An Adaptive Large Neighborhood Search for the Larger-Scale Instances of Green Vehicle Routing Problem with Time Windows

Zixuan Yu,1 Ping Zhang,1 Yang Yu,1 Wei Sun,2 and Min Huang1

1State Key Laboratory of Synthesis Automation for Process Industries, Department of Intelligent Data and Systems Engineering, Northeastern University, Shenyang 110819, China
2Business School, Liaoning University, Shenyang, China

Correspondence should be addressed to Yang Yu; yuyang@ise.neu.edu.cn

Received 5 June 2020; Revised 17 August 2020; Accepted 5 October 2020; Published 29 October 2020

Copyright © 2020 Zixuan Yu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Due to huge amount of greenhouse gases emission (such as CO2), freight has been adversely affecting the global environment in facilitating the global economy. Therefore, green vehicle routing problem (GVRP), aiming to minimize the total carbon emissions in the transportation, has become a hot issue. In this paper, an adaptive large neighborhood search (ALNS) algorithm is proposed to solve large-scale instances of GVRP. The core of ALNS algorithm is destroy operators and repair operators. In the destroy operators, a new removal heuristic applying to the characteristics of GVRP is proposed. The heuristic can quickly remove customers who bring a large amount of carbon emissions with pertinence, and these customers may be arranged more properly in future repair operators. In the repair operators, a fast insertion method is developed. In the fast insertion method, the feasibility of a new route is judged by checking the constraints of partial customers after the inserted customer, instead of checking the constraints of all customers. Thus, the computational time of the ALNS algorithm is greatly saved. Computational experiments were performed on Solomon benchmark with 100 customers and Homberger benchmark instances with up to 1000 customers. Given the same computational time, the proposed ALNS improves the average accuracy by 8.49% compared with the classic ALNS. In the optimal situation, the improvement can achieve 33.61%.

1. Introduction

The burning of fossil fuels is the main source of greenhouse gases, and the transportation industry is the largest consumer of petroleum products and the main culprit of global air pollution [1]. Therefore, in recent years, more and more attention has been paid to the green issue in the field of VRP [2].

Green VRP focuses on energy consumption and environmental issues. In recent years, GVRP has been widely studied [2, 3] and the exact solution of GVRP can be obtained [4–6], but it is far from satisfying the requirements of practical application. Therefore, a lot of metaheuristic algorithms are developed. Among many metaheuristic methods, ALNS is an effective algorithm with outstanding performance in VRP [7], especially for the large-scale instances. Up to now, there are no published papers, in which the proposed algorithm can solve GVRP instances with more than 500 customer points in a reasonable computational time in a practical application. The existing ALNS for VRP/PDP can be used to solve GVRP by changing the objective function to the total carbon emission but did not consider the characteristics of GVRP. Thus, an ALNS algorithm is developed to solve the large-scale instances of GVRP according the characteristics of GVRP.

1.1. Related Work. The current research on green routing mainly includes three types of problems: pollution routing problem (PRP), energy minimizing vehicle routing problem (EMVRP), and green vehicle routing problem (GVRP).

PRP consists of routing a number of vehicles to serve a set of customers within preset time windows and determining their speed on each route segment, so as to minimize a function comprising fuel, emission, and driver costs [8]. The two mixed-integer convex optimization models for PRP were proposed by Fukasawa et al. [9]. The nonlinear...
mixed-integer programming model for PRP was presented by Majidi et al. [10]. A bi-objective model for pickup and delivery PRP with minimizing total system cost and fuel consumption to reduce carbon emissions was proposed by Abad et al. [11]. For green transportation scheduling with pickup time and transport mode selections, an Evolutionary Strategy-Based Memetic Pareto Optimization was developed by Guo et al. [12].

EMVRP aims to minimize the total energy used in the routing [13]. In EMVRP, the energy consumption over each arc is defined as the product of the arc length and the weight of the vehicle. For EMVRP, Fukasawa et al. [14] proposed two new mixed-integer linear programming formulations: an arc-load formulation and a set partitioning formulation, and proposed a branch-and-cut algorithm for the arc-load formulation and a branch-cut-and-price algorithm for the set partitioning formulation strengthened by additional constraints.

Most studies of GVRP considered vehicle travel time and weight to be key factors in greenhouse gas emissions [3, 6, 15]. Bektas et al. [3] believe that green vehicle routing is a VRP that considers explicitly carbon dioxide equivalent emissions and reviewed some of the latest developments in this field. Sawik et al. [16] focus on multiobjective GVRP while taking into account the minimization of noise, pollution, and fuel consumption.

For the research on the exact solution of green routing, Bektas and Laporte [4] developed a branch and price (BP) for PRP. A BP algorithm for time-dependent PRP was developed by Franceschetti et al. [17]. Yu et al. [18] used integrated scheduling for reducing carbon emissions of pickup and delivery problem (PDP). Yu et al. [6] developed a branch-and-price algorithm for heterogeneous fleet green vehicle routing problem (HFGVRP). The algorithm can solve the instances with 100 customers. Yu et al. [6] developed an exact algorithm for bi-objective green ride-sharing problem (BGRSP).

Due to the complexity of GVRP, the exact algorithm can solve up to 100 customer instances [6], which is far from meeting the requirements of practical applications. Therefore, many studies have focused on heuristic or metaheuristic algorithms. Xiao et al. [19] developed a simulated annealing (SA) algorithm for CVRP with fuel consumption rate (FCVRP), which can solve up to 483 customers. Übeda et al. [20] developed a tabu search (TS) algorithm for GVRP. Zhang et al. [21] developed a hybrid artificial bee colony algorithm for environmental VRP. Montoya et al. [22] developed a multispace sampling heuristic for GVRP. Ene et al. [23] developed a hybrid metaheuristic algorithm for HFGVRP with up to 100 customers. Shui and Szeto [24] developed a hybrid rolling horizon artificial bee colony algorithm for dynamic green bike repositioning problem. Wang and Szeto [25] developed a large neighborhood search in a bike-sharing network. Based on a multigraph reformulation, Andelmin and Bartolini [26] developed a multi-start local search heuristic for GVRP with up to 470 customers. Li et al. [27] developed an ant colony optimization algorithm for multidepot GVRP with multiple objectives. Rastani et al. [28] developed an ALNS for VRP to reduce emissions stemming from logistics operations. Rezaei et al. [29] developed a GA algorithm and a SA algorithm for HFGVRP. Zhu and Hu [30] developed a hybrid algorithm (HA) for time-dependent green vehicle routing problem (TDGVRP) for up to 200 customers.

Adaptive large neighborhood search (ALNS) is an effective neighborhood search heuristic for large-scale instances of NP-hard problems. ALNS is composed of a number of competing subheuristics that are used with a frequency corresponding to their historic performance [7]. ALNS greatly improves the common shortcoming of traditional neighborhood search (NS), i.e., the difficulty in moving out of the current search space.

Due to the outstanding performance of ALNS, it is widely used in many variants of VRP. Laporte et al. [31] developed an ALNS for CARPSD (capacitated arc-routing problem with stochastic demands). Azi et al. [32] developed an ALNS algorithm for VRP with multiple routes. Keskin and Çatay [33] developed an ALNS for electric VRPTW. Mancini [34] developed a large neighborhood search (LNS) for hybrid VRP. Bach et al. [35] developed an ALNS for distance-constrained capacitated vehicle routing problem (DCVRP). Gu et al. [36] developed an ALNS for the VRP with commodity-constrained split delivery. Hof and Schneider [37] developed an ALNS for simultaneous pickup and delivery VRP with up to 1000 customers. Lahiani et al. [38] developed a hybrid ALNS for multidepot open VRP with up to 288 customers. Sacramento et al. [39] developed an ALNS for drones VRP with up to 375 customers.

Up to now, there are no published papers, in which the proposed algorithm can solve GVRP instances with more than 500 customer points in a reasonable computational time in a practical application. Although the ALNS for VRP can be used to solve GVRP by changing the objective function, the ALNS did not consider the characteristics of GVRP. Thus, an ALNS according to the characteristics of GVRP is developed to solve the large-scale instances with 1000 customers of GVRP.

1.2. Our Contributions. The main contributions are as follows:

(i) An ALNS for GVRP is developed to solve the large-scale instances with 1000 customers. In the same time, the proposed ALNS improves the average accuracy by 8.49% compared with the classic ALNS. In the optimal situation, the improvement can achieve 33.61%.

(ii) A new type of removal heuristic in destroy operators was proposed according to the characteristics of GVRP.

(iii) To save computational time, a fast insertion method was proposed. In the fast insertion method, the feasibility of the new route is judged by checking the constraints of partial customers after the inserted customer, instead of checking the constraints of all customers. Thus, the total computational time is saved by 78.9%.
The rest of the paper is organized as follows: in Section 2, GVRPTW is described and formulated; in Section 3, an ALNS algorithm for the large-scale instances with 1000 customers of GVRPTW is proposed. A new type of removal heuristic according to the characteristics of GVRP is proposed. To save computational time, a fast insertion method was proposed. In the fast insertion method, the feasibility of the new route is judged by checking the constraints of partial customers after the inserted customer, instead of checking the constraints of all customers. Section 4 presents computational results, including the experiments of Solomon instances with 100 customers and Homberger instances with up to 1000 customers. In the same time, the proposed ALNS improves the average accuracy by 8.49% compared with the classic ALNS. In the optimal situation, the improvement can achieve 33.61%. Section 5 gives conclusions.

2. Description and Formulation of GVRPTW

2.1. Problem Description of GVRPTW. GVRPTW focuses on routing vehicles to serve a group of customers within their time windows to minimize the total carbon emissions. $G = (V, E)$, defined in Euclidean space, is a complete directed graph with $V = \{0, 1, \ldots, n + 1\}$. Nodes 0 and $n + 1$ denote the start and the end depot, respectively. Let $V_0 = V \setminus \{0, n + 1\}$ be the set of customers. For customer $i$, the quantity $q_i$ of products is to be served by the vehicle within the time window $[a_i, b_i]$. In addition, the service time $s_i$ is required. $E$ represents the set of arcs. For each arc $(i, j) \in E$, let $d_{ij}$ be the distance from node $i$ to node $j$. An unlimited fleet of vehicles is located at the depot. The objective of GVRPTW is to minimize the carbon emissions, while considering the following assumptions: (1) all customers must be served; (2) each customer must be visited only once with only one vehicle; (3) each feasible route starts and ends at the depot; (4) the service at each customer $i$ must begin within the customer time window $[a_i, b_i]$, and the vehicle will wait if it arrives before the ready time $a_i$; and (5) each vehicle needs to return to the depot before the due time $b_{n+1}$.

2.2. Mathematical Model of GVRPTW

2.2.1. Notations. The following notations are used:

- $0$: start depot.
- $n + 1$: end depot.
- $v$: vehicle speed.
- $V$: set of nodes $V = \{0, 1, \ldots, n + 1\}$.
- $V_0$: set of customers $V_0 = V \setminus \{0, n + 1\}$.
- $i, j$: nodes $i, j \in V$.
- $d_{ij}$: distance between node $i$ and node $j$.
- $f_{ij}$: vehicle’s payload through arc $(i, j)$.
- $q_i$: quantity of products at the customer $i$.
- $[a_i, b_i]$: time window when service may start at node $i$, where $a_i$ and $b_i$ represent the earliest and latest time, respectively.
- $s_i$: service time at node $i$, and the service time at depot (i.e., node 0 and $n + 1$) is set to 0.
- $c_{ij}$: carbon emissions between node $i$ and node $j$.
- $u_i$: time of vehicle arriving at customer $i$.
- $d_i$: load of vehicle leaving from customer $i$.
- $p$: a feasible route visited by a vehicle, $p = (0, 1, \ldots, l, n + 1)$, where $l$ is number of customers in the route $p$.
- $Q$: capacity of vehicle.
- $w$: curb weight of vehicles.
- $x_{ij}$: decision variables; if the vehicle is driving from node $i$ to node $j$, then $x_{ij} = 1$; otherwise, $x_{ij} = 0$.
- $t_{ij}$: travel time between node $i$ and node $j$.

2.2.2. Mathematical Formulation. The carbon emissions calculation formula is used [40]:

$$
e_{ij} = r \left( \frac{a_i \times d_{ij}}{v} + \frac{\alpha_2 \times (w + f_{ij}) \times d_{ij}}{v} \right),$$

where $r$ represents carbon emissions index parameter, by which the fuel consumption on arc $(i, j)$ can be transformed into carbon emissions; $\alpha_1$ represents the emissions unrelated to the vehicle mass; $\alpha_2$ represents the emissions linear to the vehicle mass; $w$ represents curb weight of vehicles; and $v$ represents the vehicle speed. It can be seen from equation (1) that the carbon emissions from node $i$ and node $j$ are related to the following factors: distance, vehicle speed, and the total weight of the vehicle between the two nodes.

The mathematical programming model of GVRP is formulated as follows:

\[
\begin{align*}
\min & \sum_{(i,j) \in E} \sum_{i \in V_0} x_{ij} \\
\text{s.t. } & \sum_{(i,j) \in E} x_{ij} = 1, \quad \forall i \in V_0, \\
& \sum_{(0,j) \in E} x_{ij} = 1, \quad \forall i \in V_0, \\
& \sum_{(i,n+1) \in E} x_{ij} = 1, \quad \forall i \in V_0, \\
& x_{ij} \geq (u_i + s_i + t_{ij}) x_{ij}, \quad \forall (i, j) \in E, \\
& u_i \geq u_i \leq b_i, \quad \forall i \in V_0, \\
& d_j \geq (d_i + q_j) x_{ij}, \quad \forall (i, j) \in E, \\
& 0 \leq d_i \leq Q, \quad \forall i \in V_0, \\
& u_i \geq 0, \quad \forall i \in V.
\end{align*}
\]
The vehicle's current load $f_{ij}$ and the distance $d_{ij}$ between node $i$ and node $j$. Therefore, equation (2) is nonlinear with decision variables $x_{ij}$. Constraint (3) means that each customer needs to be served once. Constraints (4) and (5) represent the flow balance of the depot and the customers, respectively. Constraints (6) and (8) represent arrival time update constraints and load update constraints, respectively. Constraint (7) is the time window constraints. Constraint (9) is the load constraints. Constraint (11) defines the domains of decision variables.

Since the model is nonlinear, it cannot be solved by the solver for linear programming problem directly, such as CPLEX. To obtain the optimal solution of GVRP, some researchers developed the branch-and-price (BP) algorithm [6] and branch-and-cut and price (BCP) algorithm [14]. Although both BP and BCP are based on the set partitioning model transformed by the mathematical programming model, the BP and BCP algorithms can solve up to 100 customers. Therefore, some metaheuristic algorithms have been proposed. Up to now, there are no published papers, in which the proposed algorithm can solve GVRP instances with more than 500 customers in a reasonable computational time in a practical application. Although the ALNS for VRP can be used to solve GVRP by changing the objective function, the ALNS did not consider the characteristics of GVRP. Thus, an ALNS algorithm according to the characteristics of GVRP is developed. The proposed ALNS efficiently solves the large-scale instances with more than 500 customers even 1000 customers of GVRP.

### 3. Adaptive Large Neighborhood Search for GVRPTW

#### 3.1. Framework of Adaptive Large Neighborhood Search

ALNS, proposed by Ropke and Pisinger [7], adaptively chooses among a number of insertion and removal heuristics to intensify and diversify the search and the algorithm is robust and to some extent self-calibrating [41].

To describe the framework of ALNS in brief, the pseudocode of ALNS algorithm is shown in Algorithm 1. The initial solution $(x)$ is obtained by a construction heuristic, where each customer $i$ generates route $p = (0, i, n + 1)$, until all customers are dealt with.

When judging whether to update the current solution, the simulated annealing acceptance criterion [8] is used to prevent getting trapped in a local minimum [7].

Implicitly defining the solution space through destroy operators (i.e., step (1)) and repair operators (i.e., step (2)), rather than explicitly defining the solution space, is a significant difference between ALNS and traditional neighborhood search (NS). Therefore, ALNS greatly improves the common shortcoming of traditional NS, i.e., the difficulty in moving out of the current search space.

#### 3.2. Destroy Operators

In the destroy operators, there are several removal heuristics. In each removal heuristic, $x'$ is updated by removing $z$ customers from $x'$. The removal heuristics are selected dynamically according to their past performance [8]. The classic ALNS used random removal heuristic, worst removal heuristic, and Shaw removal heuristic in the destroy operations. In addition, forward load removal heuristic (Section 3.2.4) is proposed according to the characteristics of GVRP. For more removal heuristics, refer to Hemmelmayr et al. [42] and Alinaghian and Shokouhi [43].

##### 3.2.1. Random Removal Heuristic

Random removal heuristic was proposed by Ropke and Pisinger [7]. Random removal heuristic selects $z$ customers randomly and removes them from the solution.

##### 3.2.2. Worst Removal Heuristic

Worst removal heuristic was proposed by Ropke and Pisinger [7]. Worst removal heuristic removes the $z$ customers with the biggest difference in the carbon emissions. The difference in the carbon emissions is the difference in the carbon emissions between the original solution and the solution produced by removing the $z$ customers.

##### 3.2.3. Shaw Removal Heuristic

Shaw removal heuristic was proposed by Shaw [44]. The idea of Shaw removal heuristic is to remove customers that are similar, in order to obtain a promising solution in repair operators.

##### 3.2.4. Forward Load Removal Heuristic

In GVRP, the carbon emissions between two customers are related to the distance and the load. Therefore, in the pickup/delivery problem, if the customer with a high demand but is served earlier/later, then the carbon emissions will be more than when the customer is served later/earlier. Thus, the forward load removal heuristic is proposed according to the characteristics of GVRP, to remove customers with high demand and served early.

The pseudocode for forward load removal heuristic is shown in Algorithm 2.

In step (1), $M$ is the order of selected customer.

In step (2), the customer before $M$ in the route $p$ is the customer served earlier, i.e., $j = 1$ to $M - 1$.

In step (3), if the customer’s demand is higher than the demand of $H[M]$, i.e., $q_j > H[M]$, demand, then the customer is the high-demand customer. Thus, the customer is removed from $p$.

Steps (2) and (3) guarantee that the customers with high-demand served earlier are removed.

Forward load removal heuristic can be seen as a compromise between random removal heuristic and worst removal heuristic. Compared to random removal heuristic, forward load removal heuristic not only increases the calculation time a little but also removes the customers with huge carbon emissions. Compared with worst removal heuristic, forward load removal heuristic does not calculate the carbon emissions of each customer but uses the characteristics of the problem to simplify the judgment of customers with huge carbon emissions, which greatly reduces the calculation time of a single heuristic.
3.3. Repair Operators. In the repair operators, there are several insertion heuristics. In each insertion heuristics, $x'$ is updated by reinserting $z$ customers (removed in destroy operators) into $x'$. (+çhe insertion heuristics are selected dynamically according to their past performance [8]. (+çhe proposed ALNS used greedy insertion heuristic and regret heuristic in repair operators. In addition, to save computational time, a fast insertion method was proposed. In the fast insertion method, the feasibility of the new route is judged by checking the constraints of partial customers after the inserted customer, instead of checking the constraints of all customers. Thus, the judgment time of the feasibility of the new route is significantly saved.

3.3.1. Greedy Insertion Heuristic. Greedy insertion heuristic is proposed by Ropke and Pisinger [7]. In greedy insertion heuristic, for all the feasible positions corresponding to all customers that have been removed, find a customer with the smallest carbon emissions increase to insert.

3.3.2. Regret Insertion Heuristic. Regret insertion heuristic is proposed by Ropke and Pisinger [7]. In the regret heuristic, customers are treated in the order of their regret value. Informally speaking, the insertion, which will bring regret most if it is not done now, will be chosen. Regret-2 and Regret-3 insertion heuristics [7] were used.

3.3.3. Fast Insertion Method. In repair operators, the new route generated by inserting customer needs to be judged whether or not feasible. In traditional repair operators, the constraints of all the customers after the inserted customer
are checked to judge whether the new route is feasible. It takes too much time. In some situations (i.e., Situations 1 and 2), the feasibility of the new route can be judged by checking the constraints of partial customers after the inserted customer, instead of checking the constraints of all the customers. It is worth noting that this method is a general method and can be used in all variants of VRPTW.

Notations used in the section are as follows:

- \( j \): customer \( j \)
- \( r \): inserted customer \( r \)
- \( p = (0, 1,\ldots,j, j+1,\ldots,l, n+1) \): a feasible route;
- \( p_{\text{ins}} = (0, 1,\ldots,j, r, j+1,\ldots,l, n+1) \): a route produced by inserting customer \( r \) into \( p \);
- \( d_t \): the departure time of customer \( j \)
- \( DT = (d_{t_0}, d_{t_1},\ldots,d_{t_j}, d_{t_{j+1}},\ldots,d_{t_l}, d_{t_{n+1}}) \): departure time array of the customers in route \( p \);
- \( DT_{\text{ins}} = (d_{t_0}, d_{t_1},\ldots,d_{t_r}, d_{t_{r+1}},\ldots,d_{t_l}, d_{t_{n+1}}) \): departure time array of the customers in route \( p_{\text{ins}} \).

\textbf{Situation 1.} If there exists a customer \( k \) after the inserted customer \( r \) in the new route (i.e., \( p_{\text{ins}} = (0, 1,\ldots,j, r, j+1,\ldots,k, l, n+1) \)), with \( d_{t_k} \) in \( DT_{\text{ins}} \) equal to \( d_{t_k} \) in \( DT \), then the new route pins is a feasible route.

\textbf{Example 1.} In a feasible route \( p = (0, 1, 2, 3, 4, n+1) \), \( DT = (0, 1, 4, 5, 6, 7) \) is the departure time array of the customers in \( p \). For customer 9, \( t_{10} = t_{20} = 1 \), \([a_9, b_9] = [2, 3] \). Customer 9 can be inserted into customers 1 and 2 in \( p \) without violating the time window constraint of customer 2. In the new route \( p_{\text{ins}} = (0, 1, 9, 2, 3, 4, n+1) \), if \( d_{t_2} \) in \( DT_{\text{ins}} \) is equal to \( d_{t_2} \) in \( DT \), then \( p_{\text{ins}} \) must be a feasible route. Thus, it is unnecessary to check the constraints of customers 3 and 4.

Situation 1 means that the feasibility of the new route is judged by checking the constraints of partial customers after the inserted customer, instead of checking the constraints of all customers.

\textbf{Situation 2.} If \( r \) cannot be inserted into the route \( p \), then \( r \) cannot be inserted into any route generated by the route \( p \).

\textbf{Example 2.} If customer 9 cannot be inserted into the route \( p = (0, 1, 2, n+1) \), then customer 9 cannot be inserted into \( p_{\text{ins}} = (0, 1, 2, 3, n+1) \) generated by \( p \).

According to Situation 2, the feasibility of the route can be prejudged before inserting \( r \).

The program of the fast insertion method is shown in Algorithm 3.

Step (1) initializes the parameters.

Step (2) checks whether the route \( p \) is generated by a route in which customer \( r \) cannot be inserted into (see situation 1). If yes, output flag = false; otherwise, continue to check the feasibility of the new route.

Step (3) checks the capacity constraint of the new route, i.e., \( q_r + q_s \leq Q \), and the time constraint of customer \( r \), i.e., \( d_{t_j} + t_r \leq b_r \), are satisfied. If yes, continue to check the feasibility of the new route; otherwise, output flag = false.

Step (3-1) calculates the departure time of the inserted customer \( r \).

Step (4) checks if there exists a customer \( k \) after the inserted customer \( r \) in the new route pins with \( d_{t_k} \) in \( DT_{\text{ins}} \) equal to \( d_{t_k} \) in \( DT \) (see situation 1).

Step (4-1) checks the time constraint of customer \( k \), i.e., \( d_{t_{k-1}} + t_{k-1,k} \leq b_k \), is satisfied. If yes, continue to check the feasibility of the new route; otherwise, break and output flag = false.

Step (4-1-1) calculates the departure time of the customer \( k \).

Step (4-1-2) judges whether \( d_{t_k} \) in \( DT_{\text{ins}} \) is equal to \( d_{t_k} \) in \( DT_{\text{ins}} \). If yes, the new route is feasible; therefore, flag is set as true and break. Thus, the feasibility of the new route is judged by checking the constraints of partial customers after the inserted customer, instead of checking the constraints of all customers.

Step (4-2) means that the feasibility of the new route is judged by checking the constraints of all customers after the inserted customer \( r \).

Experiments confirmed that the total computational time through the fast insertion method is saved by 78.9% compared with the traditional insertion method.

In addition, this method is a general method, not only applicable to GVRP but also to all VRPTW variants.

\textbf{3.4. Adaptive Method.} Adaptive method is used to change the probability of heuristics in destroy operators and repair operators according to the performance of the heuristics [7]. The proposed ALNS algorithm uses a probability update formula as follows:

\[
a_{i+1}^r = \rho a_i^r + \frac{f(x') - f(x_{i+1}^r)}{f(x')},
\]

where \( a_i^r \) is the probability of the algorithm \( i \) being called in the \( t \)th iteration, \( \rho \in (0, 1) \) is the evaporation coefficient, and \( f(x') \) is the objective value of the solution \( x' \) in iteration \( t \).

Equation (12) can be seen as a combination of the evaporation method and reward method. To prevent the algorithm from prematurely converging, the probability is given within a fix range \([a_{\text{min}}, a_{\text{max}}]\).

\textbf{3.5. Flowchart of ALNS for GVRP.} The flowchart of ALNS for GVRP is shown in Figure 1.

As shown in Figure 1, the data of a test instance are read first. For more information about experiment instances, refer to Section 4.2. Then, an intimal solution (\( x \)) is obtained by construction heuristic.

In the destroy operators (Section 3.2), one of the three removal heuristics is selected according to their past performance, and the initial probability of each heuristic is equal to 1/3.

In the repair operators (Section 3.3), one of the three insertion heuristics is selected according to their past performance, and the initial probability of each heuristic is equal to 1/3. The fast insertion method is embedded in all insertion heuristics in the repair operators. When judging
the feasibility of each newly generated route, it is used to save judging time.

If the solution \((x')\) generated in this iteration is better than (carbon emission is smaller than) the best solution \((x_{\text{best}})\), update the best solution, i.e., \(x_{\text{best}} \leftarrow x'\).

If the solution \((x')\) generated in this iteration is better than the current solution \((x)\) or meets the simulated annealing acceptance criteria [8], update the current solution, i.e., \(x \leftarrow x'\).

The probability of the heuristics being called in the destroy operators and repair operators keeps changing, according to its performance in the adaptive method (Section 3.4).

When the given running time is reached, the best solution \((x_{\text{best}})\) is output and the algorithm ends.

4. Computational Results

4.1. Software and Hardware Specifications. The BP algorithm was coded in C#, whose RMP and integer branch method were solved by ILOG CPLEX 12.6. Both classic ALNS and the proposed ALNS algorithms are coded in C#. All tests were executed on a laptop with a 2.80-GHz Intel Core TM i5-4200 processor using the Microsoft Windows 10 operating system using 12.00 GB of RAM. The classic ALNS and the proposed ALNS were run 10 times in each instance.

4.2. Experiment Instances

4.2.1. Introduction of the Benchmark Instance. The used GVRPTW instances are derived from Solomon VRPTW benchmark instances [45] with 100 customers and Homberger VRPTW benchmark instances [46] with 200, 400, 600, 800, and 1000 customers. Benchmark instances are divided in three classes according to the geographical distribution of the customers: R (random), C (clustered), or RC (semiclustered) [45]. For more details on the benchmark instances, refer to [45, 46].

4.2.2. Construction of Data Acquisition System. The standard test sets from Solomon benchmark and Homberger benchmark are used in the experiment, the coordinate, and the quantity of products, and time windows and service time of the depot and customers can be obtained. Based on the coordinate of the depot and customers, the Euclidean distance \(d_{ij}\) from node \(i\) to node \(j\) is calculated. Since the vehicle speed is fixed, the calculated distance needs to be scaled to meet the time window constraints, and the distance \(d_{ij}\) was changed to \(d'_{ij} = C_d d_{ij} [47]\) and kept to four decimal places. For the quantity of products of each customer, the quantity was scaled to \(q'_{ij} = C_q q_{ij}\). For simplicity of computation, the time windows were set as 24 h, and time
windows of all customers were scaled accordingly using the coefficient $24/b_0$. Moreover, the time window $[a_i, b_i]$ of customer $i$ in Solomon instances and Homberger instances was modified as $[a_i(24/b_0), b_i(24/b_0)]$ [47]. Furthermore, the service time was set to 1.5 h for all customers [47] in Solomon instances (100 customers), and the service time $s_i$ was modified as $s_i(24/b_0)$ in the Homberger instances (200, 400, 600, 800, and 1000 customers).

Table 1: Value of used parameters of GVRP.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Typical values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>Carbon emissions index parameter</td>
<td>3.164</td>
</tr>
<tr>
<td>$a_1$</td>
<td>Emissions unrelated to the vehicle mass</td>
<td>1.078086</td>
</tr>
<tr>
<td>$a_2$</td>
<td>Emissions linear to the vehicle mass</td>
<td>0.072253</td>
</tr>
<tr>
<td>$W_1$</td>
<td>Represents curb weight of vehicle type 1</td>
<td>2500 kg</td>
</tr>
<tr>
<td>$Q_1$</td>
<td>Capacity of vehicle 1</td>
<td>2000 kg</td>
</tr>
<tr>
<td>$W_2$</td>
<td>Represents curb weight of vehicle type 2</td>
<td>9000 kg</td>
</tr>
<tr>
<td>$Q_2$</td>
<td>Capacity of vehicle 2</td>
<td>10000 kg</td>
</tr>
<tr>
<td>$V$</td>
<td>Vehicle speed</td>
<td>42 km/h</td>
</tr>
<tr>
<td>$C_d$</td>
<td>Distance coefficient</td>
<td>$2 \left(</td>
</tr>
<tr>
<td>$C_q$</td>
<td>Load coefficient</td>
<td>$0.5 \left( 200 \leq</td>
</tr>
</tbody>
</table>

Figure 1: The flowchart of ALNS for GVRP.
According to the investigation in a logistics company in Shenyang, China, two types of vehicles for the computational tests of Homberger instance [46, 48, 49] were selected by the authors. Table 1 describes the value of parameters used in experiments of GVRP.

Table 2 describes the value of parameters used in experiments of ALNS.

In the experiment, the running time of the heuristic algorithm is given according to the scale of the problem (Table 3). Given the same running time, the performance of the algorithm is evaluated by its minimal carbon emissions.

### 4.3. Computational Results of Improved ALNS Algorithm

Since the branch-and-price algorithm developed by Yu et al. [50] can solve up to 100 customer instances, for 100-customer instances, the proposed algorithm is compared with the exact solution to evaluate its performance. For instances with more than 200 customers, there is no exact solution until now, and the proposed algorithm is compared with the classic ALNS algorithm proposed by Ropke and Pisinger [7].

The destroy operators of classic ALNS consist of random removal heuristic, worst removal heuristic, and Shaw removal heuristic, and the repair operators of classic ALNS consist of greedy insertion heuristic and regret insertion heuristic (regret 2 and regret 3). In addition, the objective function is changed to the total carbon emission in the routing.

The proposed ALNS in the paper and its details can refer to Section 3.5.

Table 4 shows the results of Solomon benchmark instances with 100 customers using the vehicles with Q = 2000.

For the instances with 100 customers, only 4 instances can be exactly solved by the branch-and-price for GVRPTW [6]. As shown in Table 4, in the same given computational time (20s), the proposed ALNS improves the accuracy significantly compared with the classic ALNS.

Tables 5–9 show the results of Homberger benchmark instances with 200, 400, 600, 800, and 1000 customers, respectively.

It can be seen from Table 5 that, in the instances of 200 customers, the proposed algorithm is better than the classic algorithm. Compared with the classic ALNS, our proposed ALNS reduced carbon emissions by 7.31% on average. In addition, in the same given computational time, the number of iterations or our ALNS is 3.41 times that of classic ALNS.

The performance of the proposed ALNS in class C (clustered) is particularly outstanding: carbon emissions are reduced by 13.10% on average and, in the instance C1_2_2, reduced by 19.26%.

It can be seen from Table 6 that, in the instances of 400 customers, the proposed algorithm is better than the classic algorithm. Compared with the classic ALNS, our proposed ALNS reduced carbon emissions by 9.37% on average. In addition, in the same given computational time, the number of iterations or our ALNS is 4 times that of classic ALNS.

The performance of the proposed ALNS in class C (clustered) is particularly outstanding: carbon emissions are reduced by 14.48% on average and, in the instance C1_2_2, reduced by 23.67%. In addition, our algorithm performs more...
prominently in the instances with vehicle type 2 (e.g., C2_4_1, C2_4_2, and R2_4_2).

It can be seen from Table 7 that, in the instances of 600 customers, the proposed algorithm is better than the classic algorithm. Compared with the classic ALNS, our proposed ALNS reduced carbon emissions by 7.53% on average. In addition, in the same given computational time, the number of iterations or our ALNS is 4.77 times that of classic ALNS. The performance of the proposed ALNS in class C (clustered) is particularly outstanding: carbon emissions are reduced by 11.14% on average and, in the instance C2_6_1, reduced by 14.13%. In addition, our algorithm performs more prominently in the instances with vehicle type 2.

It can be seen from Table 8 that, in the instance of 800 customers, the proposed algorithm is better than the classic algorithm. Compared with the classic ALNS, our proposed ALNS reduced carbon emissions by 7.91% on average. In addition, in the same given computational time, the number of iterations or our ALNS is 4.97 times that of classic ALNS. The performance of the proposed ALNS in class C (clustered) is particularly outstanding: carbon emissions are reduced by 13.38% on average and, in the instance C2_8_1, reduced by 21.20%

It can be seen from Table 9 that, in the instance of 1000 customers, the proposed algorithm is better than the classic algorithm. Compared with the classic ALNS, our proposed ALNS reduced carbon emissions by 10.32% on average. In addition, in the same given computational time, the number of iterations or our ALNS is 3.56 times that of classic ALNS. The performance of the proposed

### Table 5: Computational results of the instances with 200 customers.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Classic ALNS</th>
<th></th>
<th></th>
<th>Our ALNS</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCE</td>
<td>ITER</td>
<td>MCE</td>
<td>ITER</td>
<td>RCE (%)</td>
<td>QOI</td>
</tr>
<tr>
<td>R1_2_1</td>
<td>813.73</td>
<td>984</td>
<td>794.45</td>
<td>2712</td>
<td>2.37</td>
<td>2.76</td>
</tr>
<tr>
<td>R1_2_2</td>
<td>722.85</td>
<td>1129</td>
<td>708.63</td>
<td>2290</td>
<td>1.83</td>
<td>2.03</td>
</tr>
<tr>
<td>R2_2_1</td>
<td>1096.21</td>
<td>433</td>
<td>1029.41</td>
<td>1474</td>
<td>6.09</td>
<td>3.40</td>
</tr>
<tr>
<td>R2_2_2</td>
<td>1004.70</td>
<td>367</td>
<td>924.46</td>
<td>962</td>
<td>7.99</td>
<td>2.62</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>9.16</td>
<td>3.84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1_2_1</td>
<td>827.44</td>
<td>945</td>
<td>683.45</td>
<td>6379</td>
<td>17.40</td>
<td>6.75</td>
</tr>
<tr>
<td>C1_2_2</td>
<td>727.80</td>
<td>899</td>
<td>587.61</td>
<td>4333</td>
<td>19.26</td>
<td>4.82</td>
</tr>
<tr>
<td>C2_2_1</td>
<td>655.25</td>
<td>519</td>
<td>609.00</td>
<td>2765</td>
<td>7.06</td>
<td>5.33</td>
</tr>
<tr>
<td>C2_2_2</td>
<td>626.34</td>
<td>479</td>
<td>572.09</td>
<td>1209</td>
<td>8.66</td>
<td>2.53</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>13.10</td>
<td>4.86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC1_2_1</td>
<td>648.92</td>
<td>1024</td>
<td>633.69</td>
<td>2263</td>
<td>2.35</td>
<td>2.21</td>
</tr>
<tr>
<td>RC1_2_2</td>
<td>620.04</td>
<td>1175</td>
<td>598.64</td>
<td>1991</td>
<td>3.45</td>
<td>1.70</td>
</tr>
<tr>
<td>RC2_2_1</td>
<td>880.85</td>
<td>404</td>
<td>765.83</td>
<td>1880</td>
<td>13.06</td>
<td>4.66</td>
</tr>
<tr>
<td>RC2_2_2</td>
<td>746.70</td>
<td>447</td>
<td>726.07</td>
<td>956</td>
<td>2.76</td>
<td>2.14</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>5.41</td>
<td>2.68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total average</td>
<td>7.31</td>
<td>3.41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ITER**: number of iterations; **RCEs**: reduced carbon emissions (i.e., $RCE = (MCE_{classical ALNS} - MCE_{our ALNS})/MCE_{classical ALNS} \times 100\%$); **QOI**: quotient of number of iterations (i.e., $QOI = \frac{ITER_{our ALNS}}{ITER_{classic ALNS}}$).

### Table 6: Computational results of the instances with 400 customers.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Classic ALNS</th>
<th></th>
<th></th>
<th>Our ALNS</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCE</td>
<td>ITER</td>
<td>MCE</td>
<td>ITER</td>
<td>RCE (%)</td>
<td>QOI</td>
</tr>
<tr>
<td>R1_2_1</td>
<td>1952.90</td>
<td>1755</td>
<td>1887.10</td>
<td>5563</td>
<td>3.37</td>
<td>3.17</td>
</tr>
<tr>
<td>R1_2_2</td>
<td>1793.61</td>
<td>1945</td>
<td>1743.05</td>
<td>4449</td>
<td>2.82</td>
<td>2.29</td>
</tr>
<tr>
<td>R2_2_1</td>
<td>2686.05</td>
<td>475</td>
<td>2319.57</td>
<td>2845</td>
<td>13.64</td>
<td>5.99</td>
</tr>
<tr>
<td>R2_2_2</td>
<td>2668.67</td>
<td>384</td>
<td>2220.06</td>
<td>1509</td>
<td>16.81</td>
<td>3.93</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>9.16</td>
<td>3.84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1_2_1</td>
<td>2085.39</td>
<td>872</td>
<td>1881.19</td>
<td>6115</td>
<td>9.79</td>
<td>7.02</td>
</tr>
<tr>
<td>C1_2_2</td>
<td>1810.15</td>
<td>1029</td>
<td>1680.64</td>
<td>3526</td>
<td>7.15</td>
<td>3.43</td>
</tr>
<tr>
<td>C2_2_1</td>
<td>1818.39</td>
<td>574</td>
<td>1388.01</td>
<td>4144</td>
<td>23.67</td>
<td>7.22</td>
</tr>
<tr>
<td>C2_2_2</td>
<td>1621.95</td>
<td>482</td>
<td>1341.20</td>
<td>1564</td>
<td>17.31</td>
<td>3.25</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>14.48</td>
<td>5.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC1_2_1</td>
<td>1689.13</td>
<td>1630</td>
<td>1652.51</td>
<td>3539</td>
<td>2.17</td>
<td>2.17</td>
</tr>
<tr>
<td>RC1_2_2</td>
<td>1578.49</td>
<td>2065</td>
<td>1561.58</td>
<td>3751</td>
<td>1.07</td>
<td>1.82</td>
</tr>
<tr>
<td>RC2_2_1</td>
<td>1918.34</td>
<td>460</td>
<td>1823.66</td>
<td>2389</td>
<td>4.94</td>
<td>5.20</td>
</tr>
<tr>
<td>RC2_2_2</td>
<td>1846.80</td>
<td>416</td>
<td>1667.03</td>
<td>1058</td>
<td>9.73</td>
<td>2.54</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>4.48</td>
<td>2.93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total average</td>
<td>9.37</td>
<td>4.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ALNS in class C (clustered) is particularly outstanding: carbon emissions are reduced by 17.72% on average and, in the instance C2_10_2, reduced by 33.61%. In addition, our algorithm performs more prominently in the instances with vehicle type 2.

It can be seen from Tables 5–9 that the proposed ALNS algorithm is superior to the traditional ALNS algorithm in all instances. From the perspective of the geographic distribution of the customers, the proposed algorithm is particularly effective in the instances of class C (clustered) among the three classes (R, C, and RC). Take the instances of class C with 1000 customers as an example, and the algorithm improves the average accuracy by 15.09% compared with the classic ALNS algorithm. In the optimal situation, the improvement can achieve 33.61%.

4.4. Performance Evaluation of the Forward Load Removal Heuristic. In the proposed ALNS algorithm for GVRP, forward load removal heuristic is proposed according to the characteristics of GVRP to replace random removal heuristic in the classic ALNS algorithm. The performance of forward load removal heuristic is evaluated in Table 10.

As shown in Table 10, compared with the classic removal heuristic proposed by [7], the proposed ALNS improved the average accuracy by 0.97%. In addition, in the same time, the number of iterations of the proposed ALNS is 1.3 times that of the classic ALNS.

4.5. Performance Evaluation of the Fast Insertion Method. To evaluate the performance of the fast insertion method, the traditional insertion method and the fast insertion method

### Table 7: Computational results of the instances with 600 customers.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Classic ALNS</th>
<th>Our ALNS</th>
<th>RCE (%)</th>
<th>QOI</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1_6_1</td>
<td>4298.08</td>
<td>4224.55</td>
<td>1.71</td>
<td>3.59</td>
</tr>
<tr>
<td>R1_6_2</td>
<td>3851.56</td>
<td>4408.00</td>
<td>3.26</td>
<td>2.63</td>
</tr>
<tr>
<td>R2_6_1</td>
<td>5264.92</td>
<td>3925.00</td>
<td>10.37</td>
<td>9.93</td>
</tr>
<tr>
<td>R2_6_2</td>
<td>5153.55</td>
<td>1808.00</td>
<td>19.43</td>
<td>6.04</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>4520.57</td>
<td>8.69</td>
<td>5.55</td>
</tr>
<tr>
<td>C1_6_1</td>
<td>4328.34</td>
<td>5877.00</td>
<td>8.77</td>
<td>7.10</td>
</tr>
<tr>
<td>C1_6_2</td>
<td>3950.98</td>
<td>4421.00</td>
<td>11.41</td>
<td>5.06</td>
</tr>
<tr>
<td>C2_6_1</td>
<td>3483.27</td>
<td>3852.00</td>
<td>14.13</td>
<td>7.93</td>
</tr>
<tr>
<td>C2_6_2</td>
<td>3305.61</td>
<td>1364.00</td>
<td>10.25</td>
<td>3.05</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>3666.92</td>
<td>11.14</td>
<td>5.79</td>
</tr>
<tr>
<td>RC1_6_1</td>
<td>3483.14</td>
<td>4089.00</td>
<td>2.15</td>
<td>3.06</td>
</tr>
<tr>
<td>RC1_6_2</td>
<td>3272.80</td>
<td>3680.00</td>
<td>0.66</td>
<td>2.11</td>
</tr>
<tr>
<td>RC2_6_1</td>
<td>3975.13</td>
<td>2015.00</td>
<td>5.36</td>
<td>4.11</td>
</tr>
<tr>
<td>RC2_6_2</td>
<td>3465.40</td>
<td>1167.00</td>
<td>2.86</td>
<td>2.63</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>3366.21</td>
<td>2.76</td>
<td>2.98</td>
</tr>
<tr>
<td>Total average</td>
<td></td>
<td>3090.33</td>
<td>7.53</td>
<td>4.77</td>
</tr>
</tbody>
</table>

### Table 8: Computational results of the instances with 800 customers.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Classic ALNS</th>
<th>Our ALNS</th>
<th>RCE (%)</th>
<th>QOI</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1_8_1</td>
<td>7417.07</td>
<td>7210.68</td>
<td>2.78</td>
<td>3.86</td>
</tr>
<tr>
<td>R1_8_2</td>
<td>6868.49</td>
<td>6627.29</td>
<td>3.51</td>
<td>2.55</td>
</tr>
<tr>
<td>R2_8_1</td>
<td>9113.60</td>
<td>8206.30</td>
<td>9.96</td>
<td>6.95</td>
</tr>
<tr>
<td>R2_8_2</td>
<td>7803.53</td>
<td>1767.00</td>
<td>10.36</td>
<td>3.84</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>7777.30</td>
<td>6.65</td>
<td>4.30</td>
</tr>
<tr>
<td>C1_8_1</td>
<td>9107.37</td>
<td>7177.04</td>
<td>21.20</td>
<td>6.14</td>
</tr>
<tr>
<td>C1_8_2</td>
<td>6824.46</td>
<td>6348.22</td>
<td>6.98</td>
<td>4.45</td>
</tr>
<tr>
<td>C2_8_1</td>
<td>5550.20</td>
<td>5032.88</td>
<td>9.32</td>
<td>7.81</td>
</tr>
<tr>
<td>C2_8_2</td>
<td>5484.32</td>
<td>4606.45</td>
<td>16.01</td>
<td>6.66</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>5297.88</td>
<td>13.38</td>
<td>6.26</td>
</tr>
<tr>
<td>RC1_8_1</td>
<td>6245.84</td>
<td>6155.85</td>
<td>1.44</td>
<td>3.75</td>
</tr>
<tr>
<td>RC1_8_2</td>
<td>5954.10</td>
<td>5779.15</td>
<td>2.94</td>
<td>2.48</td>
</tr>
<tr>
<td>RC2_8_1</td>
<td>6894.01</td>
<td>6575.12</td>
<td>4.63</td>
<td>6.61</td>
</tr>
<tr>
<td>RC2_8_2</td>
<td>6451.26</td>
<td>6074.73</td>
<td>5.84</td>
<td>4.59</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>6297.51</td>
<td>3.71</td>
<td>4.36</td>
</tr>
<tr>
<td>Total average</td>
<td></td>
<td>6201.36</td>
<td>7.91</td>
<td>4.97</td>
</tr>
</tbody>
</table>

Complexity
are compared with improved heuristics. The performance evaluation by the accuracy and iteration of the algorithm is shown in Table 11.

As shown in Table 11, in the same time, the number of iterations of the fast insertion method is 4.74 times than that of the traditional insertion method, which means that the fast insertion method significantly saves the judgment time of the feasibility of the new route. In the same time, the fast insertion method improves the average accuracy by 8.64% compared with the traditional insertion method. In the optimal situation, the improvement can achieve 34.93%.

### 5. Conclusion

In this paper, an adaptive large neighborhood search (ALNS) algorithm is proposed to solve large-scale instances of GVRP. In the destroy operators, a new removal heuristic applying to the characteristics of GVRP is proposed. The heuristic can quickly remove customers who bring a large amount of carbon emissions with pertinence, and these customers may be arranged more properly in future repair operators. In the repair operators, a fast insertion method, suitable for all variants of VRPTW, is developed. In the fast insertion method, the feasibility of a new route is judged by checking the constraints of partial customers after the inserted customer, instead of checking the constraints of all customers. Thus, the computational time of ALNS is greatly saved. Computational experiments were performed on Solomon benchmark with 100 customers and Homberger benchmark instances with up to 1000 customers. Given the same computational time, the proposed ALNS improves the average accuracy by 8.49% compared with the classic ALNS. In the optimal situation, the improvement can achieve 33.61%.

### Data Availability

All data included in this study are available upon request to the corresponding author.

### Conflicts of Interest

The authors declare that they have no conflicts of interest.

### Acknowledgments

This research was supported by the National Natural Science Foundation of China (71571037, 71831006, 71420107028, 71271093).
71601089, and 71620107003) and the Fundamental Research Funds for the Central Universities (N170405005).

References


[31] G. Laporte, R. Musmanno, and F. Vocaturo, "An adaptive large neighbourhood search heuristic for the capacitated arc-


