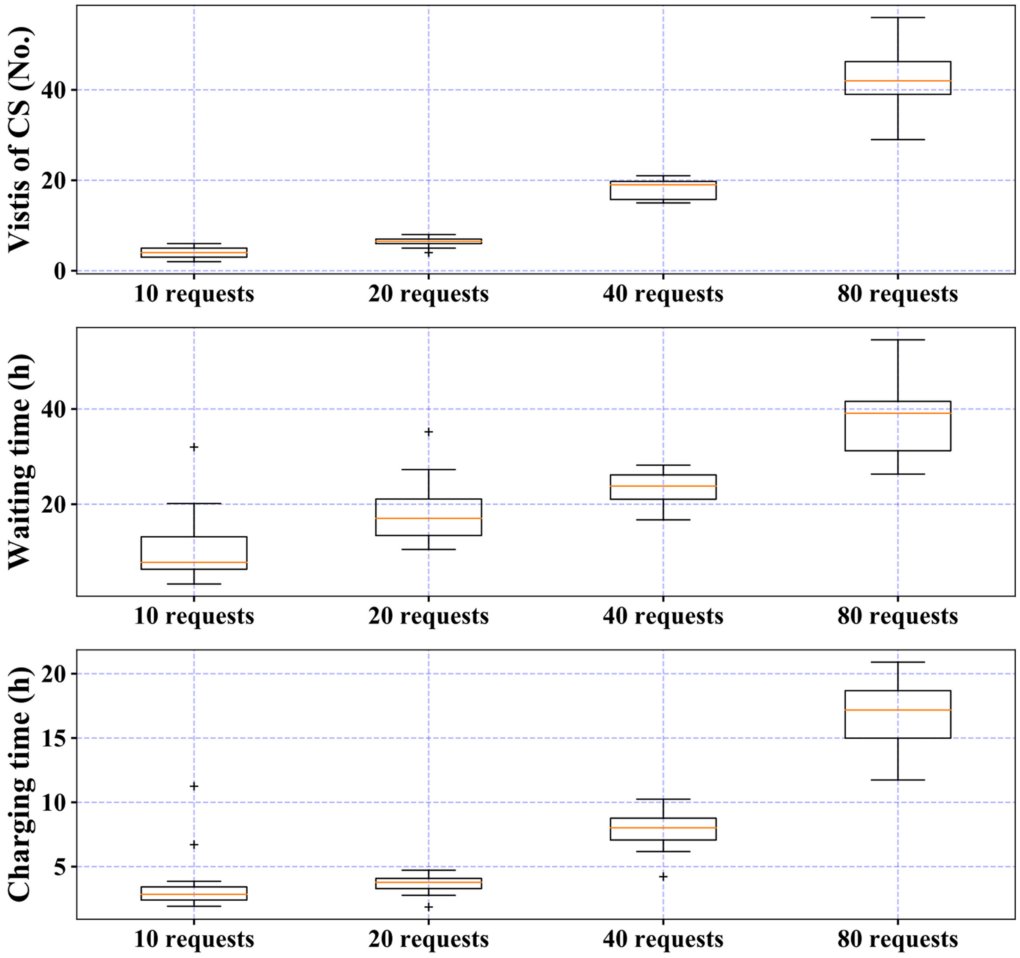


Figure 9. Visits to CSs, waiting times, and charging times at instances of different sizes.

5.4. Sensitivity Analysis

Sensitivity analysis was performed to investigate the relationship between the number of iterations in the ALNS and the quality of the results obtained. We studied the number of iterations in the ALNS for 80-request instances of 250, 500, 750, and 1000. The average results, which are the average CPU times, the average gap of number of vehicles, and the total time spent, are shown in Figure 10. As the number of iterations increased, the average CPU time increased accordingly from 205 s for 250 iterations to 609 s for 1000 iterations. However, the average gap increased by 5.6% for the number of vehicles and 6.2% for the total time spent when the number of iterations decreased to 250. Sensitivity analysis may also provide management with a trade-off between the quality of the solution and the CPU time to obtain the solution.

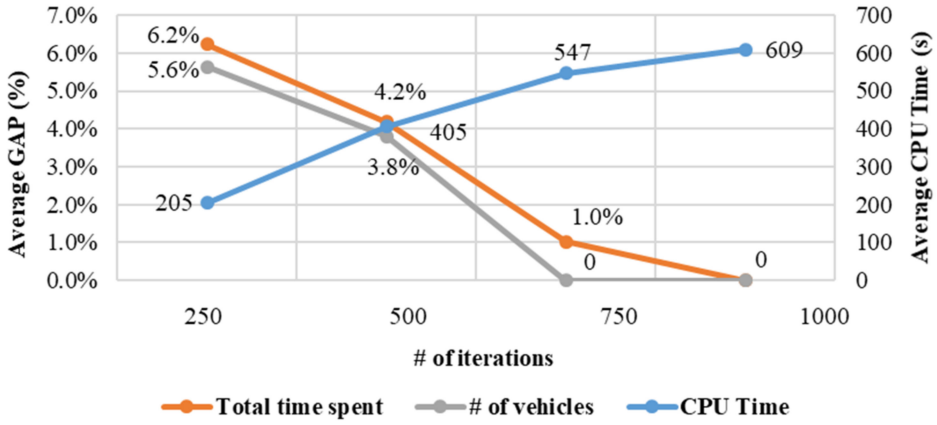


Figure 10. Sensitivity analysis of ALNS parameters.

6. Conclusions

In this paper, we addressed an electric-vehicle-routing problem with simultaneous pickup and delivery considering non-linear charging and load-dependent discharging (EVRPSPD-NL-LD), with the objectives of minimizing the number of EVs and the total working time of all EVs. First, we developed a mixed integer linear programming model for this problem. Due to the difficulty of solving the model for large-size instances directly using IBM ILOG CPLEX Optimization Studio (CPLEX), we developed an adaptive large neighbourhood search (ALNS) heuristic method. Compared with the traditional ALNS, we introduced four new or improved operators (*waiting time removal*, *waiting time insertion*, *shaw removal*, and *greedy station insertion*), which are suitable for our non-linear charging and load-dependent discharging scenarios. To obtain good results in a small amount of time, we followed a time priority approach by removing some operators and reducing the number of iterations. We modified the instances in Montoya et al. [3] by adding pickup and delivery requests and corresponding time windows. In total, 50 small- and large-size problems were generated to test our mathematical programming model and the ALNS heuristic method.

Our computational investigation and sensitivity analysis revealed that (i) in most cases, the ALNS with new operators outperforms the traditional ALNS under the objectives discussed in this paper, (ii) our ALNS heuristic method has advantages over CPLEX not only in the quality of the solutions but also in the CPU time, (iii) the time priority approach can reduce the CPU time by around 90% with a gap of less than 5%, and (iv) sensitivity analysis can help management to make decisions.

Further research can be conducted by relaxing some assumptions, such as a homogeneous fleet, or by considering other objective functions. It would be also interesting to investigate a mathematical-programming-based solution method, such as column generation or Bender's decomposition.

Author Contributions: Conceptualization, M.C. and C.Z.; methodology, C.Z.; software, C.Z.; validation, W.X. and M.C.; investigation, W.X.; resources, W.X. and Y.H.; data curation, W.X. and C.Z.; writing—original draft preparation, W.X. and C.Z.; writing—review and editing, M.C. and Y.H.; visualization, C.Z.; supervision, M.C. and Y.H.; project administration, M.C.; funding acquisition, M.C. and Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the Nature Science Foundation of Jiangsu Province (no. BK20150344) and the China Postdoctoral Science Foundation (no. 2016M601885).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, Z.; Ching, T.W.; Huang, S.; Wang, H.; Xu, T. Challenges faced by electric vehicle motors and their solutions. *IEEE Access* **2021**, *9*, 5228–5249. [CrossRef]
2. United Nations Environment Programme. *Emissions Gap Report 2021: The Heat Is On—A World of Climate Promises Not Yet Delivered*; United Nations Digital Library: Nairobi, Kenya, 2021; pp. 38–43. Available online: https://www.unep.org/resources/emissions-gap-report-2021 (accessed on 20 February 2022).
3. Montoya, A.; Guéret, C.; Mendoza, J.E.; Villegas, J.G. The electric vehicle routing problem with nonlinear charging function. *Transp. Res. Part B Methodol.* **2017**, *103*, 87–110. [CrossRef]
4. Min, H. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transp. Res. Part A Gen.* **1989**, *23*, 377–386. [CrossRef]
5. Froger, A.; Mendoza, J.E.; Jabali, O.; Laporte, G. Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. *Comput. Oper. Res.* **2019**, *104*, 256–294. [CrossRef]
6. Koyuncu, I.; Yavuz, M. Duplicating nodes or arcs in green vehicle routing: A computational comparison of two formulations. *Transp. Res. Part E Logist. Transp. Rev.* **2019**, *122*, 605–623. [CrossRef]
7. Parragh, S.N.; Doerner, K.F.; Hartl, R.F. A survey on pickup and delivery models part ii: Transportation between pickup and delivery locations. *J. Für Betr.* **2008**, *58*, 81–117. [CrossRef]
8. Zachariadis, E.E.; Kiranoudis, C.T. A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. *Expert Syst. Appl.* **2011**, *38*, 2717–2726. [CrossRef]
9. Olgun, B.; Koç, Ç.; Altıparmak, F. A hyper heuristic for the green vehicle routing problem with simultaneous pickup and delivery. *Comput. Ind. Eng.* **2021**, *153*, 107010. [CrossRef]
10. Yu, V.F.; Susanto, H.; Yeh, Y.-H.; Lin, S.-W.; Huang, Y.-T. The Vehicle Routing Problem with Simultaneous Pickup and Delivery and Parcel Lockers. *Mathematics* **2022**, *10*, 920. [CrossRef]
11. Subramanian, A.; Drummond, L.M.A.; Bentes, C.; Ochi, L.S.; Farias, R. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Comput. Oper. Res.* **2010**, *37*, 1899–1911. [CrossRef]
12. Montané, F.A.T.; Galvao, R.D. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Comput. Oper. Res.* **2006**, *33*, 595–619. [CrossRef]
13. Zhou, H.; Qin, H.; Zhang, Z.; Li, J. Two-echelon vehicle routing problem with time windows and simultaneous pickup and delivery. *Soft Comput.* **2022**, *26*, 3345–3360. [CrossRef]
14. Goeke, D. Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. *Eur. J. Oper. Res.* **2019**, *278*, 821–836. [CrossRef]
15. Soysal, M.; Çimen, M.; Belbağ, S. Pickup and delivery with electric vehicles under stochastic battery depletion. *Comput. Ind. Eng.* **2020**, *146*, 106512. [CrossRef]
16. Yilmaz, Y.; Kalayci, C.B. Variable Neighborhood Search Algorithms to Solve the Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Mathematics* **2022**, *10*, 3108. [CrossRef]
17. Löffler, M.; Desaulniers, G.; Irnich, S.; Schneider, M. Routing electric vehicles with a single recharge per route. *Networks* **2020**, *76*, 187–205. [CrossRef]
18. Schiffer, M.; Walther, G. The electric location routing problem with time windows and partial recharging. *Eur. J. Oper. Res.* **2017**, *260*, 995–1013. [CrossRef]
19. Bruglieri, M.; Pezzella, F.; Pisacane, O.; Suraci, S. A matheuristic for the electric vehicle routing problem with time windows. *arXiv* **2015**, arXiv:1506.00211. [CrossRef]
20. Desaulniers, G.; Errico, F.; Irnich, S.; Schneider, M. Exact algorithms for electric vehicle-routing problems with time windows. *Oper. Res.* **2016**, *64*, 1388–1405. [CrossRef]
21. Felipe, Á.; Ortuño, M.T.; Righini, G.; Tirado, G. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transp. Res. Part E Logist. Transp. Rev.* **2014**, *71*, 111–128. [CrossRef]
22. Kancharla, S.R.; Ramadurai, G. Electric vehicle routing problem with non-linear charging and load-dependent discharging. *Expert Syst. Appl.* **2020**, *160*, 113714. [CrossRef]
23. Lee, C. An exact algorithm for the electric-vehicle routing problem with nonlinear charging time. *J. Oper. Res. Soc.* **2021**, *72*, 1461–1485. [CrossRef]
24. Zuo, X.; Xiao, Y.; You, M.; Kaku, I.; Xu, Y. A new formulation of the electric vehicle routing problem with time windows considering concave nonlinear charging function. *J. Clean. Prod.* **2019**, *236*, 117687. [CrossRef]
25. Froger, A.; Mendoza, J.E.; Laporte, G.; Jabali, O. New Formulations for the Electric Vehicle Routing Problem with Nonlinear Charging Functions. 2017. hal-01559507. Available online: https://hal.archives-ouvertes.fr/hal-01559507 (accessed on 6 November 2020).
26. Shao, S.; Guan, W.; Bi, J. Electric vehicle-routing problem with charging demands and energy consumption. *IET Intell. Transp. Syst.* **2018**, *12*, 202–212. [CrossRef]
27. Kancharla, S.R.; Ramadurai, G. An adaptive large neighborhood search approach for electric vehicle routing with load-dependent energy consumption. *Transp. Dev. Econ.* **2018**, *4*, 10. [CrossRef]
28. Zhang, S.; Gajpal, Y.; Appadoo, S.S.; Abdulkader, M.M.S. Electric vehicle routing problem with recharging stations for minimizing energy consumption. *Int. J. Prod. Econ.* **2018**, *203*, 404–413. [CrossRef]

29. Basso, R.; Kulcsár, B.; Egardt, B.; Lindroth, P.; Sanchez-Diaz, I. Energy consumption estimation integrated into the Electric Vehicle Routing Problem. *Transp. Res. Part D Transp. Environ.* **2019**, *69*, 141–167. [CrossRef]
30. Schneider, M.; Stenger, A.; Goeke, D. The electric vehicle-routing problem with time windows and recharging stations. *Transp. Sci.* **2014**, *48*, 500–520. [CrossRef]
31. Keskin, M.; Çatay, B. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transp. Res. Part C Emerg. Technol.* **2016**, *65*, 111–127. [CrossRef]
32. Verma, A. Electric vehicle routing problem with time windows, recharging stations and battery swapping stations. *EURO J. Transp. Logist.* **2018**, *7*, 415–451. [CrossRef]
33. Koç, Ç.; Jabali, O.; Mendoza, J.E.; Laporte, G. The electric vehicle routing problem with shared charging stations. *Int. Trans. Oper. Res.* **2019**, *26*, 1211–1243. [CrossRef]
34. Zang, Y.; Wang, M.; Qi, M. A column generation tailored to electric vehicle routing problem with nonlinear battery depreciation. *Comput. Oper. Res.* **2022**, *137*, 105527. [CrossRef]
35. Kyriakakis, N.A.; Sevastopoulos, I.; Marinaki, M.; Marinakis, Y. A hybrid Tabu search—Variable neighborhood descent algorithm for the cumulative capacitated vehicle routing problem with time windows in humanitarian applications. *Comput. Ind. Eng.* **2022**, *164*, 107868. [CrossRef]
36. Barth, M.; Boriboonsomsin, K. Energy and emissions impacts of a freeway-based dynamic eco-driving system. *Transp. Res. Part D Transp. Environ.* **2009**, *14*, 400–410. [CrossRef]
37. Koç, Ç.; Karaoglan, I. The green vehicle routing problem: A heuristic based exact solution approach. *Appl. Soft Comput.* **2016**, *39*, 154–164. [CrossRef]
38. Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **2006**, *40*, 455–472. [CrossRef]
39. Xu, W. Electric Vehicle Routing with Soft Time Windows. In Proceedings of the 22nd COTA International Conference of Transportation, Changsha, China, 8–11 July 2022; pp. 2516–2525. [CrossRef]