Modified Discrete Glowworm Swarm Optimization Algorithm Based on Time Window Division for Multi-objective VRPTW

Wenbo Dong[1], Kang Zhou[1], Gexiang Zhang[2], Han-Chieh Chao[1,3,4,5]

[1] School of Mathematics and Computer Science, Wuhan Polytechnic University, China
[2] School of Electrical Engineering, Southwest Jiaotong University, China
[3] Department of Electrical Engineering, National Dong Hwa University, Taiwan
[4] Department of Computer Science and Information Engineering, National Ilan University, Taiwan
[5] School of Information Science and Engineering, Fujian University of Technology, China
Dongwb7@foxmail.com, zhoukang65@whpu.edu.cn, zhgxdylan@126.com, hcchao@gmail.com

Abstract

A modified discrete glowworm swarm optimization algorithm based on time window division (MDGSO-TWD) is proposed to solve multi-objective vehicle routing problems with time windows (MVRPTW), which is a well-known NP-hard combinatorial optimization problem. In MDGSO-TWD, customers are divided into different classes according to their time windows so as to reduce the population generation time and neighborhood search time; the accuracy of solutions is enhanced by introducing memetic local search and several improvement measures of standard GSO; a new Pareto technique is designed to screen out excellent individuals. Parameter setting of MDGSO-TWD is discussed to enhance the algorithm performance. The feasibility and effectiveness of MDGSO-TWD is verified by analyzing the improvement strategies and comparing with standard GSO. Furthermore, the superiority of MDGSO-TWD is hightlighted by comparing with the state-of-the-art algorithms and the best results reported in the literature. Experimental results show that the proposed algorithm is competitive for MVRPTW in terms of the quality of solutions and the computing time.

Keywords: MVRPTW, MDGSO-TWD, Time window division, Memetic search, Pareto

1 Introduction

A multi-objective vehicle routing problem with time windows (MVRPTW) is a typical variant of vehicle routing problems (VRP), which is a class of well-known NP-hard combinatorial optimization problem. In MVRPTW, a set of vehicles with limited capacities is to be routed from a central depot to a set of geographically distributed customers with known demands and predefined time window [1]. MVRPTW has a wide range of applications in computer science, operational research and combination optimization. In addition, many problems can be abstracted as MVRPTW. However, several variants of the VRP have been studied over the past 50 years, even the simplest version of the VRP is still computationally difficult. MVRPTW is the variant of VRP obtained by adding time windows constraints to the model and its objective is to minimize the number of vehicles and the total traveling distance. Hence, both the spatial arrangement of vehicles and the time constraints have to be considered. Meanwhile, a Pareto solution set, rather than a single solution, is required [3, 20].

Due to the inherent complexities and usefulness of the MVRPTW, more and more attention is paid to this problem and a variety of methods were presented to handle MVRPTW [21]. These methods can be roughly classified into two groups: exact algorithms and heuristic algorithms. In exact algorithms, the computing time required to obtain optimal solutions is often prohibitive when the problem size becomes large. Exact algorithms are more efficient in the cases where the solution space is restricted by narrowing time windows because there are fewer combinations of customers to define feasible routes. Heuristic algorithms are commonly devised to find the optimal or near-optimal solutions for VRPTW within a reasonable computing time.

In the past years, various methods, such as genetic algorithm [2], ant colony optimization [4-5], differential evolution [8] and neighborhood search [9], were used to solve VRPTW, but a small number of studies focused on MVRPTW [16-19]. On the other hand, most of the existing heuristic methods for MVRPTW considered multiple objectives as a single objective by using penalty functions or weighting functions [10]. Several studies discussed multi-objective optimization

*Corresponding Author: Kang Zhou; E-mail: zhoukang65@whpu.edu.cn
DOI: 10.3966/160792642018011901001

techniques for MVRPTW. Tan *et al.* [26] discussed a hybrid multi-objective evolutionary algorithm (HMOEA) that incorporates various heuristics for local exploitation in the evolutionary search and the concept of Pareto's optimality for solving the MVRPTW optimization. Ombuki *et al.* [12] propose a multi-objective genetic algorithms for vehicle routing problems with time windows was presented. Ghoseiri and Ghannadpour [13] propose an algorithm combining the goal programming and genetic algorithm was introduced to solve the multi-objective vehicle routing problems with time windows. Banos *et al.* [14] propose a simulated annealing-based parallel multi-objective approach was proposed to solve MVRPTW. These methods have obtained satisfactory results [15], however, only a point along the Pareto optimal front, instead of a set of Pareto optimal front, was considered [6].

Of various heuristic algorithms, a glowworm swarm optimization algorithm (GSO) is a new type of intelligent bionic swarm optimization algorithm. GSO was first presented by Krishnanand and Ghose in 2005 to model the collective behavior in robotics. This is a novel algorithm for the simultaneous computation of multiple optima of multimodal functions. The algorithm shares a few features with several well-known swarm intelligence based optimization algorithms, such as ACO and PSO [27-28], but GSO has several distinct characteristics [23, 29]. The agents in GSO are thought of as glowworms that carry a luminescence quantity called luciferin. The glowworms encode the fitness of their current locations as a luciferin value and pass it on to their neighbors. A glowworm identifies its neighbors and computes its movements by exploiting an adaptive neighborhood, which is bounded above by its sensor range. Using a probabilistic mechanism, each glowworm selects a neighbor that has a luciferin value higher than its own and moves toward it. These movements consider only local information and selective neighbor interactions and enable the swarm of glowworms to be partitioned into disjoint subgroups that converge on multiple optima of a given multimodal function [31]. Until now, GSO has been successfully applied to the complicated multi-objective function optimization, the signed source localization, sensor noise processing. These studies show good performance [32]. Discrete Glowworm Swarm Optimization algorithm (DGSO) has also been successfully applied to solve travelling salesman problems (TSP) [34, 37] and shows strong vitality. More information and applications of GSO can be found [33]. However, there is a big gap to enhance GSO's performance with respect to search capability, the quality of solutions and elapsed time. Furthermore, the application of GSO to MVRPTW is also worth discussing.

In this paper, a modified discrete glowworm swarm optimization algorithm based on time window division (MDGSO-TWD) is proposed to solve a class of well known combinatorial optimization problem, multi-objective vehicle routing problems with time windows (MVRPTW). MDGSO-TWD divides the customers with different time windows into different classes to decrease the generation time of population and neighborhood search time. Several measures, such as the motion equations and related motion rules of glowworm, adaptive flying step, and elite strategy, are introduced to improve the performance of proposed algorithm. To avoid the premature convergence and to increase population diversity, a memetic local search (MLS) strategy is presented to intensify local search ability [22, 25]. Furthermore, a new Pareto technique is designed to screen out excellent individuals, which returns a Pareto solution set rather than a single solution. Extensive experiments carried out a large set of test instances show that MDGSO-TWD outperforms several existing methods.

The remaining parts of this paper are organized as follows. Section 2 states the problem to solve. Section 3 presents MDGSO-TWD. Extensive experiments are shown in Section 4. Finally, concluding remarks are given in Section 5.

2 Problem Formulation

Multi-objective optimization problems widely exit in the field of engineering and science [11, 24]. As a typical class of multi-objective optimization problems, MVRPTW is given by a special node called depot, a set of customers C to be visited and a directed network connecting the depot and the customers. Assume that there are $n+1$ customers, $C=\{0,1,2,...,n\}$, and for simplicity, depot is denoted as customer 0. There are a certain number of homogeneous vehicles in the depot, so vehicles must leave from and return to the central depot. Every customer in the network must be visited only once by one of the vehicles. Since vehicle k has a limited capacity q_k, and customer i has a varying demand m_i, q_k must be greater than or equal to the summation of all demands on the route traveled by that vehicle k. It is assumed that K is the total number of vehicles and is calculated according to (1).

$$K = floor(\sum_{i=1}^{n} m_i / \alpha q_k) + 1 \quad \textbf{(1)}$$

where floor(*) represents an integer no greater than the value in the bracket; α $(0<\alpha<1)$ is an estimated value. The more the constrained conditions there are, the smaller value α has. In this paper, α=0.5. A route is defined as a line starting from the depot, going through a number of customers and returning the depot. On the other hand, any customer i must be served within a predefined time interval $[e_i, l_i]$, limited by an earliest arrival time e_i and the latest arrival time l_i. Vehicles

arriving earlier than the earliest arrival time incur waiting and waiting time is permitted without cost.

The model has two types of decision variables. For customer *i*, customer *j*, and each vehicle *k*, the decision variable x_{ijk} is equal to 1 if vehicle *k* drives from *i* to *j*, otherwise, the decision variable x_{ijk} is equal to 0. In this paper, x_{iik}=0. The decision variable a_{ik} denotes the arrival time of each vehicle *k* at node *i*. In order to formulate the model, the following notations are defined:

f_{ik}: service time for vehicle *k* at node *i*

w_{ik}: waiting time for vehicle *k* at node *i*

z_{0k}: departure time of vehicle *k* from the central depot

It can be known that ($z_{0k}= w_{0k}= f_{0k}= 0$) according to the problem. The objective of the MVRPTW is to serve all the *C* customers such that the following objectives and constraints are satisfied.

2.1 Objectives

Minimize the distance traveled by the vehicles.

Minimize the total number of vehicles used to serve the customers.

2.2 Constraints

Vehicle capacity constraints are observed.

Time window constraints are observed.

Each customer is served exactly once.

Each vehicle starts its journey from depot and return the depot.

Therefore, the problem is formulated as follows:

$$\min f_1=\sum_{k=1}^{K}\sum_{i=0}^{n}\sum_{j=0}^{n} d_{ij}x_{ijk} \quad \textbf{(2)}$$

$$\min f_2=\sum_{k=1}^{K}\sum_{j=1}^{n} x_{0jk} \quad \textbf{(3)}$$

Subject to:

$$\sum_{j=1}^{n} x_{0jk}=\sum_{j=1}^{n} x_{j0k}\le 1; \qquad (k\in\{1,\cdots,K\}) \quad \textbf{(4)}$$

$$\sum_{i,j\in S} x_{ijk}\le |S|-1; \qquad (\forall S \subseteq C/\{0\},\ 2\le|S|\le n-1) \quad \textbf{(5)}$$

$$\sum_{k=1}^{K}\sum_{j=0,j\ne i}^{n} x_{ijk}=1; \qquad (i\in\{1,\cdots,n\}) \quad \textbf{(6)}$$

$$\sum_{i=1}^{N} m_i \sum_{j=0,j\ne i}^{n} x_{ijk}\le q_k; \qquad (k\in\{1,\cdots,K\}) \quad \textbf{(7)}$$

$$\sum_{k=1}^{K}\sum_{i=0,i\ne j}^{N} x_{ijk}(a_{ik}+t_{ij}+f_{ik}+w_{ik})\le a_{jk}; (j\in\{i,\cdots,n\}) \quad \textbf{(8)}$$

$$e_i\le(a_{ik}+w_{ik})\le l_i; (i\in\{1,\cdots,N\},k\in\{1,\cdots,K\}) \quad \textbf{(9)}$$

Eqs. (2) and (3) are referred to as the two objectives of the constructed multi-objective VRPTW; Eqs. (4)-(6) are three constraints to guarantee that every route starts from and return the central depot and every customer node is visited only once by one vehicle; Eq. (7) is the capacity constraint; Eqs. (8) and (9) are two constraints for defining the time windows.

3 MDGSO-TWD

In this section, the framework of MDGSO-TWD is first presented and then each of the main processes is described in detail. In the initial state, a population of glowworms is randomly placed in the search space so that they are well distributed; all the glowworms contain an equal quantity of luciferin l_0 [31]. Each iteration consists of a luciferin-update phase followed by a movement phase based on a transition rule.

As shown in Figure 1, the time window is **first** divided according to Algorithm 1 (Note: the following symbols and algorithms would be described in detail in the following sections), we can get customer sets *class(i)*, *i=1,2,...,K*. **Second**, initialization work is done, including population initialization, parameters initialization, and Pareto set initialization. About parameters initialization, more details can be referred to [33, 34]. At this step, we can get the initial population $G^t=\{g_1^t,g_2^t,\cdots,g_N^t\}$, where *t=0*, and the initial Pareto set *NDset* = $\varnothing$. **Third**, the path is decoded and is transformed into objective function value. Meanwhile, the luciferin is updated. The detailed process can be referred to [33-34]. **Fourth**, in movement phase, each glowworm uses a probabilistic mechanism to move toward a neighbor that has a luciferin value higher than its own. **Fifth**, the location of the glowworm is updated according to the update rules. According to the above two steps, for a particular individual of glowworms $g_i^t=(g_{i_1}^t,g_{i_2}^t,\cdots,g_{i_n}^t)$, *i=1,2,...,N*, we can get $g_i^{t+1}=(g_{i_1}^{t+1},g_{i_2}^{t+1},\cdots,g_{i_n}^{t+1})$. **Sixth**, the decision domain radius of the glowworm is adjusted according to the neighbor density. **Seventh**, a modified memetic local search is used to obtain better individuals. According to the above two steps, we can get $\tilde{g}_i^{t+1}=(\tilde{g}_{i_1}^{t+1},\tilde{g}_{i_2}^{t+1},\cdots,\tilde{g}_{i_n}^{t+1})$, *i=1,2,...,N*. **Finally**, a Pareto technique is designed to screen out excellent individuals. This procedure repeats until a termination condition is satisfied. At the end of the algorithm, we can get $G^{t+1}=\{\tilde{g}_1^{t+1},\tilde{g}_2^{t+1},\cdots,\tilde{g}_N^{t+1}\}$. In what follows, we describe the main steps of MDGSO-TW step by step.

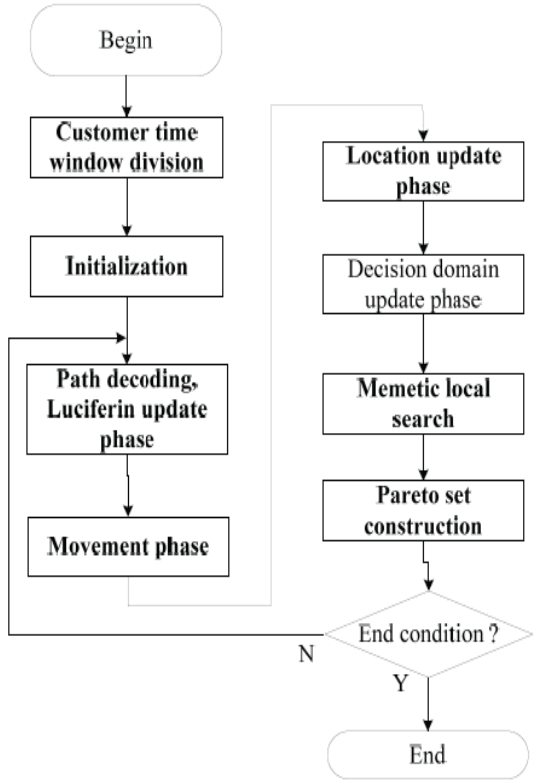

Figure 1. Schematic graph of MDGSO-TW. The bold font in the box highlights the improvements

3.1 Customer Time Window Division

In this subsection, customers are divided into different classes according to their time windows so that the population generation time and neighborhood search time could decrease. Differing from the studies in the literature, MDGSO-TWD considers the characteristic of MVRPTW. Time windows are animportant part of MVRPTW. MVRPTW could be quickly solved if we make the full use of time window. Related to the VRP, the delivery time and time window of MVRPTW should be considered. The main reason is the limitation of the customer's earliest arrival time and latest arrival time. Under this restriction, MVRPTW considers the scheduling of time in addition to the spatial aspect. At the same time, due to the limitation of depot's time window, it indirectly causes the limitation of route length. Hence, if we want to get a good solution, it is important to explore the problem of time and space. Firstly, customers are divided into different classes, then the classified results are used for initial population generation and neighborhood search to reduce time. The division of customers into different classes is useful to take different actions to different customers. For instance, customers' service time in the same class are mutually exclusive, so we only need to take actions for customers who are in different classes. This will greatly improve structure speed of the initial population. In the subsequent local search, customers' service time in the same class is also similar, so we only need to take neighborhood operations for customers who are in the same class. Compared with the blind search, this will greatly improve the search speed of the algorithm. The division of customers' time window is the foundation of the subsequent steps of MDGSO-TWD.

The procedure of time division is described in detail in Algorithm 1. First of all, the time window of depot is calculated and is divided into K different subintervals, as shown in Figure 2. Second, the service time window of the customer c is $[e_c, l_c]$. For a subinterval j, if $[e_c, l_c] \cap [e_0+j*T/K, e_0+(j+1)*T/K] \neq \varnothing$, then the customer c is divided into this category. This process repeats until all customers are classified. In Figure 2, K is an estimated value of vehicles. In the experiment, we found that the customer clustering results obtained by the clustering method are approximately equal to K, hence we use the estimated value as the number of time window segmentation; e_0 is the earliest arrival time of the depot; l_0 is the latest arrival time of the depot; $T=l_0-e_0$. Assume that c is a customer and his service time window is $[e_c, l_c]$. The pseudocode of Algorithm 1 shows the detailed steps.

Algorithm 1. Time window division

Require:
$[e_c, l_c]$: the service time window of customer c, $c \in C$
T: the time window of depot
K: the number of vehicles
Ensure: *class(i)*, for i=1 to K
1: class(i) $\leftarrow \varnothing$, for i=1 to K
2: Calculate $Timewindow(class(i)) = [e_0+(i-1)*T/K, e_0+i*T/K]$, for i=1 to K, $T = l_{depot} - e_{depot}$
3: **for** ($\forall c \in C$) **do**
4: $C \leftarrow C/\{c\}$
5: **for** (j=1; $j \leq K$; j++) **do**
6: if ($[e_c,l_c] \cap Timewindow(class(j)) \neq \varnothing$) then
7: $class(j) \leftarrow class(j) \cup \{c\}$
8: **end if**
9: **end for**
10: **end for**
11: **return** *class(i)*, for i=1 to K

Figure 2. Time window schematic diagram

We give an example to explain this algorithm. As shown in Figure 3, there are two periods [10, 20] and [20, 30], which are calculated based on the first three steps of Algorithm 1, and the service time of customers i and j is in the intervals [15, 25] and [22, 28], respectively. The customer i will be added to the

customer class whose service time window is within [10, 20] and [20, 30], and the customer *j* will be added to the customer class whose service time window is within [20, 30].

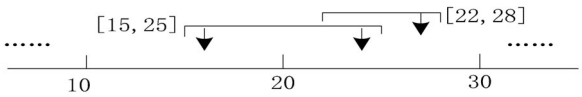

Figure 3. Customer segmentation schematic diagram

This concept is applied to construct the initial population and neighborhood search. It not only speeds up the construction of initial population, but also improves the optimization efficiency of neighborhood search.

3.2 Population Initialization

In this section, the population is initialized according to Algorithm 2. First, a two-dimensional space is constructed to match the vehicle *k* and the customer *r*. Second, based on the Algorithm 1, the customers are divided into *K* subclasses according to their time windows. If the customer selected from the first subclass satisfies the load of a vehicle, then the customer is assigned to the *ith* routing. Next, the customer selected from the second subclass is used to judge whether it meets the load of the vehicle. If this condition is not satisfied, then a new vehicle would be used to service the rest of the customers. More details are shown in Algorithm 2.

Algorithm 2. Initial population generation based on time window

Require:
K: the number of vehicles
Class(1), class(2), · · ·, class(K): time window class
N: glowworms population size
n: the number of customers
vnum: a count variable for vehicle
snum: a count variable for time window class
C(vnum): the customer group is served by the *vnu*m vehicle
q: vehicle's capacity
w: current demand of customers
e_c: the earliest arrival time of customers *c*
t[the last customer of c][c]: the arrival time from the last customer to *c*
Ensure: $G=\{g_1,g_2,\cdots,g_N\}$: initial population of glowworms
1: **repeat**
2: *vnum*=1, *snum*=1, *X*=1, *w*=0 % *X*: a variable used to do loop
3: **repeat**
4: **for** (*i=1; i<=K; i++*) **do**
5: **if** ((*class (i)* $\neq \Phi$) $\wedge$ (*w<q*) $\wedge$ (*t[the last customer of c][c]*<e_c)) **then**
6: select randomly *c* ($c \in class(i)$)
7: $C(vnum) \leftarrow C(vnum)+\{c\}$
8: $class(p) \leftarrow class(p)-\{c\}$, $p=_{1,\cdots,K}$, where $c \in class(p)$
9: **end if**
10: **end for**
11: $vnum \leftarrow vnum+1$, $w=0$
12: **until** all customers are serviced
13: $g_X \leftarrow \{C(1), C(2), \cdots, C(vnum)\}$
14: $X \leftarrow X+1$
15: **until** the value of *X* is equal to *N*
16: **return** $\{g_1,g_2,\cdots,g_N\}$

An example is also taken to explain this algorithm. In MVRPTW, the number of customers is 7 and they are divided into {1, 5}, {2, 3, 6} and {4, 7}. According to the above steps, we can get:

Vehicle 1: $0 \rightarrow 1 \rightarrow 3 \rightarrow 0$
Vehicle 2: $0 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 0$
Vehicle 3: $0 \rightarrow 6 \rightarrow 7 \rightarrow 0$

We get the coding sequence (1, 3, 5, 2, 4, 6, 7). This sequence is only a possible candidate solution. Repeat this step several times, until it reaches a population size. This method guarantees that the initial solutions are feasible and makes the initial population spread out as much as possible in the solution space.

3.3 Luciferin Update Phase

The luciferin update phase depends on the function value at the glowworm position. During the luciferin-update phase, each glowworm adds to its previous luciferin level. A luciferin quantity proportional to the fitness of its current location in the objective function domain. A fraction of the luciferin value is also subtracted to simulate the decay in luciferin with time. The luciferin update rule was given in [31]. The fitness of glowworm's current location is obtained by decoding a coding sequence of an individual. In MDGSO-TWD, a decoding method is proposed to simplify the operation and reduce the time in the decoding process. The MVRPTW is represented as a list of *n* nodes, where *n* is the number of customers, and node $i \in [1, n]$ represents a customer. The problem representation is the order of customers without route splitters like a string (9, 3, 4, 8, 7, 2, 5, 1, 6). This representation does not include the information, where the list is broken by returning the depot or terminal points. To identify such terminal points, we have adopted the split procedure, which works through the customer sequence embedded in the chromosome. When decoding a sequence, the system only needs to insert the customers into the path as much as possible. If a customer does not satisfy the constraints of time window or capacity, the system will open up a new path to serve him.

3.4 Movement Phase

During the movement phase, each glowworm uses a

probabilistic mechanism to decide to move toward a neighbor that has a luciferin value higher than its own. Differing from the related work reported in the literature [34], this paper introduces a definition of distance between individuals. Assume that there are two glowworm individuals $g_i = (g_{i_1}, g_{i_2}, \cdots, g_{i_n})$ and $g_j = (g_{j_1}, g_{j_2}, \cdots, g_{j_n})$. The distance between glowworms g_i and g_j can be defined as

$$Dist_{(g_i,g_j)} = c \times \frac{\sum_{k=1}^{n} dist(g_{i_k}, g_{j_k})}{\sum_{k=1}^{n} \max\{dist(g_{i_k}, g_{j_m}) : m = 1,2,\cdots,n\}} \quad \textbf{(10)}$$

where *dist(a,b)* is the distance between customer *a* and customer *b*; g_{i_k} and g_{j_k} represent the k^{th} component in g_i and g_j; denominator is the sum of the maximum in each row in the weight matrix; *c* is a constant.

3.5 Location Update Phase

Each glowworm is a candidate solution represented by a set of codes. Thus, each flight is equivalent to the code that updates once. So new distance update rules associated with the flight step *s* of the glowworm are designed. The rules differ from the discretization processes in the literature. The rules do not abandon the concept of the flight step of glowworm, but inherit it, which makes it more close to the original algorithm. Distance update rules are as follows: in the mobile phase, a sequence consisting of *s* different random integers is first generated, $r(s)=(r_1, r_2, ..., r_s)$, where $r_k \in [1,n]$. The glowworm $g_i^t = (g_{i_1}^t, g_{i_2}^t, \cdots, g_{i_n}^t)$ flies to the glowworm $g_j^t = (g_{j_1}^t, g_{j_2}^t, \cdots, g_{j_n}^t)$ by following the flight rules: First, we can get $PariS = \{(g_{i_k}^t, g_{j_k}^t) \mid k = 1,2,\cdots,s\}$; Second, for $k(k<=s)$, glowworm $g_i^t = (g_{i_1}^t, g_{i_2}^t \cdots, g_{i_{r_k}}^t, \cdots, g_{i_m}^t, \cdots, g_{i_n}^t)$, where $g_{i_m}^t = g_{j_{r_k}}^t$, we can get $g_i^{t+1} = (g_{i_1}^t, g_{i_2}^t \cdots, g_{j_{r_k}}^t, \cdots, g_{i_{r_k}}^t, \cdots, g_{i_n}^t)$, repeat second steps, *k* from 1 until *s*; Finally, we can get $g_i^{t+1} = (g_{i_1}^{t+1}, g_{i_2}^{t+1}, \cdots, g_{i_n}^{t+1})$ and implement the location update from g_i^t to g_i^{t+1}.

3.6 Decision Domain Update Phase

In this phase, an adaptive step size adjustment strategy is introduced into MDGSO-TWD. In standard GSO, if the glowworm's flying step length is large, it is easy to search in the global range, but also it is difficult to find the high-precision solutions because the concussion phenomenon often appears. If the flying step length is small, it will improve the accuracy of the solution, but it is easy to fall into the local optimum. So we consider to dynamically adjust the parameter of step size according to the search results of different stages, and deal with the relationship between the global optimization ability and the accuracy. In MDGSO-TWD, an adaptive step size is designed. The main idea is to make the glowworms living farther away fly faster and the glowworms living closer fly slowly so that the optimal solution can be found. This update can improve the searching efficiency of the algorithm. The adaptive step size is

$$s^* = \left| \frac{i - iter}{iter} \right| \times s \quad \textbf{(11)}$$

where *i* represents the number of iterations; *iter* represents the maximum number of iterations; *s* represents previous generations' step size.

When a glowworm flies to the target glowworm, a small step *s* can make the glowworm more close to the target glowworm without skipping the optimal solution, which can improve the accuracy of the search. Thus, the flight step at the current iteration is updated dynamically based on the results at the previous iteration. Standard GSO used a fixed values for the fly step size, so a large fly step size *s* will affect the accuracy of the solution. On the contrary, a smaller value of *s* will affect the search speed. So the dynamic adjustment of the step size according to the current situation can balance the search speed and search accuracy of the algorithm. Therefore, in the early stages of the algorithm, a large step size can ensure coarse search in a large range of solution space, which can quickly search the neighborhood of global optimal. In near optimal neighborhood, the algorithm has a small step length, which can gradually evolve into a precise search, which can be more accurate to search the optimal solution.

3.7 MLS Strategy

In this subsection, a memetic local search (MLS) strategy based on time window division is introduced. MLS uses the local heuristic search to simulate the variation process to generate a better solution than the old one. In the previous subsections, customers are divided into different classes, which would not be able to produce feasible solutions if customers' service time windows are in different classes or their time windows are far apart. So the MLS strategy is used for the customers whose service time windows are in the same class. This greatly reduces the computation of the algorithm, and strengthens the general optimization and convergence speed of the algorithm. The MLS uses swap, reverse, and 2-opt operations.

For each glowworm, the new solution obtained by the MLS is compared with the original solution, if new solution is better than old one, then the old solution will be updated by new one. Otherwise, the solution keeps unchanged. Details are given in Algorithm 3.

Algorithm 3. Memetic local search based on time window division

Require:

g_i: glowworm i

$\prec$: Pareto dominate. $g_1 \prec g_2$, if and only if objective function $f_i(g_1) \le f_i(g_2)$, $\forall i \in 1,2$, and there exits at least one objective $j \in 1,2$ that satisfies $f_j(g_1) < f_j(g_2)$. In this paper, there are two objectives, one is the number of vehicles, the other is the distance traveled by vehicles.

Ensure: g_i

1: **repeat**
2: select a number c randomly in $g_i =$ (g_{i_1} , g_{i_2} , $\cdots$, g_{i_n}) ,where $c = g_{i_p}$, $p \in \{1,2,\cdots,n\}$
3: $\exists\ class(k)$, s.t. $c \in class(k)$
4: select a customer c' randomly in *class(k)*
5: $g_i' \leftarrow memetic\_local\ search(g_i, c, c')$
6: **if** $g_i' \prec g_i$ **then**
7: $g_i \leftarrow g_i'$
8: **end if**
9: **until** terminal criterion is satisfied
10: **return** g_i

Due to the constant iteration of the algorithm, glowworms will fly to the same location, which indicates that the code of glowworms will be identical. Therefore, at each iteration of the algorithm, some glowworms will be generated randomly to replace repeating codes. This measure can maintain the diversity of the glowworm population and the glowworm has a wider space to fly.

3.8 Pareto Set Construction

In this subsection, an algorithm is introduced to find the Pareto non-dominated solution set [30]. Assume that *Pop* represents glowworm population, *S* represents a tectonic set, *S=Pop*; *NDset* is a non-dominated set and is initialized to an empty set. First of all, the codes of the glowworm is decoded, and *S* is sorted in ascending order according to the number of vehicles. And then an exclusive operation is used to filter the glowworms. Exclusive operation is as follows: if two solutions have the same number of vehicles, we only keep the solution with the minimum traveling distance. Because only those individuals can participate in the structure of Pareto solution set. More details are described in Algorithm 4.

Algorithm 4. Construction algorithm of Pareto solution set

Require:

NDset: Pareto non-dominated set

g_i : glowworm i

$\prec$: Pareto dominate

Ensure: *NDset*

1: $S \leftarrow Pop$
2: Sort (S) according to the number of vehicles
3: Exclusive Operation: if the decoded number of vehicles are the same, we only keep minimum value path of glowworms individuals
4: **for** ($\forall g \in S$) **do**
5: **if** ($\exists g' \in NDset,\ s.t.\ g \prec g'$) **then**
6: $g' \leftarrow g$
7: **else**
8: $NDset \leftarrow NDset \bigcup \{g\}$
9: **end if**
10: $S \leftarrow S - \{g\}$
11: **end for**
12: **return** *NDset*

4 Experiments and Analysis

To investigate the performance of the proposed MDGSO-TWD, extensive experiments are carried out standard Solomon's VRPTW benchmark problem instances available at http://web.cba.neu.edu/~msolomon/problems.htm [36]. The standard Solomon's problems consist of 56 data sets, which have been extensively used for different heuristics in literature over the past years. There are three different sizes of data sets (25, 50 and 100) in the database. Each data set contains 56 test problems, which are divided into six categories: *C1*, *C2*, *R1*, *R2*, *RC1* and *RC2*. Problems in sets *R1* and *R2* have the customers' locations generated randomly over a square. Problems in sets *C1* and *C2* have the clustered customers whose time windows generated based on a known solution. Problems in sets *RC1* and *RC2* have a combination of randomly placed and clustered customers. Problems in sets *R1*, *C1* and *RC1* have narrow time windows and a small capacity of the vehicle, while problems in sets *R2*, *C2* and *RC2* have larger time windows and a larger capacity of the vehicle. All the experiments are implemented by using programming language VC 6.0 and on a PC with 1.5GHz CPU and 4GB memory.

As described above, a number of elements, such as algorithmic components and parameter choices, many affect the performance of our algorithm. Thus, a series of experiments was designed to investigate their impact. In what follows, Section 4.1 discusses the parameter setting by generating the solutions to a few selected problems. The experimental results and analysis are presented in Section 4.2.

Table 1. Experiment results of different numbers of glowworms

glowworms	10	20	30	40	50	60	70
C101	(13,1226.1)	(13,1216.2)	(13,1148.2)	(13,1135.4)	(13,1130.1)	(12,1208.4)	(12,1193.7)
glowworms	80	90	100	110	120	130	140
C101	(12,1178.3)	(12,1172.2)	(12,1163.3)	(12,1160.4)	(12,1144.9)	(12,1123.3)	(12,1100.9)
glowworms	150	160	170	180	190	200	210
C101	(12,1093.0)	(12,1087.5)	(12,1051.4)	**(11,976.6)**	**(11,976.6)**	**(11,976.6)**	**(11,976.6)**

4.1 Parameters Settings

In MDGSO-TWD, according to the studies in [34], several parameters are set as follows: ρ=0.6, γ=0.7, β=0.05, n_t=20, l_0=15, r_0=7. We investigate how the population size and the number of iterations affected the convergence performance of the algorithm. As usual, large values of population size and the number of iterations produce good accuracy of solutions, however, this will result in much computing time. So it is necessary to balance the number of glowworms and the number of iterations to obtain satisfactory results.

The precision of solutions can often be improved by increasing the number of glowworms. However, when the number of glowworms reaches a certain value, it is very hard to further improve the precision by increasing the number of glowworms. On the contrary, the increase causes the growing up of time complexity to affect the optimization efficiency of the algorithm. Therefore, in this section, the number of glowworms is discussed according to the problem's size under the high-precision precondition. the results on the comparisons of several numerical tests in scale 100 are shown in Table 1. The experiments were repeated for 20 times and the average values was calculated for each group. We use v=(n, c) to represent a candidate solution of the MVRPTW, where n is the number of vehicles and c is the traveling distance.

As shown in Table 1, the numerical example is *C101*. Problems in sets *C1* have the clustered customers, narrow time windows and a small capacity of the vehicle, so we chose *C101* to do the parameter selection experiment. As can be seen from the table, as the number of glowworms increases, the accuracy of the vehicles and the traveling distance have been improved. When the number of glowworms is around 180, the solution remains the same, and it reaches the known optimal solution in the database. Since the results we obtained is the average results, in this paper, the number of glowworms is set to 180.

To obtain the relationship between the number of iterations and the calculation results, this paper uses the numerical example *R201*, which has 100 customers. Different from *C1*, problems in sets *R2* have the customers' locations generated randomly over a square, and they have larger time windows and a larger capacity of the vehicle. Therefore, we choose these two different and representative examples to do the parameter selection experiment. Specific results are shown in Figure 4.

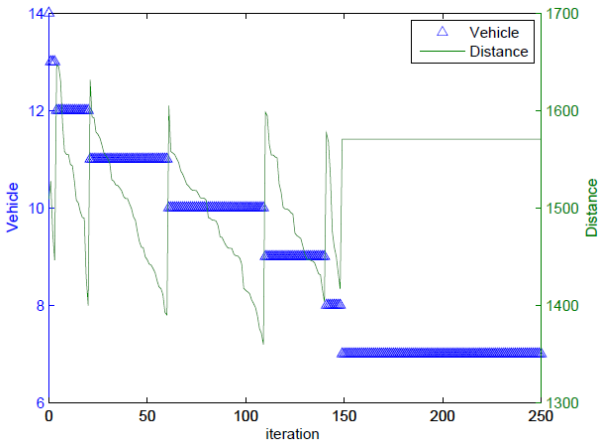

Figure 4. Influence of the number of iterations on convergence. The blue line represents the changing process of the vehicles, and the green line represents the changing process of the total distance.

As can be seen from Figure 4, with the increasing of iterations, the number of vehicles converges from 14 to 7. When the number of vehicles remains constant, the traveling distance of the vehicle decreases with the increasing of iterations. At last, when the number of vehicles is 7, the traveling distance of the vehicle stays at the value of 1570. This shows that the algorithm converges to the known optimal solution. So in this paper, the recommended value of the maximal number of iterations is 200.

4.2 Experimental Results and Comparisons

In this section, several instances of six types of problems with respective 25, 50 and 100 customers in the database are used to verify the effectiveness and feasibility of the proposed algorithm.

Comparisons with standard GSO. The first comparison is performed between MDGSO-TWD and the standard GSO in [34]. Experimental results with respect to the instance *R101* are shown in Figure 5. It can be seen from Figure 5 that the proposed algorithm reduces from 33 vehicles to 21 vehicles, while the traditional GSO reduces from 32 vehicles to 25 vehicles, more than 4 cars. The routing distance of MDGSO-TWD steadily decreases from 2700 to 1700, while the traditional GSO decreases from 2500 to 1900. The results indicate the advantage of MDGSO-TWD.

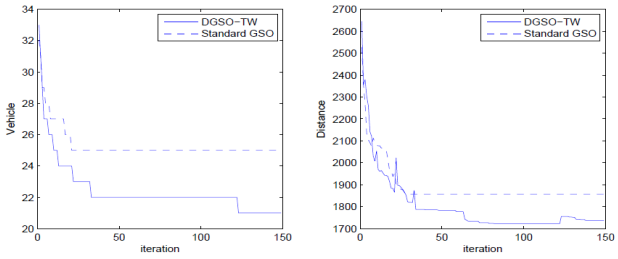

Figure 5. Comparisons of MDGSO-TWD and standard GSO

Comparisons with improvement strategies. We assess the importance of the different components in our algorithm to its overall performance. The algorithm contains several ideas and ingredients, aiming at improving the quality of the solutions. In this section, the role of the improved strategies proposed in this paper to the algorithm performance is evaluated by some experimental results. The average results of seven instances are used to quantify the contribution of each strategy and the results are as shown in Table 2. In Table 2, we use *C1* to contrast experiments. The geographical data are clustered in problem in *C1*, meanwhile, *C1* has a short scheduling horizon and allows only a few customers per route. Hence, *C1* is relatively difficult to solve, and if *C1* can be resolved, other types should also be easily solved.

It can be observed from Table 2 that GSO-Memetic search can always provide better performance than GSO. This indicates that the introduction of the memetic search can greatly improve the qualities of the solutions. Furthermore, MDGSO-TWD is better than GSO and GSO-Memetic search with respect to the performance. So the improved strategy in this paper is effective. It is obvious that the computation time of GSO-Adaptive step is always less than GSO for each instance. It is shown that the adaptive step reduces the computation time. GSO-Memetic search always needs large computation time to obtain good solutions. However, MDGSO-TWD requires less computation time than GSO-Memetic search while it possesses competitive solutions. Considering the quality of the solutions and computation time, these strategies are effective.

Comparisons with different heuristics. MDGSO-TWD is compared with the algorithms reported in the literature. Table 3 compares the routing performance between four popular heuristics and MDGSO-TWD based on the average number of vehicles and average total distance.

As can be seen in Table 3, MDGSO-TWD produces the smallest average number of vehicles for categories *R2* and *RC2* and the smallest average total distance for the category *C1*. The average number of vehicles for the categories of *R2* is 2.82\%, which is lower than HMOEA with the second best average total distance.

Table 2. The impact of different components on the quality of the solutions

Data set		*C101*	*C102*	*C103*	*C104*	*C105*	*C106*	*C107*
GSO	NV	18	15	13	11	16	15	15
	TD	1524.36	1395.99	1375.47	1124.34	1366.02	1392.81	1422.18
	CT	342	325	309	287	332	330	323
GSO-Adaptive step	NV	17	16	13	11	17	15	16
	TD	1549.19	1410.35	1363.02	1313.90	1466.09	1392.60	1460.88
	CT	202	194	185	171	202	198	194
GSO-Memetic search	NV	13	12	11	10	12	12	12
	TD	1091.77	1242.75	1275.20	1187.82	1072.45	1188.70	1106.06
	CT	244	236	225	209	238	236	232
MDGSO-TWD	NV	12	12	11	10	12	11	12
	TD	1032.79	1159.20	1190.48	1183.68	1038.02	1108.67	1050.85
	CT	189	184	177	166	185	184	179

Note. "NV"=Average number of vehicles; "TD"=Average total distance; "CT"=Computation time(s).

Table 3. Comparisons with different algorithms

Data set	Best result		GA[12]	ACO[8]	HMOEA[26]	ACO-Tabu[8]	MDGSO
C1	Number of Vehicles	10	10	10	10	10	10
	Total Distance	826.7	835.33	881.44	828.74	841.92	**828.38**
C2	Number of Vehicles	3	3	3.5	3	3.3	3
	Total Distance	589.24	597.12	641.25	590.69	612.75	591.49
R1	Number of Vehicles	12.67	13.58	13.6	12.92	13.1	12.92
	Total Distance	1156.36	1247.93	1383.20	1187.35	1213.16	1228.60
R2	Number of Vehicles	2.82	3.76	4.1	3.55	4.6	**3.45**
	Total Distance	932.35	957.81	1098.22	951.74	952.3	1033.53
RC1	Number of Vehicles	11.88	13.24	12.7	12.38	12.7	12.75
	Total Distance	1368.23	1414.81	1211.12	1355.36	1415.62	1392.09
RC2	Number of Vehicles	3.38	3.95	5.6	4.25	5.6	**3.75**
	Total Distance	1075.67	1193.06	1209.44	1068.26	1120.37	1162.40

Although the average total distance of MDGSO-TWD is not the smallest for categories *RC2* and *R2*, MDGSO-TWD only requires an average of 3.75 and 3.45 vehicles to serve all customers in the categories of *RC2* and *R2*, which is much smaller than the 5.6 and 4.6 vehicles required by ACO-Tabu.

Comparisons with best results. To better verify the stability of the algorithm, the experiments are totally repeated for 20 times and the average is calculated for each group. The statistical results of the experiments are presented in Table 4, where the average value of the optimal solution, the relative deviation between the average and the known optimal solution and the standard deviation between the average and the known optimal solution for each case on the traveling distance are listed.

As can be seen from Table 4, the relative deviation between the average and the known optimal solution is small. When the number of customers is 25, the relative deviation is below 1%, and the standard deviation is 0, which shows that the algorithm can find the known optimal solution. The calculation time is less than 3s, which shows that the algorithm has fast convergence speed and high efficiency. With the increase of customers, the average deviation of the algorithm, the standard deviation and the average calculation time increases to a certain degree, and the overall trend is going up with the increase of customers. In Table 4, the maximum average deviation of the numerical example is less than 9%, and the standard deviation is small. It can be concluded that the algorithm has strong robustness. In addition, in terms of the average computational time, when the number of customers is 100, the computation time increases a lot, but the computation time has a great reduction, comparing with to the results reported in literature. So we can draw conclusions that the algorithm has high efficiency and can converge to a satisfactory solution in an acceptable time.

In the sequel, we consider VRPTW as a multi-objective problem, so the results should be a Pareto set rather than a single solution. Some instances in the database are used. Table 5 presents a summary of results.

From Table 5, we can see that the proposed algorithm can solve MVRPTW very well. The algorithm produces a lot of new solutions, but we only list those solutions whose vehicles or traveling distance is smaller than reference solutions. These solutions are compared with the reference solutions, which provide more options for the decision maker. In the experiments, some numerical examples have a single Pareto solution, which means that the number of vehicles and the traveling distance arrives at the optimum. A tick, on the other hand, indicates that the solution we obtained is the same as the best known. The results obtained by our algorithm are quite good as compared to the best published results found in literature.

From the experimental results, the algorithm is sensitive to data sets and shows different robustness with respect to the different types of data sets. The distribution of data sets and the width of time window can affect the performance of the algorithm. The three kinds of problems *C1*, *R1* and *RC1* have smaller load of the vehicle and their time window is narrow, so the customers that each vehicle can serve are less. On the contrary, the three kinds of problems, *C2*, *R2* and *RC2*, have larger load of the vehicle and their time window is wide, so each vehicle can serve more customers.

From the experimental analysis and comparisons of experimental results, the algorithm can acquire the good solutions for some instances with respect to a certain type because of the different complexity of the problems. Or the algorithm can get the approximate optimal solutions of other instances and their errors are also in the acceptable range. The results show that the introduced algorithm has a certain superiority to several meta-heuristic approaches reported in the literature.

5 Concluding Remarks

In this paper, we introduced a multi-objective discrete glowworm algorithm based on time window division for solving MVRPTW, which incorporates several ideas such as customer time window division, memetic local search and a new Pareto technique. A large number of experiments demonstrated the benefits of our approach on a variety of problems. Experimental results carried out on various instances indicate that the algorithm outperforms the existing heuristic algorithms.

The future research work mainly focuses on more numerical experiments to further verify the performance of the algorithm such as scalability and robustness. In addition, large VRPTW, more types of VRPTW, multi-depot vehicle routing problems with time windows [35], vehicle routing problems with soft time windows, and even more application problems [7] can be considered.

Acknowledgments

The work was supported by National Natural Science Foundation of China (61179032, 61373047, 61672437), the Special Scientific Research Fund of Food Public Welfare Profession of China (201513004-3), the Guiding Scientific Research Project of Hubei Provincial Education Department (B2017078), subproject of the National Key Research and Development Program of China (2017YFD0401102-02) and the Humanities and Social Sciences Fund Project of Hubei Provincial Education Department(17Y071). Kang Zhou is corresponding author.

Table 4. Comparisons with best results reported in the literature

Instances	customers	Best Known [36]		MDGSO-TWD			
		Source[36]	distance	Ave V	Ave RD(%)	Stan D	Ave RT(s)
R201	25	KDMSS	617.1	618.3	0.19	0	2
R208	25	IV+C	328.2	329.3	0.34	0	2
C101	25	KDMSS	191.3	191.8	0.26	0	3
C201	25	CR+L	214.7	215.5	0.37	0	3
RC101	25	KDMSS	461.1	462.1	0.21	0	2
RC208	25	C	269.1	269.6	0.19	0	2
R104	50	KDMSS	625.4	659.8	5.50	15.75	41
R204	50	IV	506.4	530.4	4.74	20	30
C102	50	KDMSS	361.4	392.2	8.52	25.16	67
C204	50	KLM	350.1	365.6	4.43	11.46	70
RC104	50	KDMSS	545.8	546.5	0.13	0	54
RC204	50	DLP	444.2	459.5	3.44	23.77	58
R101	100	KDMSS	1637.7	1714.3	4.68	27.37	440
R201	100	KLM	1143.2	1188.2	3.94	12.60	465
C101	100	KDMSS	827.3	828.9	0.19	0	880
C205	100	CR+KLM	586.4	625.7	6.70	18.22	857
RC101	100	KDMSS	1619.8	1690.2	4.35	8.91	580
RC205	100	IV+C	1154	1201.3	4.10	16.80	610

Note. "Ave V"=Average value; "Ave RD"=Average relative deviation; "Stan D"=Standard deviation; "Ave RT"=Average running time.

Table 5. Comparisons with best results with respect to multiple objectives

Instances	Customers	Best Known [36]	MDGSO-TWD
R105	25	{(6, 530.5)}	{(5, 556.7),(6, 531.5)}
R201	25	{(4, 463.3)}	{(2, 523.7), (3, 487.1), (4, 478.2)\}
R202	25	{(4, 410.5)}	{(2, 471.9), (3, 436.0), (4, 411.2)}
R205	25	{(3, 393.0)}	{(1, 504.5), (2, 405.9), (3, 393.7)}
R206	25	{(3, 374.4)}	{(1, 413.2), (2, 392.2), (3, 376.9)}
R210	25	{(3, 404.6)}	{(1, 516.0), (2, 413.6), (3, 405.2)}
R207	25	{(3, 361.6)}	{(1, 398.0), (2, 374.2), (3, 362.3)}
C202	25	{(2, 214.7)}	{(1, 223.3), (2, 215.5)}
RC202	25	{(3, 338.0)}	{(1, 551.6), (2, 376.1), (3, 339.1)}
RC203	25	{(3, 326.9)}	{(1, 432.5), (2, 358.9), (3, 330.2)}
RC204	25	{(3, 299.7)}	{(1, 327.5), (2, 316.7), (3, 300.2)}
RC206	25	{(3, 324.0)}	{(1, 482.0), (2, 351.1), (3, 325.1)}
RC207	25	{(3, 298.3)}	{(2, 308.6), (3, 298.9)}
R201	50	{(6, 791.9)}	{(3, 1075.9), (4, 1043.7), (5, 840.8), (6,799.8)}
R202	50	{(5, 698.5)}	{(3, 826.1), (4, 801.5), (5, 795.1)}
R203	50	{(5, 605.3)}	{(2, 766.8), (3, 711.8), (4, 702.4), (5, 696.1)}
R205	50	{(4, 690.1)}	{(3, 811.0), (4, 783.9), (5, 766.3)}
R206	50	{(4, 632.4)}	{(2, 780), (3, 701.6), (4, 687.3)}
R210	50	{(4, 645.6)}	{(2, 748.8), (3, 737.4), (4, 706.5)}
R209	50	{(4, 600.6)}	{(2, 724.3), (3, 682.3), (4, 676.9)}
C201	50	{(3, 360.2)}	{(2, 485.5), (3, 361.7)}
C202	50	{(3, 360.2)}	{(2, 403.8), (3, 380.2)}
RC201	50	{(5, 684.8)}	{(3, 919.2), (4, 962.1), (5, 899.1), (6, 817.4)}
RC202	50	{(5, 613.6)}	{(3, 793.6), (4, 771.8), (5, 755.6), (6, 735.4)}
RC203	50	{(4, 555.3)}	{(2, 715.5), (3, 626.4)}
RC205	50	{(5, 630.2)}	{(3, 910.2), (4, 791.1), (5, 716.2)}
RC206	50	{(5, 610.0)}	{(3, 717.2), (4, 693.2), (5, 686.1)}
RC207	50	{(4, 558.6)}	{(2, 786.6), (3, 599.7), (4, 580.9)}
R203	100	{(4, 935.0)}	{(4, 1123.1), (5, 1066.7), (6, 978.5)}
R204	100	{(3, 789.7)}	{(3, 1130.1), (4, 927.7), (5,831.8), (6, 826.2)}
R205	100	{(3, 994.4)}	{(3, 1422.3), (4, 1154.2), (5, 1156.7)}
RC201	100	{(9, 1261.8)}	{(7, 1639.9), (8, 1553.6), (9, 1542.2)}
RC202	100	{(8, 1092.3)}	{(5, 1734.2), (6, 1671.6), (7, 1503.4), (8, 1325.8)}
RC203	100	{(4, 1026.6)}	{(4, 1269.8), (5, 1187.1), (6, 1020.1)}
RC205	100	{(7, 1154.0)}	{(6, 1634.2), (7, 1388.5)}

References

[1] Z. Ursani, D. Essam, D. Cornforth, R. Stocker, Localized Genetic Algorithm for Vehicle Routing Problem with Time Windows, *Applied Soft Computing*, Vol. 11, No. 8, pp. 5375-5390, December, 2011.

[2] G. B. Alvarenga, R. G. Mateus, G. de Tomi, A Genetic and Set Partitioning Two-phase Approach for the Vehicle Routing Problem with Time Windows, *Computers & Operations Research*, Vol. 34, No. 6, pp. 1561-1584, June, 2007.

[3] Y. Chen, X. Zhou, Runtime Analysis of a Multi-objective Evolutionary Algorithm for Obtaining Finite Approximations of Pareto Fronts, *Information Sciences*, Vol. 262, pp. 62-77, March, 2014.

[4] X. He, L. Ma, Quantum-inspired Ant Colony Algorithm for Vehicle Routing Problem with Time Windows, *Systems Engineering-Theory & Practice*, Vol. 33, No. 5, pp. 1255-1261, May, 2013.

[5] Y. Chi, Multi-objective Optimization Based on Ant Colony Algorithm, *Computer Engineering*, Vol. 35, No. 15, pp. 168-169, August, 2009.

[6] X. Zhang, Y. Tian, R. Cheng, Y. Jin, An Efficient Approach to Non-dominated Sorting for Evolutionary Multi-objective Optimization, *IEEE Transactions on Evolutionary Computation*, Vol. 19, No. 2, pp. 201-213, April, 2015.

[7] T.-T. Nguyen, T.-K. Dao, H.-Y. Kao, M.-F. Horng, C. S. Shieh, Hybrid Particle Swarm Optimization with Artificial Bee Colony Optimization for Topology Control Scheme in Wireless Sensor Networks, *Journal of Internet Technology*, Vol. 18, No. 4, pp. 743-752, July, 2017.

[8] Y.-S. Yen, H.-C. Chao, R.-S. Chang, A. Vasilakos, Flooding-limited and Multi-constrained QoS Multicast Routing Based on the Genetic Algorithm for MANETs, *Mathematical & Computer Modelling An International Journal*, Vol. 53, No. 11-12, pp. 2238-2250, June, 2011.

[9] M. Polacek. R. F. Hartl, K. Doerner, M. Reimann, A Variable Neighborhood Search for the Multi-depot Vehicle Routing Problem with Time Windows, *Journal of Heuristics*, Vol. 10, No. 6, pp. 613-627, December, 2004.

[10] E. Cao, M.-Y. Lai, K. Nie, Research on Improved Differential Evolution Algorithm for Vehicle Routing Problem with Time Windows, *Journal of system simulation*, Vol. 21, No. 8, pp. 2420-2423, 7, 2009.

[11] Q. Zhang, H. Li, MOEA/D: A multiobjective Evolutionary Algorithm Based on Decomposition, *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 6, pp. 712-731, December, 2007.

[12] B. Ombuki, B. J. Ross, F. Hanshar, Multi-objective Genetic Algorithms for Vehicle Routing Problem with Time Windows, *Applied Intelligence*, Vol. 24, No. 1, pp. 17-30, February, 2006.

[13] K. Ghoseiri, S. F. Ghannadpour, Multi-objective Vehicle Routing Problem with Time Windows Using Goal Programming and Genetic Algorithm, *Applied Soft Computing*, Vol. 10, No. 4, pp. 1096-1107, September, 2010.

[14] R. Banos, J. Ortega, C. Gil, A. Fernandez, F. Toro, A Simulated Annealing-based Parallel Multi-objective Approach to Vehicle Routing Problem with Time Windows, *Expert Systems with Applications*, Vol. 40, No, 5, pp. 1696-1707, April, 2013.

[15] R. Banos, J. Ortega, C. Gil, A. L. Marquez, F. de Toro, A Hybrid Meta-heuristic for Multi-objective Vehicle Routing Problems with Time Windows, *Computers & Industrial Engineering*, Vol. 65, No. 2, pp. 286-296, June, 2013.

[16] J. Tang, Z. Pan, R. Y. K. Fung, H. Lau, Vehicle Routing Problem with Fuzzy Time Windows, *Fuzzy Sets & Systems*, Vol. 160, No. 5, pp. 683-695, March, 2009.

[17] O. Braysy, G. Hasle, W. Dullaert, A Multi-start Local Search Algorithm for the Vehicle Routing Problem with Time Windows, *European Journal of Operational Research*, Vol. 159, No, 3, pp. 586-605, December, 2004.

[18] J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, G. Vitry, Multi-start Iterated Local Search for the Periodic Vehicle Routing Problem with Time Windows and Time Spread Constraints on Services, *Computers & Operations Research*, Vol. 41, pp. 196-207, January, 2014.

[19] A. Lim, X. Zhang, A Two-stage Heuristic with Ejection Pools and Generalized Ejection Chains for the Vehicle Routing Problem with Time Windows, *Informs Journal on Computing*, Vol. 19, No. 3, pp. 443-457, July, 2007.

[20] K. Deb, H. Jain, An Evolutionary Many-objective Optimization Algorithm Using Reference-point-based Nondominated Sorting Approach, Part I: Solving Problems with Box Constraints, *IEEE Transactions on Evolutionary Computation*, Vol. 18, No. 4, pp. 577-601, August, 2014.

[21] S. F. Ghannadpour, S. Noori, R. Tavakkoli-Moghddam, A Multi-objective Vehicle Routing and Scheduling Problem with Uncertainty in Customers' Request and Priority, *Journal of Combinatorial Optimization*, Vol. 28, No. 2, pp. 414-446, August, 2014.

[22] J. Deng, L. Wang, A Competitive Memetic Algorithm for Multi-objective Distributed Permutation Flow Shop Scheduling Problem, *Swarm and Evolutionary Computation*, Vol. 32, pp. 121-131, February, 2017.

[23] W.-N. Chen, J. Zhang, H. S. H. Chung, W.-L. Zhong, W.-G. Wu, Y.-H. Shi, A Novel Set-based Particle Swarm Optimization Method for Discrete Optimization Problems, *IEEE Transactions on Evolutionary Computation: A Publication of the IEEE Neural Networks Council*, Vol. 14, No. 2, pp. 278-300, April, 2010.

[24] M. Li, S. Yang, X. Liu, Bi-goal Evolution for Many-objective Optimization Problems, *Artificial Intelligence*, Vol. 228, No. C, pp. 45-65, November, 2015.

[25] M. Samorani, M. Laguna, Data-mining-driven Neighborhood Search, *Informs Journal on Computing*, Vol. 24, No. 2, pp. 210-227, May, 2011.

[26] K. C. Tan, Y. H. Chew, L. H. Lee, A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem with Time Windows, *Computational Optimization and Applications*, Vol. 34, No. 1, pp. 115-151, May, 2006.

[27] W. Hu, G. G. Yen, X. Zhang, Multiobjective Particle Swarm Optimization Based on Pareto Entropy, *Journal of Software*,

Vol. 25, No. 5, pp. 1025-1050, May, 2014.

[28] C.-W. Xie, X.-F. Zou, X.-W. Xia, Z.-J. Wang, A Multi-Objective Particle Swarm Optimization Algorithm Integrating Multiply Strategies, *Acta Electronica Sinica*, Vol. 43, No. 8, pp. 1538-1544, August, 2015.

[29] S. P. Kapse, S. Krishnapillai, An Improved Multi-objective Particle Swarm Optimization Algorithm Based on Adaptive Local Search, *International Journal of Applied Evolutionary Computation*, Vol. 8, No. 2, pp. 1-29, April, 2017.

[30] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182-197, April, 2002.

[31] K. N. Krishnanand, D. Ghose, Glowworm Swam Optimisation: A New Method for Optimizing Multi-modal Functions, *International Journal of Computational Intelligence Studies*, Vol. 1, No. 1, pp. 93-119, May, 2009.

[32] K. N. Krishnanand, D. Ghose, Glowworm Swarm Optimization for Simultaneous Capture of Multiple Local Optima of Multimodal Functions, *Swarm Intelligence*, Vol. 3, No. 2, pp. 87-124, June, 2009.

[33] K. N. Krishnanand, D. Ghose, Glowworm Swarm Based Optimization Algorithm for Multimodal Functions with Collective Robotics Applications, *Multiagent and Grid Systems*, Vol. 2, No. 3, pp. 209-222, September, 2006.

[34] W. Dong, K. Zhou, Q. Fu, Y. Duan, The Discrete Glowworm Swarm Optimization Algorithm with an Adaptive Neighborhood Search, *Journal of Computational & Theoretical Nanoscience*, Vol. 13, No. 6, pp. 3824-3834, June, 2016.

[35] B. Crevier, J. Cordeau, G. Laporte, The Multi-depot Vehicle Routing Problem with Inter-depot Routes, *European Journal of Operational Research*, Vol. 176, No. 2, pp. 756-773, January, 2007.

[36] http://web.cba.neu.edu/~msolomon/problems.htm.

[37] Y.-Q. Zhou, Z.-X. Huang, H.-X. Liu, Discrete Glowworm Swarm Optimization Algorithm for TSP Problem, *Acta Electronica Sinica (in Chinese)*, Vol. 40, No. 6, pp. 1164-1170, June, 2012.

Biographies

Wenbo Dong born in 1992, master. His research interests include evolutionary algorithm and optimization algorithm.

Kang Zhou born in 1965, professor, Ph.D. His research interests include DNA Computing, Intelligent Algorithm, Data Mining, Operations Research.

Gexiang Zhang born in 1974, Ph.D., professor, Ph.D. supervisor. His research interests include natural computing, pattern recognition and signal processing, etc.

Han-Chieh Chao received the MS and Ph.D degrees in electrical engineering from Purdue University in 1989 and 1993, respectively. He is a professor of the Department of Electrical Engineering, National Dong Hwa University, where he also serves as the president. He is also with the Department of Computer Science and Information Engineering, National Ilan University, School of Mathematics and Computer Science, Wuhan Polytechnic University, and School of Information Science and Engineering, Fujian University of Technology. His research interests include high-speed networks, wireless networks, and IPv6-based networks and applications.