energies

MDPI

Article

Energy Saving-Oriented Multi-Depot Vehicle Routing Problem with Time Windows in Disaster Relief

Peng Xu, Qixing Liu and Yuhu Wu *

Key Laboratory of Intelligent Control and Optimization for Industrial Equipment of Ministry of Education, School of Control Science and Engineering, Dalian University of Technology, Dalian 116081, China
* Correspondence: wuyuhu@dlut.edu.cn

Abstract: This paper studies the distribution of emergency relief for electric vehicles (EVs), which considers energy saving, multi-depot, and vehicle routing problems with time windows, and the named energy saving-oriented multi-depot vehicle routing problem with time windows (ESMDVRPTW). Our aim is to find routes for EVs such that all the shelter demands are fulfilled during their time windows and the total cost traveled by the fleet is minimized. To this end, we formulate the ESMDVRPTW as a mixed-integer linear programming model. Since the post-disaster transportation network contains a large number of vertices and arcs composed of vertices, we propose a two-stage approach to solve the ESMDVRPTW. The first stage is to obtain the minimal travel cost between any two vertices in real-time on a post-disaster transportation network using the proposed Floyd algorithm combined with the neighboring list (Floyd-NL algorithm). In the second stage, we develop the genetic algorithm (GA) incorporating large neighborhood search (GA-LNS), which determines the delivery scheme of shelters. Simulation results of the MDVRPTW benchmark illustrate that the performance of the GA-LNS is better than GA , simulated annealing (SA) and tabu search (TS). Finally, case studies are constructed on two real cases acquired from the OpenStreetMap (OSM) generated by the Quantum Geographic Information System (QGIS) in Ichihara city, Japan, and the test results of case studies show the effectiveness of the proposed two-stage approach.

Keywords: electric vehicles; energy saving-oriented; multi-depot; Floyd-NL algorithm; GA-LNS algorithm

Citation: Xu, P.; Liu, Q.; Wu, Y. Energy Saving-Oriented Multi-Depot Vehicle Routing Problem with Time Windows in Disaster Relief. *Energies* **2023**, *16*, 1992. https://doi.org/10.3390/en16041992

Academic Editor: Elyas Rakhshani

Received: 16 January 2023
Revised: 8 February 2023
Accepted: 15 February 2023
Published: 17 February 2023

1. Introduction

With the continuous development of human society and economy, the ecological environment has been seriously damaged, causing natural disasters to occur frequently. When a disaster occurs, emergency rescue departments occupy a crucial position, which need to carry out a rapid and reasonable rescue. One of the most critical tasks is to dispatch EVs to transport needed materials to the shelter, such as medical bags, tents, and food. Since the urgency of shelter is different, it is necessary to set the time window in shelter to provide rescue reasonably. Then, we consider the multiple depots in post-disaster transportation networks to make emergency rescue more flexible. Furthermore, since EVs have their energy consumption when driving on the arc, the saving energy of EVs means that EVs can provide materials for more shelters under the same energy consumption. In summary, the emergency logistics within our work is formulated as an energy saving-oriented multi-depot vehicle routing problem with time windows (ESMDVRPTW).

In the field of traffic planning, the vehicle routing problem (VRP) is a popular problem. The VRP was first proposed by the Ref. [1] in 1959, and its variants have been widely studied in the literature. The capacitated VRP (CVRP) is also a very important consideration in practical application. In the Ref. [2], the authors proposed a vehicle load constraint, and in the design of the objective function, the study not only considered minimizing total travel cost, but also considered minimizing the travel distance of each arc. Minimizing the

travel distance of each arc can ensure that there is not much difference in the workload of each driver, ensuring the fairness of workload and the balance of income. In general, the capacity constraint considered by the CVRP is that the weight of the materials carried by the vehicle cannot exceed the weight delivered to the customer. Tao et al. [3] addressed the CVRP with three-dimensional loading. The author considered the shape constraint of three-dimensional loaded materials, which was a complex problem combining three-dimensional loaded materials with CVRP. In the Ref. [4], the authors presented a VRP of simultaneous pickup and delivery, which considered how customers had both pickup and delivery requests. The authors presented an adaptive large neighborhood search algorithm (ALNS) to minimize the working time of EVs and the numbers of EVs.

Time window constraint is also a factor that is often considered in VRP. Sánchez et al. [5] presented a mixed-integer linear programming model to solve the electric location routing problem with time windows (E-LRPTW), which obtained the location of EV recharging stations under different distribution services. In this model, the author considered the state of charge, battery capacities and the time windows of customers and used the clustering strategy based on the k-means algorithm to get the potential location for recharging stations. In the Ref. [6], the authors proposed an electric vehicle routing problem with time windows (EVRPTW) considering different EV recharging strategies. The authors used an ALNS to solve the proposed problem and verified that the cost of a partial recharging strategy is less than the cost of a full recharging strategy.

The multi-depot vehicle routing problem with time windows (MDVRPTW) is also an extension of a VRP application. In the Ref. [7], the authors presented a practical logistics problem applied to the delivery and installation of electronic products. The author divided the delivery vehicle and the installation vehicle into two parts. However, in order to improve the service quality, the difference between the delivery vehicle arriving and the installation vehicle arriving at the same customer should not be too large, which is a time constraint problem. There is more than one depot for electronic products, so the author considered this practical logistics problem as MDVRPTW. Adelzadeh et al. [8] studied the practical application of distribution by using different kinds of vehicles. Different kinds of vehicles represented different capacity, speed and cost of consumption, which made the study more complicated. The objective functions considered by the authors were minimizing the driving distance and minimizing the waiting time. Reducing the driving distance can reduce the operating cost of the enterprise, reduce the waiting time can improve the quality of customer service, and also increase the enterprise's reputation and income.

Additionally, there is more and more research on applying VRP and some VRP variants to emergency logistics. In the Ref. [9], they took into account that in emergency logistics, vehicle speed will be changed with different kinds of disasters and proposed a bio-inspired algorithm, which selected the optimal paths from alternative paths considering travel time. In the Ref. [10], a greedy-search-based multi-objective genetic algorithm was proposed to solve the emergency rescue logistics. The objective function was to minimize the unsatisfied demand for resources, total travel time, and total transportation cost to generate distribution tasks for each distribution center. In the Ref. [11], the authors considered a new optimization model, which is based on the actual problem of a large number of disaster-affected sites, relatively concentrated distribution, and a small number of emergency supplies. In the Ref. [12], they presented a double-layer ant colony optimization algorithm. The algorithm determines the feasible solution by minimizing the cost of emergency logistics transportation, which improves the distribution efficiency. Therefore, the vehicle routing and scheduling problems of emergency logistics are worth studying.

In terms of vehicle energy savings, Li et al. [13] studied an advanced engine thermal management system that reduced energy consumption by controlling problems with high precision. In the system, there are many time-varying disturbances or time delays, which brings challenges to the temperature tracking of the actuator. To solve this problem, an adaptive optimal control strategy was designed by combining state, disturbance and energy

consumption optimization models. In the experiment, the steady-state error of temperature can be achieved within 0.3 °C by this strategy, and high-precision temperature tracking can also be achieved under the constantly changing complex working conditions. In the Ref. [14], the authors considered energy management on a hydrogen powered vehicle. Compared with the traditional rule based on expert experience, the author proposed a multiple linear regression model to learn rule parameters, which was called the fuzzy rule-based strategy. In order to solve the path, the author firstly used the dynamic programming (DP) algorithm to solve the energy calculation of the off-line path, and then used fuzzy rules to select the path. Finally, the author compared the rules based on expert experience and fuzzy rules based on a genetic algorithm. Hydrogen energy consumption decreased by 1.5% and 10%, respectively. Hou et al. [15] proposed a multi-layered model predictive control framework, which combined the life management and real-time speed of fuel cells. In the upper layer, the SOC state of the battery was solved by convex optimization. In the lower layer, a model predictive control based on the equivalent consumption minimum strategy was designed according to the SOC state of the upper layer. Compared with the rule-based strategy, the multi-layered model predictive control can save more than 10% of energy. Zhu et al. [16] proposed a Bayesian-Gated Recurrent Unit model (B-GRU) that combined the Bayesian Theory and GRU to quantify the uncertainty of the degradation prediction results for PEMFCs.

In addition, we have some relevant research basis for VRP, where the VRP is extended to the application of hybrid EV. In the Ref. [17], the authors studied the finite time domain optimal control problem of discrete power systems in a continuous domain, which was applied to the optimal control problem of hybrid EV power systems. Combined with EV and VRP, in the Ref. [18], minimized battery energy consumption of EV was taken as the optimization objective, and the influence of slope on battery energy consumption was considered. In terms of solving algorithms, this paper presented a method combining 2-opt and GA to solve this problem. However, the slope was the value set by our simulation, which was not combined with the actual geographical data. In the Ref. [19], the actual slope information was extracted and applied to calculate the EV energy consumption. In another paper, we considered the energy consumption of EVs combined with the energy consumption of shelters. Each shelter had a different initial charge and rate of consumption. EVs needed to be able to deliver power to shelters in a timely manner. This paper is currently under review. The application of a vehicle routing problem with timed-paths (VRPTP) in emergency logistics was studied in the Ref. [20].

The main contributions of this paper are as follows: (1) We consider the ESMDVRPTW on a post-disaster transportation network and design the cost function aimed at reducing the energy consumption of EVs, which is rarely considered; (2) to solve the ESMDVRPTW, we propose a two-stage approach. The first stage is to obtain the minimal travel cost between any two vertices in real time by Floyd-NL. Compared with the Floyd Algorithm, the complexity of Floyd-NL is less in solving a post-disaster transportation network; (3) the second stage is to solve the delivery scheme by GA-LNS. Differently from GA, we use a new destory and repair operator in GA-LNS, which can expand the neighborhood of the current delivery scheme to get a better delivery scheme.

The remainder of the paper is organized as follows. The problem description is present in Section 2. Section 3 introduces the first stage of our approach, which uses the Floyd-NL. In Section 4, we introduce the second stage of our approach, which determines the delivery scheme of shelters. Simulation results and examples are shown in Section 5. Conclusions are summarized in Section 6.

2. Problem Formulation and Definition

The ESMDVRPTW can be defined on an incomplete and undirected graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ represents the vertex set. A vertex set consists of three subsets: V_1, V_2 and V_3, where $V_1 = \{v_1, \ldots, v_\alpha\}$ is a set of the α depots, V_2 is the set of shelters that needs to be serviced, denoted by $\{v_{\alpha+1}, \ldots, v_{\alpha+\beta}\}$, and $V_3 = \{v_{\alpha+\beta+1}, \ldots, v_n\}$ represents

the set of vertices along the path. $E \subset V \times V$ denotes the set of all available arcs on the post-disaster transportation network. Each arc $(v_i, v_j) \in E, i \neq j$ associates with an energy consumption c_{ij} of the EV and a travel time t_{ij} of the EV.

K denotes a fleet of homogeneous EVs, which has the identical battery capacity Q with a constant speed v. The materials capacity of each EV is denoted by q. As for shelters, each shelter i requires a material demand d_i. Furthermore, each shelter $i \in V_2$ must be serviced within a time window $[l_i, r_i]$. l_i is the earliest service time of shelter i. When the k-th EV arrives at shelter i before l_i, the EV should wait until l_i. r_i represents the latest service time of shelter i. Service time at shelter i is denoted by s_i. For the sake of convenience, the time windows of all depots are $[l_0, r_0]$. All the EVs cannot leave depots earlier than l_0 and must return no later than r_0.

We define the decision variable x_{ijk} as follows:

$$x_{ijk} = \begin{cases} 1 & \text{if the } k\text{-th EV travels from } i \text{ to } j \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Another decision variable y_{ik} denotes the arrival time of the k-th EV to service shelter i, and we assume $y_{0k} = l_0$. Let x, y be matrices with proper dimensions holding all variables.

The mathematical model of the ESMDVRPTW can be shown as

$$\min \quad F(x,y) = c(x) + T(x,y) \tag{2}$$

$$s.t. \quad \sum_{k\in K}\sum_{i\in V} x_{ijk} = 1, \quad \forall j \in V_2 \tag{3}$$

$$\sum_{i\in V} x_{ihk} - \sum_{j\in V} x_{hjk} = 0, \quad \forall h \in V_2 \cup V_3, \forall k \in K \tag{4}$$

$$\sum_{h\in V_1}\sum_{i\in V_2\cup V_3} x_{ihk} = \sum_{h\in V_1}\sum_{j\in V_2\cup V_3} x_{hjk} \leq 1, \quad \forall k \in K \tag{5}$$

$$\sum_{i\in V}\sum_{j\in V_2} x_{ijk} \cdot d_j \leq q, \quad \forall k \in K \tag{6}$$

$$\sum_{i\in V}\sum_{j\in V} x_{ijk} \cdot c_{ij} \leq Q, \quad \forall k \in K \tag{7}$$

$$y_{ik} + s_i + t_{ij} - M_{ij}(1 - x_{ijk}) \leq y_{jk}, \forall i \in V, \forall j \in V_2 \cup V_3, \forall k \in K \tag{8}$$

$$y_{ik} + s_i + t_{i0} - M_{i0}(1 - x_{i0k}) \leq r_0, \forall i \in V_2 \cup V_3, \forall k \in K \tag{9}$$

$$y_{0k} = l_0. \tag{10}$$

In the equation, $F(x,y)$ represents the cost function, where $x = \{x_{ijk} = 0,1, i, j \in V, \forall k \in K\}$, $y = \{y_{ik}, i \in V, \forall k \in K\}$. The first term $c(x)$ of $F(x,y)$ is the total energy cost of EVs after all the shelter demands are fulfilled and all EVs return to depots, where we can define $c(x)$, as described in Equation (11). c_d can be thought as the cost for one unit of energy.

$$c(x) = c_d \sum_{k\in K}\sum_{i,j\in V} c_{ij} x_{ijk} \tag{11}$$

Furthermore, the last term $T(x,y)$ of $F(x,y)$ is the total cost of violating time windows. $T(x,y)$ is in Equation (12). c_t is the cost for one unit of violating time windows.

$$T(x,y) = c_t\Big(\sum_{k\in K}\sum_{i\in V} \max\{y_{ik} - r_i, 0\} + \sum_{k\in K}\sum_{i\in V} \max\{l_i - y_{ik}, 0\}\Big). \tag{12}$$

Equation (3) requires that each shelter is served by an EV once. Equation (4) states flow conservation constraints. Equation (5) shows that each EV starts from a depot and stops at a depot. Equation (6) requires that the sum of material demand of each path cannot exceed the EV capacity. Equation (7) requires that the energy consumption of each path cannot exceed the battery capacity of EV. Equation (8) ensures that EVs access i and j in

order, and the constant M_{ij} is defined as large enough to eliminate subtours. Equation (9) requires that EVs return to depots no later than r_0. Equation (10) states that the EVs start from the depot at l_0.

3. The First Stage of the Approach: Floyd-NL Algorithm

Based on the mathematical model of ESMDVRPTW, the first stage of our approach is the Floyd-NL algorithm, which obtains the minimal travel cost between any two vertices on a post-disaster transportation network. When the disaster occurs, the transportation network changes, and some arcs are even impassable. We need the Floyd-NL algorithm to calculate the minimal travel cost between any two vertices in real-time again.

The Floyd algorithm [21] is an algorithm to find the shortest path in a weighted graph with positive or negative edge weights (but no negative period), which is mainly applied to a fully connected graph. However, the graph of a disaster situation in actual cases is usually incomplete. The Floyd-NL algorithm is proposed to reduce the number of traverse arcs to reduce the algorithm running time. We will explain the Floyd-NL algorithm in the Algorithm 1.

Algorithm 1 Floyd-NL algorithm

Input: *start_point*, *end_point*, the travel cost c_{ij}
Output: *optimal_path*,*dist*

Define routing matrix p, neighboring list *neighboring_list*
for i in $[1:n]$ **do**
  for j in $[1:n]$ **do**
    $p(i,j) = j$;
    $dist(i,j) = c_{ij}$;
    if c_{ij} is not equal to infinity **then**
      $neighboring\_list\{i\} = [neighboring\_list\{i\}, j]$;
    end if
  end for
end for
for h in $[1:n]$ **do**
  for i in $[1:n]$ **do**
    for j in $neighboring\_list\{h\}$ **do**
      if $dist(i,j) > dist(i,h) + dist(h,j)$ **then**
        $dist(i,j) = dist(i,h) + dist(h,j)$;
        $p(i,j) = p(i,h)$;
      end if
    end for
  end for
end for
Find the optimal route *optimal_path* between the *start_point* and *end_point* through the routing matrix p;

In the Algorithm 1, we assume that if the arcs exist between any two vertices, then we set the actual cost between them. However, the cost between non-neighboring vertices is set to infinity. By dealing with the non-neighboring vertices in the initial neighboring matrix of vertices, we can obtain the vertex neighboring table without infinity. First, we use QGIS software to extract geographic information data. QGIS is a desktop GIS software and provides data display, editing and analysis functions. Because the number of neighboring vertices of a vertex is less than the total number of all vertices through the observation of the datasets extracted by QGIS, through the neighboring list, we no longer need to traverse non-neighboring vertices in the operation of the algorithm. When solving our problem, this improvement can efficiently reduce the computational complexity compared with the Floyd algorithm.

Two parts can consider the computational complexity of the Floyd-NL algorithm. The first part is from line 1 to line 10 in the pseudocode, which contains two n loops. Thus, the computational complexity of this part is $\mathcal{O}(n^2)$. The other part is from line 11 to line 20 in the pseudocode, which contains three loops. The first and second are n loops. The length of each element in *neighboring_list* represents the total number of neighboring vertices for each shelter. As we mentioned in the last paragraph, we can see that the number of neighboring vertices of a vertex is less than the total number of all vertices. Therefore, we assume that the length of elements in *neighboring_list* is z_i, where $z_i \leq \lambda \ll n$, λ is the maximum length of z_i. Thus, the third loop is at most λ, and the computational complexity of this part is $\mathcal{O}(\lambda n^2)$. Based on the above discussion, the computational complexity of the Floyd-NL algorithm is $\mathcal{O}(\lambda n^2)$. As we know, the computational complexity of the Floyd algorithm is $\mathcal{O}(n^3)$. As a result, the improvement of the Floyd algorithm in our problem can reduce the computational complexity effectively.

In order to verify the effectiveness of the Floyd-NL algorithm, we use QGIS to extract some real-world traffic data, which contains 445 nodes in total and 355 edges that can be connected between vertices. We select six groups of different start vertices and end vertices and apply the Floyd-NL algorithm and the Floyd algorithm to obtain the minimal travel cost between them. The starting vertices and ending vertices of the six groups are $[6, 122]$, $[7, 147]$, $[12, 110]$, $[16, 113]$, $[19, 162]$, $[21, 102]$, respectively. In Table 1, the first row is the running time of the Floyd-NL algorithm (unit: second), and the second row is the running time of the Floyd algorithm (unit: second). The results show that the running time of the Folyd-NL algorithm is much lower than that of the Floyd algorithm, which illustrates the effectiveness of the Floyd-NL algorithm. The third row represents the minimal energy cost calculated by the Floyd algorithm and Floyd-NL algorithm (unit: watt). In terms of calculating the minimal energy cost, both the Floyd algorithm and Floyd-NL algorithm can calculate the optimal value, but the Floyd-NL algorithm can effectively reduce the running time.

Table 1. A comparison of algorithm running time and cost with Floyd-NL, the Floyd algorithm.

Algorithm	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
Floyd-NL	0.063	0.047	0.0312	0.047	0.045	0.032
Floyd	9.98	8.99	9.017	9.052	8.99	8.97
cost	1840	324	326	382	131	105

4. The Second Stage of the Approach: GA-LNS Algorithm

The GA algorithm is a highly parallel, random, and adaptive optimization algorithm. In order to obtain a better delivery scheme, we add the LNS algorithm [22] to the genetic algorithm. The LNS algorithm can search for a better solution in the neighborhood of a solution. The flowchart of the GA-LNS algorithm is given in Figure 1. We will introduce the GA-LNS algorithm in detail to find the delivery scheme of each shelter.

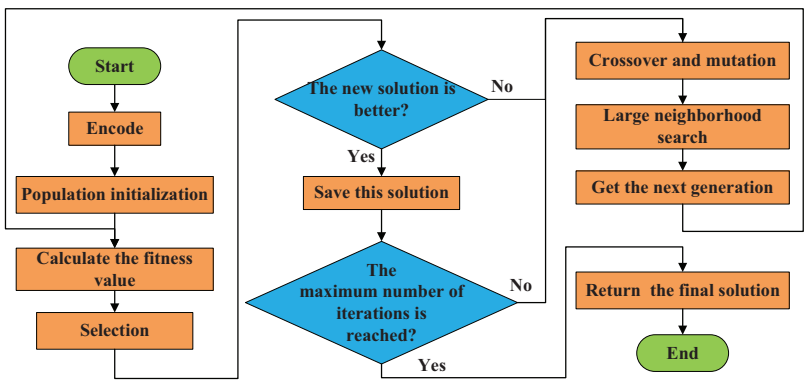


Figure 1. The flowchart of the GA-LNS algorithm.

4.1. Encode

Encoding is abstraction of the problem to be solved into a series of specific symbols, which are arranged in a certain order through a certain mechanism. We will use a simple example to illustrate how our problem was encoded.

We assumed the number of shelters β was five, the number of depots α was two, and the number of vehicles $|K|$ was two. Figure 2 shows a feasible chromosome as 23,645, where the depot is denoted by 0, 1. The chromosome sequence is segmented according to the mathematical model constraints in Section 2. The chromosome may be divided into two parts. Part one is 23, and part two is 645. We calculated the distance between each depot and the first node A of each part, and added the nearest depot before A; at the same time, we calculated the distance between the last node B of each part and the depot, and added the nearest depot after node B. As shown in Figure 2, we generated two routes, $0 \rightarrow 2 \rightarrow 3 \rightarrow 1$,$1 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow 0$.

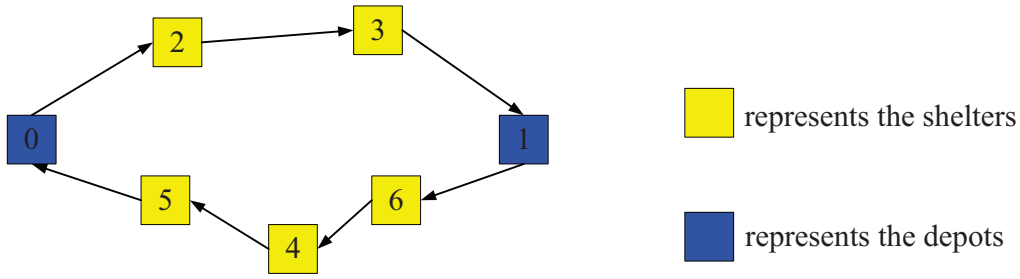


Figure 2. A reasonable chromosome encoded in MDVRPTW.

4.2. Population Initialization

The size of the population depends on the properties of the problem. In general, it contains several hundred or thousands of possible solutions. Within our work, each solution consists of some shelters, and the EV starts from a depot and ends at a depot after service. Therefore, the form of initial solutions equals the form of encoding, and the initial solutions are chosen from the total population.

4.3. Fitness Function

The fitness of chromosomes refers to the measure of the dominance of chromosomes in population survival, which is used to distinguish the quality of chromosomes. Fitness is calculated using the fitness function. The fitness function is also called the evaluation function, which mainly judges the fitness of a chromosome through chromosome characteristics.

Since the above encoding cannot consider that each delivery scheme meets the EV time window constraints, we need to use a function with penalty terms to solve the problem. In addition, chromosomes with large fitness values need to be selected when selecting operations. Therefore, we set the fitness function as $1/F(x, y)$.

4.4. Selection

Roulette wheel selection is used in the proposed method for probabilistic selection based on chromosome performance. The roulette wheel selection method scales the fitness values of individuals in the population so that the sum of the rescaled fitness values equals 1. In roulette, the probability of selecting a chromosome is directly proportional to their fitness. The process of selecting chromosomes is shown in Figure 3, where the fitness of chromosome 1 is greater than that of other chromosomes, so chromosome 1 will be selected as the optimal solution at present.

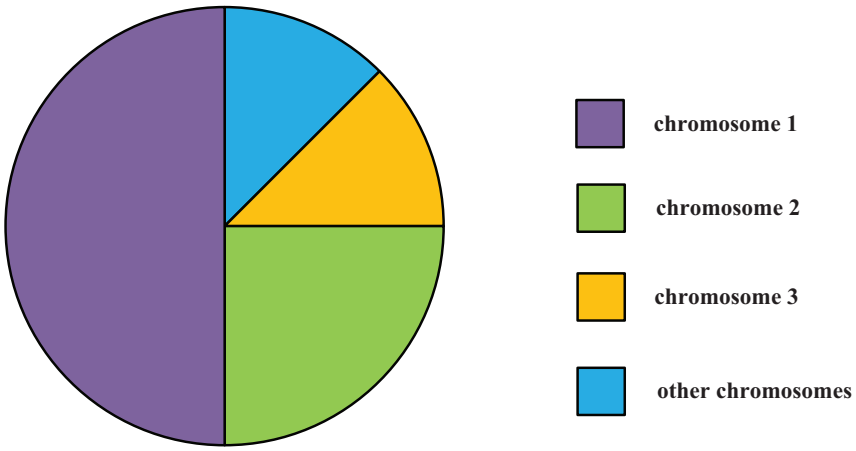


Figure 3. The process of select chromosomes in the GA-LNS algorithm.

4.5. Crossover

The process of crossover generating the new offspring is shown in Figure 4. We used the partial-mapped crossover approach to crossover the chromosomes. The crossover can improve the solution diversity. We give two parent chromosomes, randomly select two crossing positions a and b, such as $a = 2$, $b = 5$, and then the cross-fragment of parent two is exchanged with the cross-fragment of parent 1. It is noticed that there exists the repeated gene after crossover. We give the following corresponding relationship in the lower part of Figure 4. If the repeated gene occurs, change the first repeated gene observing the corresponding relationship.

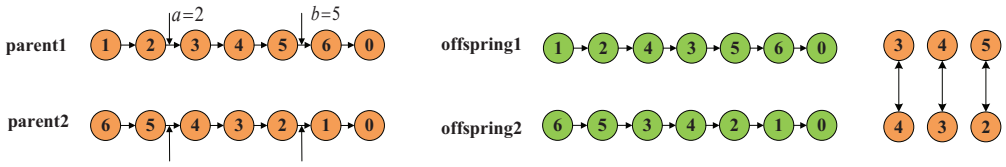


Figure 4. The crossover process in the GA-LNS algorithm.

4.6. Mutation

In the mutation operation, we use insertion mutation to mutate the chromosomes. The mutation process is shown in Figure 5. The parent chromosome randomly selects one gene, and inserts this gene to a random position, then the offspring chromosome is generated.

Figure 5. The mutation process in GA-LNS algorithm.

4.7. Large Neighborhood Search

If only the GA is used, there is no good search for the neighborhood of the solution, only random, crossover and mutation. These are not enough to solve well, so we added a LNS algorithm based on the GA. The advantages of a GA based on population and LNS for neighborhood expansion are combined. LNS is improved step by step by using Destroy and Repair alternately. With Algorithm 2, we can see the flow of the LNS algorithm.

Algorithm 2 LNS algorithm

Input: an initial solution S_0
1: $Sbest = S_0$;
2: **repeat**
3: $S_r = rep(dest(S_0))$;
4: **if** $fitness(S_r) > fintess(S_0)$ **then**
5: $S_0 = S_r$;
6: **end if**;
7: **if** $fitness(S_r) > fintess(Sbest)$ **then**;
8: $Sbest = S_r$;
9: **end if**;
10: **until** (stop criterion is met)
11: **return** $Sbest$

In Algorithm 2, we first initialize the initial solution as S_0, which is the output after the crossover and mutation of the genetic algorithm. Additionally, we assign the optimal solution $Sbest$ to our initial solution S_0. Continuing, as in lines 2 through 10 of Algorithm 2, the algorithm enters a loop until the termination condition is satisfied. When the algorithm executes, we set the loop to execute only once, although we could change it to more times or set a more specific stop condition, such as the algorithm stops below a certain value. Because we combine the genetic algorithm with the large neighborhood search algorithm, the GA-LNS algorithm will have iterative cycles, so it is not necessary to set more cycles for the LNS algorithm separately.

In the 3 line of Algorithm 2, we generate a new solution S_r using the $rep(dest(\cdot))$ operator. The $rep(dest(\cdot))$ operator mainly has destroy and repair—two parts.

(1) destory operator

In the destory operator, S_{remove} represents the shelter to be removed, R represents the set of shelters to be removed, N represents the number of shelters to be removed, and S_{remain} represents the remaining partial solutions after N shelters are removed from S_0. The process of the destory operator is as follows:

First, randomly remove a shelter from S_0 into R as the first element of the set R. Then, one shelter M is randomly selected from the set R at a time, and all shelters in S_{remain} are sorted as S_{list} in order of their greatest to least relevance to M. Select the shelter S_{remove} from S_{remain} according to Equation (15), remove the S_{remove} from S_0, and add it to R. Repeat this process $N-1$ times. The correlation equation is shown in Equation (13).

$$\Phi(i,j) = 1/\left(c_{ij} + w_{ij}\right). \tag{13}$$

$\Phi(i,j)$ represents the correlation between i and j, c_{ij} represents the energy consumption and w_{ij} indicates whether i and j are served by the same EV. If i and j are served by the same EV, w_{ij} is 0, otherwise it is 1. In order to avoid over-reliance on the correlation function, a random function is applied to the selection of S_{remove}, as described in Equation (14).

$$idx = \left[(Rand)^D \times |S_{remain}|\right] \tag{14}$$

$Rand$ is a random number between 0 and 1, and D is a constant, which we set to 8 in our experiment. $|S_{remain}|$ represents the number of remaining shelters, rounded to give the index value idx.

Finally, S_{remove} can be calculated according to Equation (15).

$$S_{remove} = S_{list}[idx] \tag{15}$$

(2) repair operator

For the repair operator, we first separately calculated the fitness of all the insertable points of all shelters in set R into S_{remain}. For each shelter, we found the position pos with the least fitness after insertion among all the insertable points. Then, we compared the

pos of each shelter and found the largest one to insert into S_{remain} first. We repeated this operation until we completely inserted the elements of set R into S_{remain} and obtained S_r.

We then calculate the fitness values for S_0 and S_r (see Algorithm 2, line 4–6). If the fitness of S_r is greater than that of S_0 , change S_0 to S_r and calculate the fitness values for S_r and S_{best} (see Algorithm 2, line 7–9). If the fitness of S_r is greater than that of S_{best} , change S_{best} to S_r. After the termination condition, the output S_{best} is given to the genetic algorithm to continue the iterative solution.

Based on the above features, we will give the simulation results in the next section.

5. Simulation Results

5.1. Simulation Results of the Datasets Modified Using Cordeau Problem

To prove the practicability of the GA-LNS to obtain the delivery scheme on a post-disaster transportation network, we considered modifying part of the Cordeau problem [23] to get the MDVRPTW benchmark. Different from the Cordeau problem, we considered the energy consumption of EVs between any two vertices. The details of the benchmark are presented in Table 2. In Table 2, column 1 indicates the benchmark. Columns 2, 3 respectively represent the number of depots and shelters with different benchmarks.

Table 2. Test data parameters.

Benchmark	Number of Depots	Number of Shelters
C01	4	48
C02	4	96
C03	6	144
C04	4	96
C05	4	144
C06	6	144

For the GA-LNS and GA, we set the population to 100, where the probability of crossover is 0.95, and the probability of mutation is 0.05. Each time, the result of the genetic algorithm is followed by the large neighborhood search algorithm; for SA, we set the initial temperature to 6000, the termination temperature to 0.001, and temperature drop step to 0.9; for TS, we set the length of the tabu list to 30. We determined through experimental tests that we will get a better result by running 100 iterations. In the cost function, c_d was set to 1.0, and c_t was set to 100.0.

Then, the comparison of travel cost with the GA-LNS, GA [24], SA [25] and TS [26] algorithm was presented in Table 3. Column 1 shows the benchmark, the type of which is given in Table 2. The column 2 shows the average of the 10 simulations performed by the GA-LNS algorithm on the benchmark. Columns 3, 4, 5 show the simulation results with a different algorithm. The data calculated by the algorithm are the minimal travel cost where EVs can travel from the depot and return to the depot after serving all shelters. The results indicate that the GA-LNS algorithm has better performance. In the C04, the gap between the results of the GA-LNS algorithm and other algorithms is minimal. The results of the GA-LNS algorithm are 41.6%, 39.3% and 36.1% of that of GA, SA and TS, respectively. In the C02, the gap between the results of the GA-LNS algorithm and other algorithms is the maximum. The results of the GA-LNS algorithm are 25.8%, 24.4% and 23% of those of GA, SA and TS, respectively.

In Table 4, column 1 indicates the benchmark. Column 2 shows the GA-LNS algorithm running time of 100 iterations on the benchmark. Columns 3, 4 and 5 show the running time of 100 iterations with GA, SA and TS (unit: second). Although the running time of GA-LNS is longer than that of other algorithms, the solutions of each benchmark are completed within about one hour. The solution results of GA-LNS are better than those of other algorithms. Moreover, as shown in Figure 6, we used four algorithms to draw an iterative curve on the C01.

Table 3. A comparison of travel cost with the GA-LNS, GA, SA and TS algorithm.

Benchmark	GA-LNS	GA	SA	TS
C01	1841.63	4107.48	4190.23	4536.21
C02	1827.67	7095.72	7500.97	7937.73
C03	2295.44	6178.40	7144.12	6963.35
C04	1594.87	3832.74	4059.53	4414.24
C05	2235.78	7043.94	7675.14	7675.01
C06	2215.25	6309.35	6806.60	7175.82

Table 4. A comparison of algorithm running time with the GA-LNS, GA, SA and TS algorithm.

Benchmark	GA-LNS	GA	SA	TS
C01	544.94	84.59	10.24	9.78
C02	2211.45	128.30	26.62	24.37
C03	2400.96	131.55	23.77	23.77
C04	625.64	79.49	13.89	10.29
C05	3588.77	127.30	24.80	25.40
C06	3760.03	115.63	29.37	30.62

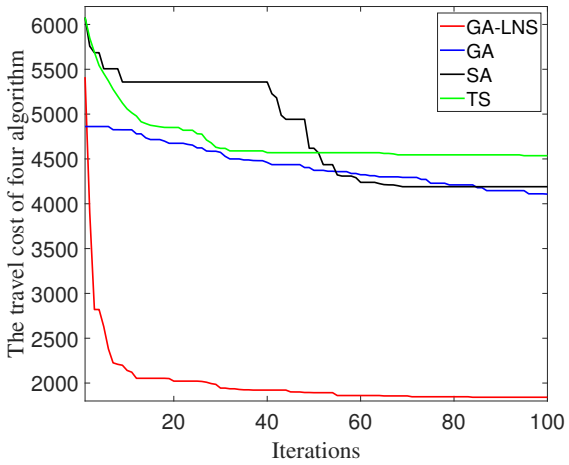


Figure 6. The iterative curve of four algorithms on C01.

5.2. Simulation Results for a Practical Example

In this section, we consider two different cases acquired from the OpenStreetMap (OSM) in Ichihara city, Japan. The post-disaster transportation network in Figure 7 has about 11,000 vertices and more than 15,000 arcs composed of vertices. In Figure 7a, there are 4 depots and 48 shelters, which is called case 1 for short; in Figure 7b, there are 4 depots and 60 shelters, which is called case 2 for short. The locations of depots and shelters are shown in Figure 7. The blue squares represent the depots. The red circles represent the shelters.

The two cases have some of the same settings. The service time of each shelter was set to $s_i = 90$ min, and their demand was set to $d_i = 50$ units. These data can be easily modified to adapt to the changes of different shelters. The capacity of each EV was $q = 200$ units, and the velocity of each EV was v = 30 km/h. In case 1, we considered that up to 8 EVs can be used; in case 2, we considered using up to 12 EVs. We aimed to find the optimal delivery scheme while minimizing the total travel cost. We considered the different priorities of shelters by changing the time windows in Table 5. Numbers 0–51 were the information of the depots and shelters of case 1. Numbers 0–63 were the information of the depots and shelters of case 2. In Table 5, columns 1 and 5 are the shelter and depot numbers. Columns 2 and 6 are the coordinates of the shelter and depot (unit: miter), where the coordinate system is the European petroleum survey group: 6677 (EPSG:6677). TW1 and TW2 represent two different time windows (unit: minute). The parameters of GA-LNS are the same as those in Section 5.1. The simulation results and the data of different cases

are shown in Figures 8 and 9. The specific routes of the delivery scheme for two cases are shown in Tables 6 and 7, respectively. It can be seen that the constraints of different time windows affect the optimal delivery scheme of EVs.

Table 5. The time windows and coordinates of two cases.

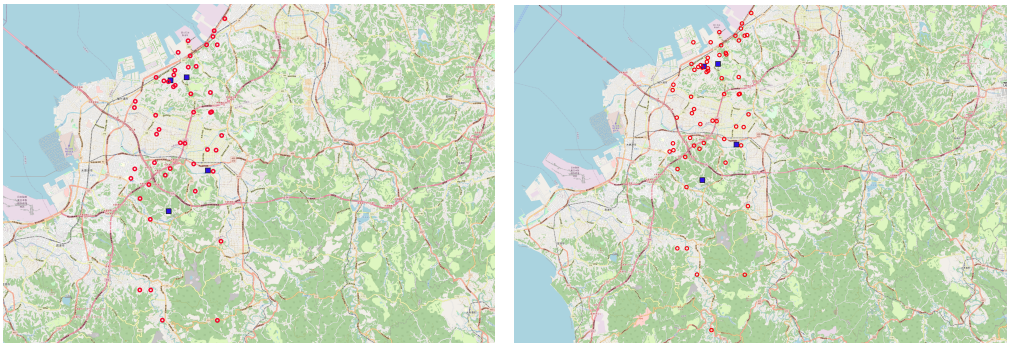
Num.	Coordinate	TW1	TW2	Num.	Coordinate	TW1	TW2
0	(14,630, −61,892)	(0, 1236)	(0, 1236)	32	(12,335, −70,787)	(358, 405)	(10, 1126)
1	(17,465, −68,665)	(0, 1236)	(0, 1236)	33	(13,764, −65,592)	(449, 504)	(0, 1125)
2	(15,848, −61,670)	(0, 1236)	(0, 1236)	34	(16,515, −70,226)	(0, 1112)	(50, 1112)
3	(14,500, −71,735)	(0, 1236)	(0, 1236)	35	(15,203, −59,802)	(31, 100)	(0, 1114)
4	(13,504, −64,524)	(0, 1127)	(0, 1127)	36	(16,392, −68,179)	(87, 158)	(30, 1112)
5	(15,729, −66,635)	(0, 1125)	(0, 1125)	37	(16,380, −64,290)	(0, 1113)	(0, 1113)
6	(13,577, −65,948)	(0, 1129)	(0, 1129)	38	(16,115, −60,059)	(283, 344)	(60, 1107)
7	(18,471, −73,989)	(727, 782)	(727, 782)	39	(13,113, −72,350)	(665, 716)	(0, 1110)
8	(15,379, −66,590)	(0, 1130)	(0, 1130)	40	(14,243, −69,027)	(0, 1106)	(0, 1106)
9	(16,566, −60,848)	(621, 702)	(0, 1127)	41	(18,158, −59,226)	(479, 522)	(479, 522)
10	(17,355, −59,229)	(0, 1130)	(0, 1130)	42	(14,473, −62,002)	(567, 624)	(0, 1105)
11	(13,534, −61,683)	(255, 324)	(255, 324)	43	(16,169, −62,913)	(264, 321)	(20, 1125)
12	(17,868, −68,733)	(534, 605)	(534, 605)	44	(12,999, −69,737)	(166, 235)	(0, 1127)
13	(14,617, −68,528)	(357, 410)	(0, 1129)	45	(14,952, −61,148)	(68, 149)	(10, 1126)
14	(14,984, −62,273)	(448, 505)	(448, 505)	46	(17,833, −58,609)	(16, 80)	(0, 1129)
15	(18,087, −67,141)	(0, 1107)	(0, 1107)	47	(11,963, −63,483)	(359, 412)	(359, 412)
16	(14,852, −62,383)	(30, 92)	(30, 92)	48	(17,656, −64,307)	(541, 600)	(0, 1123)
17	(17,743, −64,259)	(567, 620)	(0, 1106)	49	(18,211, −79,941)	(448, 509)	(20, 1125)
18	(18,726, −57,264)	(384, 429)	(384, 429)	50	(17,433, −67,095)	(1054, 1127)	(0, 1127)
19	(11,930, −68,561)	(475, 528)	(0, 1105)	51	(13,154, −77,664)	(0, 1122)	(0, 1122)
20	(11,898, −63,952)	(99, 148)	(0, 1112)	52	(13,695, −59,809)	(1001, 1066)	(70, 1126)
21	(17,926, −58,174)	(179, 254)	(0, 1110)	53	(16,519, −60,722)	(0, 1123)	(0, 1123)
22	(17,649, −62,809)	(278, 345)	(0, 1106)	54	(18,369, −59,175)	(725, 786)	(0, 1121)
23	(13,422, −68,090)	(10, 73)	(0, 1136)	55	(11,975, −69,158)	(0, 1124)	(0, 1124)
24	(11,640, −69,272)	(0, 1135)	(0, 1135)	56	(14,312, −66,889)	(286, 347)	(286, 347)
25	(14,125, −61,943)	(812, 883)	(0, 1133)	57	(14,970, −62,056)	(186, 257)	(0, 1105)
26	(15,954, −58,929)	(732, 777)	(732, 777)	58	(13,788, −62,190)	(95, 158)	(95, 158)
27	(14,039, −79,934)	(65, 144)	(0, 1131)	59	(14,328, −61,783)	(385, 436)	(0, 1101)
28	(12,324, −77,661)	(169, 224)	(169, 224)	60	(15,314, −84,753)	(35, 87)	(0, 1111)
29	(14,876, −61,496)	(0, 1130)	(0, 1130)	61	(12,261, −66,333)	(471, 534)	(471, 534)
30	(18,507, −66,038)	(261, 316)	(0, 1128)	62	(17,658, −59,643)	(0, 1110)	(0, 1110)
31	(15,968, −60,911)	(546, 593)	(0, 1128)	63	(13,769, −68,755)	(562, 629)	(0, 1100)

Table 6. The specific routes of the delivery scheme for case 1.

Number of EV	Delivery Route of TW1	Number of EV	Delivery Route of TW2
EV1	0-27-30-32-24-42-39-1	EV1	0-48-4-9-44-47-14-5-7-0
EV2	0-44-43-12-29-25-5-4-2	EV2	0-25-24-22-33-8-12-32-30-2
EV3	1-11-10-49-51-8-1	EV3	1-6-51-11-41-13-1
EV4	1-16-20-22-48-9-7-0	EV4	1-37-43-49-19-10-38-36-40-0
EV5	2-35-34-18-41-1	EV5	2-17-21-20-18-50-46-1
EV6	2-23-28-13-19-17-1	EV6	2-23-15-28-29-3
EV7	3-36-40-38-47-14-37-15-6-50-1	EV7	3-16-42-45-39-35-27-34-26-31-2
EV8	3-46-45-21-33-31-26-2		

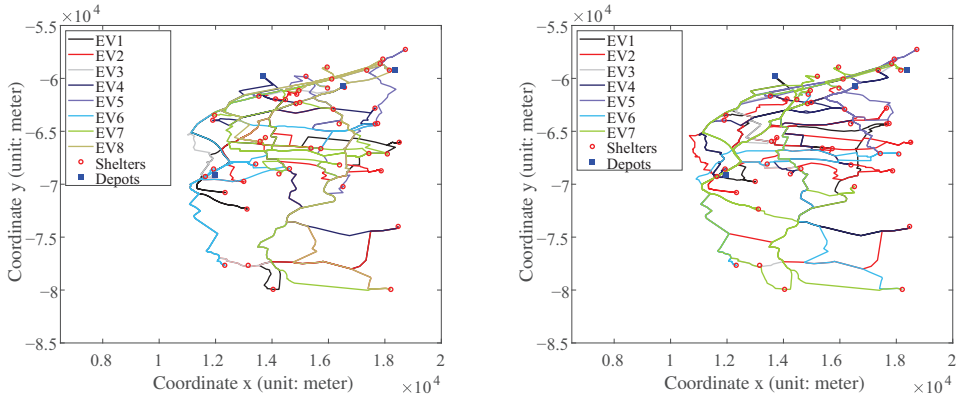
Table 7. The specific routes of the delivery scheme for case 2.

Number of EV	Delivery Route of TW1	Number of EV	Delivery Route of TW2
EV1	0-22-12-2	EV1	0-54-13-53-45-39-41-26-5-1
EV2	0-15-45-56-19-17-51-3	EV2	0-28-31-6-17-46-51-23-27-41-3
EV3	0-16-28-47-48-54-52-50-3	EV3	0-36-40-38-9-14-48-49-1
EV4	1-58-57-32-41-25-1	EV4	1-63-60-55-15-44-61-49-1
EV5	1-36-44-43-55-49-10-39-37-3	EV5	1-47-43-10-19-29-50-3
EV6	1-27-21-13-31-3	EV6	1-4-62-57-35-21-0
EV7	2-8-11-6-9-1	EV7	2-22-33-32-2
EV8	2-23-24-18-42-3	EV8	2-59-25-1
EV9	2-35-30-59-61-26-40-1	EV9	2-34-18-30-20-37-3
EV10	3-60-29-33-62-5-1	EV10	3-58-56-52-7-1
EV11	3-46-34-38-14-4-1	EV11	3-16-11-24-8-12-2
EV12	3-20-63-7-53-3		



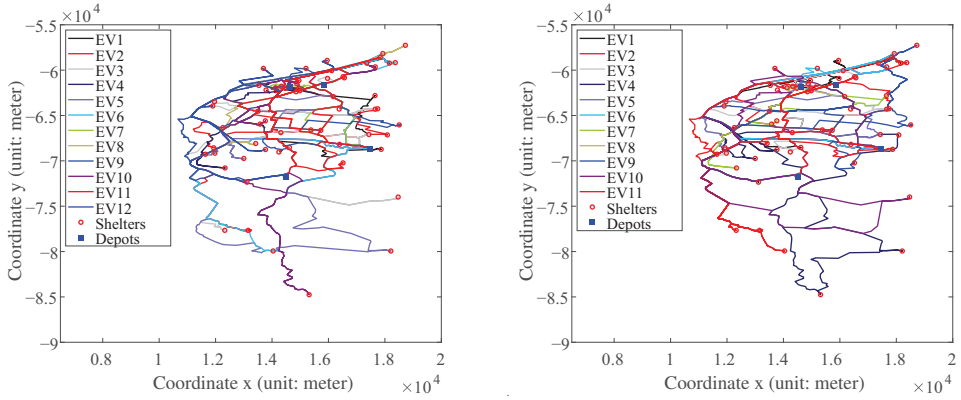
(**a**) (**b**)

Figure 7. Map of depots and shelters situation. (**a**) The depots and shelters situation of case 1. (**b**) The depots and shelters situation of case 2.



(**a**) (**b**)

Figure 8. Optimal delivery scheme of different time windows for case 1. (**a**) The delivery scheme of TW1 for case 1. (**b**) The delivery scheme of TW2 for case 1.



(**a**) (**b**)

Figure 9. Optimal delivery scheme of different time windows for case 2. (**a**) The delivery scheme of TW1 for case 2. (**b**) The delivery scheme of TW2 for case 2.

6. Conclusions

In this paper, we studied an energy saving-oriented multi-depot vehicle routing problem with time windows (ESMDVRPTW) which was solved by an effective method consisting of the Floyd-NL algorithm and GA-LNS algorithm. Since the post-disaster transportation network is complicated, we first used the Floyd-NL algorithm to obtain the

minimal travel cost between any two vertices in real-time. The complexity of the Floyd-NL algorithm is less than the Floyd algorithm in solving the post-disaster transportation network. In the second stage, we used the GA-LNS algorithm to solve the delivery scheme. Simulation results of the benchmark showed the performance of the GA-LNS outperforms GA, SA, TS. Finally, we used a test on two real cases acquired from the OpenStreetMap (OSM) in Ichihara city, Japan, to verify the effectiveness of the proposed two-stage approach.

Author Contributions: Conceptualization, Y.W.; Funding acquisition, Y.W.; Methodology, Q.L. and Y.W.; Software, P.X.; Validation, P.X.; Writing—original draft, P.X. and Q.L.; Writing—review & editing, Q.L. and Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 61973052 and the National Natural Science Foundation of China under Grant 62173062.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to acknowledge the Toyota Motor Corporation for support in this work. The authors would also like to thank Shohei Kishi of Sophia University, Tokyo, for his help with the graph modeling used in the practical example.

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

References

1. Dantzig, G.B.; Ramser, J.H. The truck dispatching problem. *Manag. Sci.* **1959**, *6*, 80–91. [CrossRef]
2. Oyola, J.; Løkketangen, A. GRASP-ASP: An algorithm for the CVRP with route balancing. *J. Heuristics* **2014**, *20*, 361–382. [CrossRef]
3. Tao, Y.; Wang, F. An effective tabu search approach with improved loading algorithms for the 3L-CVRP. *Comput. Oper. Res.* **2015**, *55*, 127–140. [CrossRef]
4. 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. [CrossRef]
5. Sánchez, D.G.; Tabares, A.; Faria, L.T.; Rivera, J.C.; Franco, J.F. A Clustering Approach for the Optimal Siting of Recharging Stations in the Electric Vehicle Routing Problem with Time Windows. *Energies* **2022**, *15*, 2372. [CrossRef]
6. Erdelić, T.; Carić, T. Goods delivery with electric vehicles: Electric vehicle routing optimization with time windows and partial or full recharge. *Energies* **2022**, *15*, 285. [CrossRef]
7. Bae, H.; Moon, I. Multi-depot vehicle routing problem with time windows considering delivery and installation vehicles. *Appl. Math. Model.* **2016**, *40*, 6536–6549. [CrossRef]
8. Adelzadeh, M.; Mahdavi Asl, V.; Koosha, M. A mathematical model and a solving procedure for multi-depot vehicle routing problem with fuzzy time window and heterogeneous vehicle. *Int. J. Adv. Manuf. Technol.* **2014**, *75*, 793–802. [CrossRef]
9. Zhang, X.; Zhang, Z.; Zhang, Y.; Wei, D.; Deng, Y. Route selection for emergency logistics management: A bio-inspired algorithm. *Saf. Sci.* **2013**, *54*, 87–91. [CrossRef]
10. Chang, F.S.; Wu, J.S.; Lee, C.N.; Shen, H.C. Greedy-search-based multi-objective genetic algorithm for emergency logistics scheduling. *Expert Syst. Appl.* **2014**, *41*, 2947–2956. [CrossRef]
11. He, Y.; Wen, J.; Huang, M. Study on emergency relief VRP based on clustering and PSO. In Proceedings of the IEEE 2015 11th International Conference on Computational Intelligence and Security (CIS), Shenzhen, China, 19–20 December 2015; pp. 43–47.
12. Liu, D.; Ji, S. Research on efficient online planning of emergency logistics path based on double-layer ant colony optimization algorithm. *Int. J. Comput. Appl.* **2019**, *41*, 400–406. [CrossRef]
13. Li, K.; Chen, H.; Zhao, J.; Eriksson, L.; Gao, J. An advanced control strategy for engine thermal management systems with large pure time delay. *Appl. Therm. Eng.* **2023**, *224*, 120084. [CrossRef]
14. Hou, S.; Yin, H.; Pla, B.; Gao, J.; Chen, H. Real-time energy management strategy of a fuel cell electric vehicle with global optimal learning. *IEEE Trans. Transp. Electrif.* **2023**. [CrossRef]
15. Hou, S.; Yin, H.; Xu, F.; Benjamín, P.; Gao, J.; Chen, H. Multihorizon predictive energy optimization and lifetime management for connected fuel cell electric vehicles. *Energy* **2023**, *266*, 126466. [CrossRef]
16. Zhu, W.; Guo, B.; Li, Y.; Yang, Y.; Xie, C.; Jin, J.; Gooi, H.B. Uncertainty quantification of proton-exchange-membrane fuel cells degradation prediction based on Bayesian-Gated Recurrent Unit. *eTransportation* **2023**, 100230. [CrossRef]
17. Wu, Y.; Zhang, J.; Shen, T. A logical network approximation to optimal control on a continuous domain and its application to HEV control. *Sci. China Inf. Sci.* **2022**, *65*, 1–18. [CrossRef]
18. Liu, Q.; Xu, P.; Wu, Y.; Shen, T. A hybrid genetic algorithm for the electric vehicle routing problem with time windows. *Control Theory Technol.* **2022**, *20*, 279–286. [CrossRef]

19. Liu, Q.; Xu, P.; Wu, Y. Electric vehicle routing optimization considering the inclines of roads in post-disaster charging relief. In Proceedings of the IEEE 2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI), Nanjing, China, 28–30 October 2022; pp. 1–6.
20. Liu, Q.; Xu, P.; Wu, Y.; Shen, T. A two-stage algorithm for vehicle routing problem with timed-path in disaster response. In Proceedings of the IEEE 2021 60th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Tokyo, Japan, 8–10 September 2021; pp. 36–41.
21. Floyd, R.W. Algorithm 97: Shortest path. *Commun. ACM* **1962**, *5*, 345. [CrossRef]
22. Pisinger, D.; Ropke, S. Large neighborhood search. In *Handbook of Metaheuristics*; Springer: Berlin, Germany, 2010; pp. 399–419.
23. Righini, G.; Salani, M. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Comput. Oper. Res.* **2009**, *36*, 1191–1203. [CrossRef]
24. Kramer, O.; Kramer, O. *Genetic Algorithms*; Springer: Berlin, Germany, 2017.
25. Wei, L.; Zhang, Z.; Zhang, D.; Leung, S.C. A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. *Eur. J. Oper. Res.* **2018**, *265*, 843–859. [CrossRef]
26. Qiu, M.; Fu, Z.; Eglese, R.; Tang, Q. A Tabu Search algorithm for the vehicle routing problem with discrete split deliveries and pickups. *Comput. Oper. Res.* **2018**, *100*, 102–116. [CrossRef]