Research Article

Multiobjective Green Time-Dependent Location-Routing Problem and Algorithms

Hua-Xin Zhang¹ and Chun-Miao Zhang²

¹Department of Basic Education, Jiaxing Vocational and Technical College, Jiaxing 314036, China
²Department of Intelligent Manufacturing, Jiaxing Vocational and Technical College, Jiaxing 314036, China

Correspondence should be addressed to Chun-Miao Zhang; 1121402110@zjut.edu.cn

Received 6 February 2022; Revised 18 March 2022; Accepted 25 April 2022; Published 21 May 2022

Academic Editor: Saurabh Pratap

Copyright © 2022 Hua-Xin Zhang and Chun-Miao Zhang. 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.

To reduce the logistic cost and carbon emission and improve customer satisfaction, this study proposes a multiobjective green time-dependent location routing problem (MOGTDLRP) model in which the objectives are to minimize the distribution total cost, delivery time, and fuel consumption. This model will be solved by several hyperheuristic algorithms which include the high-level heuristics and the low-level heuristics. There are three acceptance criterions for the solution: improving and equal, all moves and accept all solutions, and dynamic acceptance criteria. Through the case, the performance of the algorithm and the influence of various factors on the solution are analyzed in this study. The experimental results show that the proposed model can effectively reduce logistic costs, carbon emissions, and vehicle travel time.

1. Introduction

Urban logistics plays an important role in the economic vitality of the cities. Green logistics is defined as planning and execution of logistics activities in a more environmentally friendly way by considering external factors such as waste, noise, energy usage, and greenhouse gas (GHG) emissions [1, 2]. At the same time, the urban freight transport is known to be responsible for congestion. The location-routing problem (LRP) is one of the most important combinational optimization problems in supply chain management and logistics system planning and conducts joint decision-making regarding the locations of arbitrary types of facilities and the routing of vehicles [3]. The LRP with environmental issues was called the green LRP (GLRP) which aims to reduce fuel consumption and vehicle exhaust emissions [4]. Koç [5] proposed a model which aims to minimize the total costs including the depot costs, vehicle and routing costs, and emissions costs. Yu [6] investigated the location-routing problem with time-dependent demands as an extension of the location-routing problem by considering the time-dependent demand characteristic. Alamatsaz [7] developed a mixed-integer programming model considering time windows for customers and drivers. There are many other researchers [8–11] who investigated the green LRP model with the objective of minimizing the total costs including depot costs, vehicle and routing costs, and emission costs. However, in addition to reducing costs, other factors such as travel time, fuel consumption, and customer satisfaction should be taken seriously. Therefore, multiobjective GLRP (MOGLRP) problems have gradually become a research hotspot since 2019. Toro [12] proposed a multiobjective model for the GLRP to minimize the operational cost and the environmental effect. Rabani [13] introduced a new variant of the MOGLRP to minimize the total travel distance and the total costs including vehicle fixed cost and CO₂ emissions. Tang [14] established a biobjective model to reduce costs and carbon emissions from the perspective of a sustainable supply chain network.

For further analysis, the papers on GLRP in Table 1 are compared with five factors: (i) year, (ii) the number of objective functions, (iii) time window, (iv) vehicle speed, and (v) algorithm.

There are 11 papers about single GLRP and 10 papers about multiobjective GLRP in Table 1. In the above models, some do not consider the speed of the vehicle, and the rest
consider the speed to be a constant. However, speed is the critical parameter in calculating fuel consumption and emissions [25]. The fuel consumption is accepted which was affected by many factors in which the vehicle speed is the most critical one, and the vehicle emission is proportional to the fuel consumption [26]. In the real road network, the vehicle speed is the time-dependent function because of the traffic flow, weather, accidents, and other factors. The model based on time-dependent road network is more instructive to the real logistics distribution. So, a three-objective model named the multiobjective green time-dependent location routing problem (MOGTDLRP, as shown in Figure 1. Its goal is to reduce the total costs (including open depot costs, vehicle costs, and distribution costs), the vehicle fuel consumption, and the delivery time.

Since the LRP is a NP-hard problem, MOGTDLRP is also a NP-hard problem. In the MOGTDLRP, vehicles travel on different types of roads at different speeds at different times, and deliver goods to customers within the time window. Since exact algorithms can only solve small-scale optimization problems in a reasonable amount of time, metaheuristics are selected to solve the large-scale problems.

These metaheuristics include genetic algorithm [27–29], the ant colony optimization [30], and A Tabu search [17]. Different from the metaheuristics, several hyperheuristics (HHs) are proposed in this study. The hyperheuristic system, attempting to find the right sequence of heuristics in a given situation rather than trying to solve a problem directly, was defined as a “heuristic selection heuristic” algorithm [32, 33]. In the HH, the domain barrier isolates the high-level heuristic (HLH), including the selection strategy of the operator and the receiving mechanism of the solution and the low-level heuristic (LLH) which contains a series of low-level operators, problem definitions, objective functions, and other information. The role of the selection strategy is to monitor the performance information of the LLH to select a good and suitable operator, and the receiving mechanism determines whether to replace the parent solution according to the quality of the child solution and controls the search direction of the algorithm and the convergence speed. In this study, we design HHs with heuristic algorithms (artificial bee colony algorithm (ABC), ant colony optimization (ACO), and Tabu search algorithm (TS)) as HLH to solve MOGTDLRP. And we design three acceptance criterions

<table>
<thead>
<tr>
<th>References</th>
<th>Year</th>
<th>Objective</th>
<th>Time window</th>
<th>Speed</th>
<th>Algorithm</th>
<th>References</th>
<th>Year</th>
<th>Objective</th>
<th>Time window</th>
<th>Speed</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>[8]</td>
<td>2020</td>
<td>Single</td>
<td></td>
<td></td>
<td>Heuristic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: The diagram of the MOGTDLRP.
Some assumptions need to be satisfied: (1) each customer must be served only once, (2) each vehicle must return to the original depot, (3) every vehicle shall not be overloaded, and (4) the load of each depot must not exceed its capacity. Based on the above assumptions and definitions, the model can be represented as follows:

$$
\min f_1 = \sum_{j \in D} FD_j y_j + \sum_{i \in V} \sum_{j \in V} \sum_{k \in H} FV_i x_{ijh} + \sum_{i \in V} \sum_{j \in V} \sum_{k \in H} F_{ijh} x_{ijh}
$$

$$
\min f_2 = \sum_{i \in V} \sum_{j \in V} \sum_{k \in H} \left( x_{ijh} ^{\text{max}} TT_{ijh} ^{\text{min}} + \max(e_i - AT_j, 0) + ST_j \right).
$$

$$
\min f_3 = \sum_{i \in V} \sum_{j \in V} \sum_{h \in H} F_{ijh} x_{ijh}.
$$

The following constraints must be satisfied:

$$
\sum_{i \in V} x_{ijh} = 1, \quad \forall j \in C,
$$

$$
\sum_{h \in H} x_{ijh} = \sum_{h \in H} x_{ijh}, \quad \forall j \in C,
$$

$$
\sum_{h \in H} x_{ijh} = 1, \quad \forall i \in C,
$$

$$
x_{ijh} + \sum_{k \in H} x_{ijk} \leq 1, \quad i \in V, j \in V, i \neq j, h \in H,
$$

$$
\sum_{h \in H} x_{ijh} \leq z_{ij}, \quad \forall i \in C, j \in D,
$$

$$
\sum_{h \in H} x_{ijh} \leq z_{ij}, \quad \forall i \in C, j \in D,
$$

Table 2: Parameters and values.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
<th>Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>Set of the candidate depots</td>
<td>$V_j(t)$</td>
<td>Travel speed of the vehicle $h$</td>
</tr>
<tr>
<td>$C$</td>
<td>Set of the customers</td>
<td>$K_{ijh}$</td>
<td>Load of vehicle $h$ leaving $i$ and traveling to $j$</td>
</tr>
<tr>
<td>$q_i$</td>
<td>The demands of the customer $i$</td>
<td>$ST_j$</td>
<td>Service time of customer $j$</td>
</tr>
<tr>
<td>$ST_j$</td>
<td>Service time of customer $j$</td>
<td>$AT_{jh}$</td>
<td>The time when vehicle $h$ arrives at node $j$</td>
</tr>
<tr>
<td>$x_{ijh}$</td>
<td>Coordinate of the node $i$</td>
<td>$TT_{ij}$</td>
<td>Travel time between nodes $i$ and $j$</td>
</tr>
<tr>
<td>$[a_i, e_i]$</td>
<td>Time window of the node $i$</td>
<td>$1$</td>
<td>1 if depot $r$ is open, and 0 otherwise</td>
</tr>
<tr>
<td>$P_j$</td>
<td>Capacity of the depot $j$</td>
<td>$z_{ij}$</td>
<td>1 if customer $i$ is served by depot $m$, and 0 otherwise</td>
</tr>
<tr>
<td>$FD_j$</td>
<td>Cost of the depot $j$</td>
<td>$FV_{ij}$</td>
<td>Cost of the vehicle $h$</td>
</tr>
<tr>
<td>$Q_h$</td>
<td>Capacity of the vehicle $h$</td>
<td>$x_{ij}$</td>
<td>1 if customer $i$ is served by depot $m$, and 0 otherwise</td>
</tr>
<tr>
<td>$d_{ij}$</td>
<td>Distance between the two nodes $i$ and $j$</td>
<td>$F_{ijh}$</td>
<td>Fuel consumption of the vehicle $h$ from node $i$ to node $j$</td>
</tr>
</tbody>
</table>

(ACs) for the solution, Improving and Equal (IE), All Moves (AM) and Accept all solutions, and Dynamic Acceptance Criteria (DA) and a series of low-level operators that conform.

In summary, the main contributions and innovations of this study are as follows:

1. The effects of multiple objectives, time-dependent network, and fuel consumption are considered, and the MOGTDLRP mathematical model is constructed.
2. Three HLHs and three ACs combine to form nine HH to analyze the performance of the algorithms.
3. We propose a set of benchmarks of the MOGTDLRP and analyze the simulation results.

The construction of this study is as follows. Section 2 is about the MOGTDLRP model. Section 3 is an introduction to the HHs. Section 4 is the results and analysis. Section 5 is the conclusion and outlook.

2. The Model of MOGTDLRP

The model of MOGTDLRP with three optimization objectives will be introduced in this section. The MOGTDLRP can be defined on an asymmetric directed graph $G = (V, E)$, where $V$ is composed of the set of the candidate depots $D$ ($D = \{1, 2, \ldots, M\}$) and the set of customers $C$ ($C = \{1, 2, \ldots, N\}$) and $E$ is a set of edges ($E = \{(i, j); i, j \in V, i \neq j\}$). The demand $q_i$, service time $ST_j$, coordinate $(x_i, y_i)$, and hard time window $[a_i, e_i]$ of customer $i$ are known. The capacity $Q_h$ and rent cost $C_{ijh}$ are known. Each vehicle $h$ in the homogeneous fleet has the same capacity $Q_h$ and rent cost $C_{ijh}$. The symbol $d_{ij}$ is the distance between the two nodes $i$ and $j$ ($i \neq j \in V$). The delivery time of a vehicle depends on the distance, where the speed changes according to the departure time and the arc being traversed. $V_h(t)$ is the travel speed of the vehicle $h$ traversing different arcs. $TT_{ij}$ is the travel time between nodes $i$ and $j$. $K_{ijh}$ is the load vehicle $h$ leaving $i$ and traveling to $j$. $ST_j$ is the service time of customer $j$. $AT_{jh}$ is the time when vehicle $h$ arrives at node $j$. The decision variable $y_{ijh}$ equals 1 if depot $r$ is open and equals 0 otherwise. $z_{ij}$ equals 1 if customer $i$ is served by depot $m$ and equals 0 otherwise.
\[
\sum_{h \in H} x_{ijh} + z_{ik} + \sum_{m \in D, m \neq k} z_{jm} \leq 2, \quad \forall i, j \in C, k \in D, i \neq j,
\]

(10)

\[
\sum_{i \in D} \sum_{j \in C} K_{ijh} = \sum_{n \in C} \sum_{j \in V} q_j x_{ijh}, \quad \forall h \in H,
\]

(11)

\[
\sum_{i \in C} \sum_{j \in D} K_{ijh} = 0, \quad \forall h \in H,
\]

(12)

\[
\sum_{q_j x_{ijh}} z_{ik} \leq P_k y_k, \quad \forall k \in D,
\]

(13)

\[
\sum_{i \in C} \sum_{j \in D} (K_{ijh} - q_j) x_{ijh} = \sum_{i \in V} \sum_{h \in H} K_{ijh} x_{ijh}, \quad \forall j \in C,
\]

(14)

\[
K_{ijh} \leq q_h x_{ijh}, \quad \forall i, j \in V, i \neq j, h \in H,
\]

(15)

\[
K_{ijh} \geq q_j x_{ijh}, \quad \forall i \in V, j \in C, h \in H,
\]

(16)

\[
AT_{jh} = x_{ijh} \cdot \left( \max \{e_{ij}, AT_{jh} \} + ST_i + TT_{ijh} \right), \quad \forall i, j \in V, h \in H,
\]

(17)

\[
0 \leq AT_{jh} \leq l_j, \quad \forall j \in C, h \in H,
\]

(18)

\[
x_{ijh} \in \{0, 1\}, \quad \forall i, j \in V, h \in H,
\]

(19)

\[
y_j \in \{0, 1\}, \quad \forall j \in D,
\]

(20)

\[
z_{ij} \in \{0, 1\}, \quad \forall i \in C, \forall j \in D.
\]

(21)

The description and explanation are as follows. Equation (1) is the costs’ objective. Equation (2) is travel time objective. Equation (3) is the third objective which is the total fuel consumption. Constraints (4) and (5) make sure that each customer is served exactly once. Constraints (6) and (7) impose that each customer is served by one depot and one vehicle, respectively. Constraints (8)–(10) forbid illegal routes that the corresponding vehicles do not return to the depot. Constraints (11) and (12) make sure that the demand of each customer is satisfied. Constraint (13) guarantees that the load of each selected depot must be less than its capacity. Constraint (14) is the dynamic equilibrium of the load of each vehicle after visiting customer j. Constraints (15) and (16) guarantee that overloading is not allowed. Constraints (17) and (18) are the bounds on the arrival time at each node: a vehicle must arrive at a customer before the closing time windows. Finally, the last three constraints are decision variables.

3. Algorithms

Metaheuristics are wide-ranging. However, choosing the best algorithm and configuring the parameters and operator to solve the problem is very difficult [34]. Numerical experiments have shown that it is impossible to develop a single metaheuristics algorithm that is always efficient for a diverse set of optimization problems. Hyperheuristic algorithm (HH) is a super heuristic algorithm, which provides a high-level heuristic (HLH) method. It can solve various combinatorial optimization problems by managing or manipulating a series of low-level heuristics (LLH). HHs can help researchers to reduce the inherent time and effort required to set up a new domain, as it is a difficult task for testers without deep prior domain-specific knowledge [35].

There are two main tasks in the design of HH: one is the high-level selection strategy and the other is the design of problem domain operator. Using the HH to solve the MOGTDDL problem can solve the selection problem of the local optimization operators. If iterating all operators, a lot of time will be wasted. The HH can decide whether to enable the operator through a reasonable selection mechanism according to the history of the operator, which not only saves time but also improves the quality of the solution.

In the HH, there are two different target strategies: operator selection strategy and reconciliation receiving mechanism. In addition, there is an information transmitter between the HLH and the LLH, which transmit information, such as the information of the selection operator, the judgment information of the receiving mechanism, the running time and frequency of the operators, and the number of consecutive unimproved times of the current solution. Therefore, the HLH plays a vital role in improving the performance of the HH. In this study, we have designed nine operators and three HLHs including Tabu search algorithm (TS), ant colony optimization (ACO), and artificial bee colony algorithm (ABC). Next, we will introduce HH according to the following steps: (1) coding method and initial population generation; (2) HLHs; (3) acceptance criteria; (4) the LLHs pool of operators, \( \xi \).

3.1. Coding Method and Initial Population Generation. In the MOGTDDL problem domain, a complete solution is a collection of all routes \( R = [r_1, r_2, \ldots, r_m] \). Each route \( r_i \) has the same structure: the first and the end node is the opened depot, and the middle part is the customer number. For example, the route \( \{D_2-C_1-C_3-C_{10}-D_2\} \) means that a certain vehicle departs from the depot \( D_2 \) to deliver goods to customers \( C_1, C_3, \) and \( C_{10} \) in order, and then, returns to \( D_2 \). The initial solution is critical to the quality of the final solution. In the traditional LR, when generating an initial route, only the capacity constraints of the depots and vehicles need to be satisfied, and the optimization objective is the distribution distance. However, in the MOGTDDL, it is also necessary to meet the time window of the depots and customers. In order to produce high-quality initial solutions, it is necessary to design more efficient algorithms. There are two classical algorithms for vehicle-routing problem (VRP) with time windows for reference: Solomon’s Insertion algorithm [36] and IMPACT algorithm [37]. However, these two algorithms need to set a lot of parameters. Zhang [8], in her paper, designed one algorithm named the insertion method based on travel time (IMTT) to generate the initial solution. This method has good solution quality and does not need to set many parameters. Therefore, this study used...
the IMTT to generate the initial solutions. Its steps are as follows:

First step: select the opened depots according to the barycentric method
Second step: randomly select a customer node to generate a new route
Third step: calculate the total travel time of this route
Fourth step: calculate the impact values of the remaining customers inserted into the route
Fifth step: insert new customers into the route according to the impact values until one or more constraints are not satisfied

Figure 2 shows the generation process of the initial solution. Depot0 is the selected depot; Insi represents the insertable position; Ci is the customer in the route.

3.2. High-Level Heuristic

3.2.1. The Artificial Bee Colony (ABC) Algorithm. The ABC includes three elements: foods, employed bees, and unemployed bees which have the onlooker bees and the scouts make up. The foods are the solutions in the MOGTDLRP. Employed bees are responsible for finding foods and carrying information about the foods. Onlooker bees follow the employed bees with a certain probability to select the foods. Scouts are responsible for searching for foods randomly to enhance the algorithm’s ability to jump out of local optimal solutions. In the initialization phase of the ABC algorithm, SN solutions, that are the location of the food sources, are generated by IMTT. Then, these initial solutions are evaluated to obtain their fitness values.

In the ABCCHH, the food sources are the bottom operators, and the optimization process is the process of the bee colony searching for foods. The high-level strategy idea of the ABC is as follows.

First, 20% of the bees in the colony are set as scout bees whose roles are to find excellent sources of food. Each scout bee calculates 20 times at each operator. The count table score (i) of the operator i will be added by 1 point when the fitness improves. The operators are sorted in the descending order according to their scores when excellent food has been found, all individuals become employed bees to enjoy the food, and individual is updated by the following equation:

\[ X_{i}^{t+1} = X_{i}^{t} \cdot (p_{i} \cdot S_{a} + \text{rand}(S - S_{a} - S_{\text{best}}) + S_{\text{best}}), \]

\[ i = 1, 2, \ldots, SN, a, b = 1, \ldots, |S|, a \neq b, \]

where \( i \) represents the \( i_{th} \) individual in the population, \( t \) is the number of iterations, \( S_{a} \) represents the \( a \)th operator in the operator set, \( SC_{a} \) is the score ranking of operator \( S_{a} \), and \( P_{i} \) means that the \( S_{a} \) operator is selected with the probability of \( P_{i} \). When the \( t \) generation solution \( X_{t} \) is better than the \( t-1 \) generation solution, \( P_{i} = 1 \); otherwise, it is calculated according to equation. \( \text{rand}(S - S_{a} - S_{\text{best}}) \) means that an operator is randomly selected after the operator set \( S \) eliminates the \( S_{a} \) and \( S_{\text{best}} \) operators, and \( S_{\text{best}} \) is the operator with the highest score in the \( S \) set.

3.2.2. Related Definitions

Definition 1. Status minus.

For the problem of real number coding, let the state of bee \( i \) be \( x_{id} \). According to equation (23), bee \( i \) randomly approaches the position \( x_{kd} \) of another bee for information transmission:

\[ x_{id} = x_{id} + \phi_{d}(x_{id} - x_{kd}), \quad i = 1, 2, \ldots, SN, d = 1, 2, \ldots, D, \]

which means an operation bit \( d \) is randomly selected, and the element corresponding to the \( d \) bit of \( x_{id} \) and \( x_{kd} \) forms a field \([a b]\), where \( a \) is a smaller value and \( b \) is a larger value. Then, \( \lambda(x_{id}, x_{kd}) \) is the reverse order of \( x_{id} \) from \( a \) to \( b \).

For example, let \( x_{id} = [1 2 3 4 5 6] \), \( x_{kd} = [3 5 2 1 6 4] \), and random number \( \beta = 4 \); then, \( x_{id}(4) = 4, x_{kd}(4) = 1, \) and \( [a b] = [1 4] \). According to the definition, \( x_{id} - x_{kd} \) means that the 1–4 bits of \( x_{id} \) are arranged in the reverse order to produce a new solution \( x_{id} = [4 3 2 1 5 6] \).

Definition 2. State number multiplication.

\( C \times x_{id} \) represents the number multiplication of states that means taking the first.

\( [C \times D] \) elements of \( x_{id} \), and the symbol \([\]\)indicates rounding up.

For example, \( x_{id} = [1 2 3 4 5 6] \) and \( C = 0.5 \); then, \( [C \times D] = 3 \) and \( x_{id} = [1 2 3] \).

Definition 3. Status sum.

Status sum are 2-opt switching operations.

For example, let \( x_{id} = [1 2 3 4 5 6] \) and \( XO = [2 3 1 4] \). First, we convert XO to XO = (2, 3) (1, 4)). Then, \( x_{id} + XO \) indicates that the second bit and the third
3.2.3. The Ant Colony Optimization (ACO) Algorithm. The high-level strategy is based on ACO. There are \( m \) ants distributed among the bottom operators. Each ant determines the next vertex according to the pheromone in each route, and the quality of the operator is evaluated by visibility function \( \eta_i \):
\[
\eta_i(t) = \gamma \eta_i(t-1) + \sum_{k} \frac{I_{kj}}{T_{kj}(t)},
\]
where \( I_{kj} \) is the improved value (which can be negative) after ant \( k \) applies the bottom operator \( j \) in generation \( t \), \( T_{kj}(t) \) is the CPU running time occupied by the calculation, and \( \gamma \) is a value between 0 and 1.

The equation of \( r_{ij}(t) \), the pheromone on each arc, is as follows:
\[
r_{ij}(t) = (1-\rho)r_{ij}(t-1) + \frac{\sum_{k} P_{kj}(t)}{\sum_{k} I_{kj}(t)}.
\]
where \( \rho \) (\( 0 < \rho < 1 \)) is the evaporation coefficient of pheromone, \( (1-\rho) \) represents the persistence coefficient of pheromone, \( P_{kj}(t) \) represents the order in which ant \( k \) selects the bottom operator, \( \#_{ij}(P_{kj}(t)) \) represents the number of times the arc \((i,j)\) appears in the route of ant \( k \), \( I(P_{kj}(t)) \) is the amount of improvement of ant \( k \) in the route, and \( T(P_{kj}(t)) \) is the CPU time.

The ant colony selects operators based on pheromone and visibility; the equation is as follows:
\[
V_{ij}(t) = \alpha \eta_{i}(t) + \beta r_{ij}(t).
\]

There is a variable PV to improve the selection probability of some operators with poor performance:
\[
\begin{align*}
PV_{ij}(t) &= \max\{V_{ij}(t), Q\sigma V_{ij}(t)\}, \\
Q &= \frac{\max\{0, V_i h(t) + \epsilon\}}{10n},
\end{align*}
\]
where \( H \) is the set of bottom operators, \( n = |H| \) is the number of the bottom operators, and \( \epsilon \) and \( \sigma \) can ensure that the underlying heuristic operator with poor performance still has a small but nonzero probability of being chosen. If arc \((i,j)\) is an illegal arc, \( PV_{ij}(t) = 0 \). If the performance of all bottom operators is not good, \( q = 0 \), and set all PV values to 1 to ensure that all bottom operators are selected with equal probability. The probability that the arc \((i,j)\) is selected can be expressed as
\[
\text{probability}_{ij}(t) = \frac{PV_{ij}(t)}{\sum_{h \in H} PV_{ih}(t)}.
\]

3.2.4. Tabu Search (TS) Algorithm. The high-level strategy idea of the TS is as follows. Score evaluates the performance of operators. Then, select an operator with a larger score to improve the current solution in the iterative process. Each operator has the same initial score. When the operator improves the current solution, the score of this operator will add 1; otherwise, it will subtract 1. Detailed process is as follows:

First, an initial score \( r_k = 0 \) is set for each operator \( k \). The score interval of each operator is \([r_{min}, r_{max}]\); \( r_{min} \) and \( r_{max} \) represent the lowest and highest scores, respectively. The operator \( k \) that is not in the Tabu list and has the highest score will be selected. If it can improve the objective function, the score of operator \( k \) will add 1. Otherwise, the score will subtract 1 and be put into the Tabu list with fixed Tabu length. Operators in the Tabu list are exempted according to the first in first-out principle.

3.3. Acceptance Criteria. The acceptance criterion is the process of whether the new solution is obtained after processing by the operator. If the new solution always unconditionally replaces the parent solution, the optimization time will be too long. However, the HH will fall into a local search if it just saves the improved solution and omits the inferior solution. Other acceptance criteria include the accepting improved solutions completely and accepting inferior solutions in a certain proportion.

In this study, three acceptance criteria are proposed:

1. Improving and equal (IE): accept all improved solutions and reject nonimproved solutions.
2. All moves (AM): accept all solutions.
3. Dynamic acceptance criteria (DA): if the current optimal solution is not updated too many times, which means that the solution falls into local search, the acceptance probability of inferior solution should be increased to ensure global search. The acceptance probability is calculated as follows:
\[
p = \sqrt{f_{t-1} - f_t + \frac{M_i}{(f_t + f_{best})/2}},
\]
where \( f_{t-1} \) is the parent solution, \( f_t \) is the child solution, \( f_{best} \) is the optimal solution of generation \( t \), \( M_i \) is a constant, and \( M_i \) is the number of times that \( f_{best} \) has not been updated.

3.4. Pool of LLH. This section introduces the nine LLH operators including four operators operating within one route, three operators operating between two routes, and two operators operating the selection of the depots.
(1) 2-opt inside one route

(2) Or-opt inside one route

(3) Rev inside one route

(4) Move inside one route

(5) Or-opt between two routes

(6) Interchange between two routes

(7) Crossover between two routes

(8) Depot replace

(9) Depot interchange
### 3.5. Pseudocode of the HH

Table 3 lists the pseudocode of the HH, and Figure 3 shows the flowchart.

#### 4. Simulation Experiment and Analysis

All programs are coded in MATLAB R2018a and executed on a computer with an Intel (R) Core (TM) i5-5200U CPU @2.20GHz, 4 GB of RAM, and Windows 7 operating system.

#### 4.1. Benchmarks

The experimental data come from [8]. Table 4 shows the detailed depot data, and Table 5 includes the time windows of all depots. In the tables, “Ben” is benchmark’s name, “Para” is the abbreviation of parameter, “cap” is the capacity, “coor” is the coordinate, “costs” is the fixed open costs, and “TW” notes the time window. Table 6 shows the data of the vehicles. The data about customers come from the Solomon benchmarks.

#### 4.2. The Time-Dependent Speed Function

Ioannou [37] first proposed the vehicle travel time-dependent speed function in his study about vehicle-routing problem and calculated the travel time by the length of the road segment and the travel speed, that is, the travel speed is a time-dependent function. This time-dependent speed function can avoid jumps in travel time and ensure the FIFO characteristics of the time-varying network; that is, the vehicle that departs first arrives at the destination first. For the road network, it can be considered to conform to the FIFO characteristic without overtaking.

In this study, to simulate urban road conditions and rush hours, we divide the roads into five types and divide the total service time into four equal time periods. Table 7 shows the values.

---

**Table 3: Pseudocode of the HH.**

```
(1) //Initialization
(2) Set the parameters
(3) Initialize Solution (Pop)
(4) Current Solution = Pop (r)
(5) Best Solution = Current Solution
(6) Fitness (Best Solution) = Fitness (Current Solution)
(7) //Main loop
(8) while t < ITER do
(9)   //High-level Selection Strategy
(10)  operator = Select (ξ)
(11)  //low-level heuristics
(12)  [Child Solution] = Implement(Current Solution, operator)
(13)  //High-level Acceptance Criterion
(14)  if Fitness (Child Solution) < Fitness (Current Solution) then
(15)     Current Solution = Child Solution
(16)  else
(17)     Current Solution = Accept or Not (Child Solution, Current Solution)
(18)  end if
(19)  //Save the global best solution
(20)  if Fitness (Best Solution) > Fitness (Child Solution) then
(21)    Best Solution = Child Solution
(22)  end if
(23)  Update related data
(24) end while
```

---

**Figure 3: The flowchart of the HH.**

The road type is obtained by equation (30), where \( \text{Grade} (i, j) \) is the type of road between nodes \( i \) and \( j \) and \( \text{Mod} (a, b) \) is the remainder of the number \( a \) divided by the number \( b \):
\[ \text{Mod} (a, b) + 1. \]  

4.3. Parameter Settings. Population size is 100, the algorithm iteration number is 200, ant path length \( LP = 11 \), weight parameter \( \alpha = (0.6, 0.7, 0.8), \) \( \beta = (0.6, 0.7, 0.8) \), and \( \gamma = (0.6, 0.7, 0.8) \), and enhancement coefficient \( \varepsilon = 0.001 \) and \( \sigma = 1.001 \). Tabu search algorithm parameter maximum score is \( \text{rankingMax} = 5 \), minimum score is \( \text{rankingMin} = 0 \), and Tabu length is \( \text{tabuLength} = 4 \).

4.4. Comparative Analysis of the Nine Combined HHs. Three high-level algorithms (ABC, ACO, and TS) and three acceptance criterion (AM, IE, and DA) can form nine HHs named ABC + AM, ABC + IE, ABC + DA, ACO + AM, ACO + IE, ACO + DA, TS + AM, TS + IE, and TS + DA. These nine HHs solve the benchmarks of the MOGTDLRP.

The results can reflect the performance of the algorithm. The relevant data are as follows:

Objective function: \( f_1 \), minimum total cost.

Instances: C1 (C101–C109).

Comparison parameters: deviation percentage \( RD \) obtained as

\[ RD = \frac{f_A - f_{\text{best}}}{f_{\text{best}}} \times 100, \]

where \( f_A \) is the best result obtained by algorithm A and \( f_{\text{best}} \) is the average of the results obtained by the nine HHs.

The flow of the nine HHs:

Step 1: generating the initial population by the IMTT algorithm and calculating the objective functions to get the initial father solutions S11, S21, and S31.

Step 2: setting relevant algorithm parameters.

Step 3: selecting the operator of the LLH according to HLH and calculating the objective functions to get the child solutions S12, S22, and S32.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline
\hline
C1 & cap & 950 & 800 & 1010 & 970 & 920 & 990 & 1030 & 930 & 870 & 890 \\
& x & 40 & 64 & 35 & 44 & 29 & 18 & 69 & 28 & 43 & 42 \\
& y & 50 & 13 & 79 & 57 & 40 & 82 & 93 & 8 & 63 & 17 \\
& costs & 40000 & 45000 & 42000 & 41000 & 48000 & 50000 & 38000 & 49000 & 47000 & 46000 \\
\hline
C2 & cap & 960 & 750 & 910 & 820 & 720 & 790 & 800 & 790 & 890 & 900 \\
& x & 35 & 10 & 52 & 46 & 81 & 11 & 94 & 78 & 88 & 72 \\
& y & 50 & 54 & 56 & 60 & 24 & 59 & 40 & 77 & 66 & 49 \\
& costs & 20000 & 19000 & 22000 & 21000 & 18000 & 23000 & 24000 & 17000 & 25000 & 24000 \\
\hline
R1 & cap & 1020 & 810 & 720 & 790 & 890 & 1070 & 740 & 700 & 1100 & 790 \\
& x & 35 & 92 & 82 & 25 & 64 & 100 & 10 & 3 & 8 & 64 \\
& y & 35 & 77 & 17 & 82 & 17 & 87 & 72 & 51 & 60 & 60 \\
& costs & 85000 & 94000 & 94000 & 89000 & 100000 & 92000 & 97000 & 87000 & 99000 & 96000 \\
\hline
R2 & cap & 1050 & 900 & 1090 & 850 & 790 & 940 & 970 & 1180 & 900 & 1020 \\
& x & 40 & 15 & 40 & 24 & 50 & 62 & 79 & 10 & 80 & 87 \\
& y & 50 & 52 & 3 & 93 & 76 & 60 & 100 & 95 & 11 & 75 \\
& costs & 18000 & 19000 & 17000 & 21000 & 26000 & 24000 & 23000 & 19000 & 24000 & 25000 \\
\hline
RC1 & cap & 1300 & 1200 & 900 & 800 & 1080 & 780 & 1090 & 1240 & 900 & 1100 \\
& x & 40 & 86 & 23 & 55 & 28 & 68 & 72 & 34 & 26 & 61 \\
& y & 50 & 37 & 94 & 100 & 92 & 52 & 19 & 61 & 88 & 44 \\
& costs & 86000 & 91000 & 87000 & 99000 & 96000 & 100000 & 85000 & 94000 & 93000 & 97000 \\
\hline
\end{tabular}
\caption{Data of the depots.}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
Ben & TW \\
\hline
C1 & 0.1236 \\
C2 & 0.3390 \\
R1 & 0.2300 \\
\hline
\end{tabular}
\caption{Time windows of the depots.}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
Ben & TW & Ben & TW \\
\hline
C1 & 0.1236 & R2 & 0.1000 \\
C2 & 0.3390 & RC1 & 0.2400 \\
R1 & 0.2300 & RC2 & 0.9600 \\
\hline
\end{tabular}
\caption{The time-dependent speed function.}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
Ben & C1 & C2 & R1 & R2 & RC1 & RC2 \\
\hline
Capacity & 200 & 700 & 200 & 1000 & 200 & 1000 \\
Vehicle & 800 & 2700 & 500 & 2500 & 450 & 2500 \\
\hline
\end{tabular}
\caption{The data of the vehicles.}
\end{table}
Step 4: determining whether to accept the current child solutions according to the acceptance criteria

Step 5: updating related parameters

Step 6: if the algorithm termination condition is satisfied, the algorithm ends; otherwise, go to step 3

The results of C101–C109 obtained by nine HHs are in Table 8.

The maximum value of RD in Table 8 is 1.95, which indicates that the solutions obtained by these nine algorithms are not very different. We continue to analyze the performance of the algorithm with Boda algorithm. Each HH has 1 to 9 scores according to the RD value. The algorithm with the smallest RD gets 9 points, the algorithm with the second-smallest RD gets 8 points, and so on. Figure 4 shows the score of the nine HHs. The abscissas 1–9 in the figure correspond to 9 HHs. Figure 4 is a pie chart of the number of optimal solutions obtained by the 9 combination algorithms on C1 examples; that is, the number of times that RD is equal to 0.

Through the data of Table 8 and Figures 4 and 5, it is known that the ABC + DA combination algorithm scores the highest with 80 points, followed by ACO + DA with an equal score of 69, and the third place is TS + DA with a score of 67. The pie chart shows that the number of times ABC + DA algorithm obtains the optimal solution accounts for 47.06%; ACO + DA accounted for 17.65%; TS + DA accounted for 23.53%; ABC + IE and ACO + IE accounted for 5.88%, respectively, and other combinations do not get the optimal solution. Under ABC high-level strategy, the total score is 30 + 48 + 80 = 158; ACO senior strategy score is 27 + 45 + 69 = 141 points; TS high-level strategy score is 34 + 59 + 67 = 160 points. The score of the AM acceptance criteria is 30 + 27 + 34 + 91, IE is 48 + 45 + 59 = 152, and DA is 80 + 69 + 67 = 16 points. The scores of the three strategies have little difference, but the difference of the solution acceptance criteria is large. AE accepts all solutions, which is easy to cause too divergent search range and difficult to find the optimal solution. IE only accepts noninferior solutions, which is easy to fall into the trap of local optimization. DA accepts all noninferior solutions and conditionally selects inferior solutions. This method gets a higher score because of which it takes into account the advantages of AE and IE and avoid their shortcomings. In general, the ABC + DA is the best combined HH algorithm in the C1 case.

4.5. Analysis of the Single Objective Optimization Problem. MOGTDLTP is a multiobjective model, which includes three objectives: total cost, travel time, and fuel consumption. Different optimization objectives lead to different distribution routes. In this section, the difference of the solutions of the different single objective optimization problem is discussed.

Instances: RC2 (RC201–RC208).

Algorithm: ABC-DA
Single objective: \( f_1 \), minimum total cost, \( f_2 \), minimum delivery time, and \( f_3 \), minimum fuel consumption.

The results are in Table 9.

In Table 9, OBJ is the objective, TC is total cost, TI is travel time, and FC is fuel consumption; RD is the difference percentage calculated by

\[
RD_1 = \frac{TC - TC_{\text{min}}}{TC_{\text{min}}} \times 100,
\]

\[
RD_2 = \frac{TI - TI_{\text{min}}}{TI_{\text{min}}} \times 100,
\]

\[
RD_3 = \frac{FC - FC_{\text{min}}}{FC_{\text{min}}} \times 100.
\]

The results show that the optimal solutions are different under different optimization objectives. Compared with the \( f_2 \) and \( f_3 \) objectives, the total cost obtained with the \( f_1 \) objective is the minimum, but the delivery time and fuel consumption are not the minimum. The values of RD indicate the deviation percentage between the current value and the minimum value. The larger the value, the greater the deviation. When taking travel time as the optimization objective, the RD values get the maximum seven times, indicating that when taking travel time as the optimization objective, although the total travel time gets the optimization, the total cost and fuel consumption are high. Therefore, it is not appropriate to take travel time as a separate optimization objective. When \( f_1 \), the total cost is the optimization objective; the average value of RD is 6.62, and when \( f_3 \), the fuel consumption is the optimization objective; the average value of RD is 5.30. Therefore, the solution obtained under the \( f_3 \) objective is better.

4.6. Comparison between Single Objective and Multiobjective. Instances: RC201 and RC208

Optimization objectives: \( f_1 \), minimum total cost, \( f_2 \), minimum travel time, \( f_3 \), minimum fuel consumption, and MO, multiobjective.

Algorithm: ABC + DA

Results are shown in Table 10.
In Table 10, MO-TC is the solution with the minimum total cost, MO-TI is the solution with the minimum delivery time, and MO-FC is the solution with the minimum fuel consumption in the Pareto solution set.

In the single objective function of RC201, the minimum total cost is 180116, the minimum travel time is 2335.3, and the minimum fuel consumption is 1548.9. In the multi-objective function of RC201, the minimum total cost is 2400.
180155 > 180116, the deviation is $RD = 0.02$, the minimum travel time is $2332 < 2335.3$, and the deviation is $RD = -0.14$; the minimum fuel consumption is $1641 > 1548.9$, and the deviation is $RD = 5.95$. In the single objective function of RC208, the minimum total cost is $179485$, the smallest travel time is $1604.1$, and the smallest fuel consumption is $945.14$.

In the multiobjective function of RC208, the minimum total cost is $179439 < 179485$, the deviation is $RD = 0.02$, the minimum travel time is $1599 < 1604.1$, and the deviation $RD = -0.32$; the minimum fuel consumption is $915 < 945.14$, and the deviation $RD = -3.2$. Four of the six $RD$ values are less than 0, indicating that the combination of cost, time, and fuel consumption under multiobjective function is better than that under single objective function. Figure 6 is the curves of the combined values of RC201 and RC208, respectively. In the figure, the solid lines are the combined curve obtained by single objective and the dotted lines are the combined curve under multiobjective. The dotted lines are mostly below the solid lines, which also indicate that the combined values obtained by the multiobjective function are better.

4.7. Results and Analysis of All Benchmarks. In this section, we use ABC+DA algorithm to solve all benchmarks and analyze the results. The number of iterations is set to 200. All results are in Table 11 in which “MC” is the minimum total cost, “MT” is the minimum travel time, and “MF” is the minimum fuel consumption. “AC,” “AF,” and “AT,” respectively, represent the average cost, average fuel consumption, and average time of the whole Pareto solution set. “AV” means the average number of activated vehicles.

The data in Table 11 indicate the following:

(1) In general, due to the high quality of solutions, the number of solutions of every benchmark is not a huge amount.

(2) Because of the cost of the depots, the values of MCs are too large which indicate that the ratio of depots’ cost to logistic cost is greater than the ratio of fuel consumption and travel time cost. However, as the number of customers increases, the fuel consumption and travel time cost increase, leading the change in selection depots.
(3) More vehicles need to be enabled to participate in distribution to meet customer time window needs. Taking the RC201 case as an example, the total demand of all customer is 1724 and the capacity of vehicles is 1000. Theoretically, only two vehicles need to be enabled, but in the MOGTDLRP, three vehicles need to be enabled. Due to the use of large capacity vehicles, the number of multiobjective optimal solutions of C2, R2, and RC2 is more than that of corresponding C1, R1, and RC1. The average number of Pareto optimal solutions for C1 is 5.78, while C2 is 10.75, which is about twice that of C1. The average number of Pareto optimal solutions for R1 is 6.42; while R2 is 6.72, which is greater than 6.42. The average number of Pareto optimal solutions for RC1 is 5.75; while RC2 is 7.5, which is greater than 5.75. From a practical point of view, logistic companies want to use fewer vehicles, so when designing the algorithm, the number of vehicles is the priority condition for the solution. On the contrary, using fleets of different capacities is also an effective way for logistic networks to achieve a low-carbon economy.

(4) Figures 7(a)–7(c) are the two-dimensional projections of C201 total cost and travel time, travel time and fuel consumption, and total cost and fuel consumption. Figure 7(d) is the three-dimensional diagrams of the solutions of C201. The relationship between cost and fuel consumption is linear because the fuel consumption is part of the total cost. Figures 7(a), 7(b), and 7(d) solution distribution is relatively uniform, indicating that the quality of the solution is higher.

5. Conclusion and Outlook

In this study, we proposed the MOGTDLRP model with three objectives and a time-dependent speed function with dividing one day into four periods and definition five road types. Fifty six instances were obtained based on the Solomon benchmarks by referring to the relevant literature. In every benchmark, there are 100 customers whose coordinates, requirements, and time windows are known and 10 depots whose coordinates, capacity, and time window are also known. The benchmarks are solved by nine efficient HHs with the high-quality initial solution obtained by IMTT. The results indicated that the MOGTDLRP could effectively reduce the fuel consumption and the total costs.

According to the analysis of experimental results, this study can get the following conclusions and suggestions:

(1) The proposed HHs have high accuracy in solving the MOGTDLRP model and can obtain high-quality solution in reasonable computing time. The MOGTDLRP model can effectively reduce logistic costs, fuel consumption, and travel times.

(2) Factors such as customer distribution and customer time windows should be the same issues that logistic companies need to pay attention to as depot costs.

(3) Heterogeneous fleet can reduce the distribution cost.

(4) The traffic speed limit affects the delivery time and vehicle fuel consumption, so the nearest delivery scheme does not necessarily have the least delivery time or fuel consumption.

In summary, the time-dependent network is closer to the realistic transportation network than the static network. However, the route from node to node is unique in the time-dependent network. However, there are multiple routes between nodes in real road networks. Furthermore, the actual urban road network is complex, and the speed of vehicles on the same road at the different time periods is random and fluctuating. At the same time, when a vehicle moves from one node to another, it often passes through multiple nodes, and the routes between the nodes are not unique. Therefore, how to solve the optimization problem of green location-path of random road network and multinode road network is the next research direction.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors’ Contributions

Hua-xing Zhang conceptualized the study; Chun-miao Zhang curated the data; Chun-miao Zhang carried out formal analysis; Hua-xing Zhang carried out funding acquisition; Hua-xing Zhang investigated the study; Chun-miao Zhang developed the methodology; Hua-xing Zhang administrated the project; Hua-xing Zhang collected resources; Chun-miao Zhang helped with software; Hua-xing Zhang wrote the original draft; Hua-xing Zhang reviewed and edited the study.

Acknowledgments

This work was supported by the Zhejiang Provincial Natural Science Foundation of China, under Grant Y18F030036.

References


selection for software product line testing,” *IEEE Computational Intelligence Magazine*, vol. 12, no. 2, pp. 34–45, 2017.

