IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 5, 2018, accepted December 21, 2018, date of current version February 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894681

Brainstorming-Based Ant Colony Optimization for Vehicle Routing With Soft Time Windows

LIBING WU[1,2], (Member, IEEE), ZHIJUAN HE[1], YANJIAO CHEN[1], DAN WU[3], AND JIANQUN CUI[4]

[1]School of Computer Science, Wuhan University, Wuhan 430072, China
[2]Hubei Key Laboratory of Transportation Internet of Things, Wuhan University of Technology, Wuhan 430070, China
[3]School of Computer Science, University of Windsor, Windsor, ON N9B 3P4, Canada
[4]School of Computer Science, Central China Normal University, Wuhan 430079, China

Corresponding author: Yanjiao Chen (chenyj.thu@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772377, Grant 61672257, and Grant 61702380, in part by the Natural Science Foundation of Hubei Province of China under Grant 2017CFA007 and Grant 2017CFB134, in part by the Science and Technology Planning Project of Shenzhen under Grant JCYJ20170818112550194, in part by the Fund of Hubei Key Laboratory of Transportation Internet of Things under Grant WHUTIOT-2017A0011, in part by the Technological Innovation Projects of Hubei Province under Grant 2017AAA125, and in part by the Equipment Pre-Research Joint Fund of Ministry of Education of China (Youth Talent) under Grant 6141A02033327.

ABSTRACT In this paper, we propose a novel ant colony optimization algorithm based on improved brainstorm optimization (IBSO-ACO) to solve the vehicle routing problem with soft time windows. Compared with the traditional ant colony algorithm, the proposed IBSO-ACO can better address the local optimum problem, since we have carefully designed an improved brainstorming optimization algorithm to update the solutions obtained by the ant colony algorithm, which enhance the solution diversity and the global search ability. Furthermore, we use the classification method to accelerate the convergence of the proposed algorithm. The extensive experimental results have confirmed that the proposed IBSO-ACO algorithm can achieve a lower routing cost at a high convergence rate than the traditional ant colony algorithm and the simulated annealing ant colony algorithm.

INDEX TERMS Vehicle routing problem with soft time window, improved brainstorm optimization.

I. INTRODUCTION

In light of achieving efficient logistics, to minimize transportation cost while meeting the time requirement has long been the interest of both industrial and academia communities. Vehicle routing optimization aims at finding a solution that uses a minimum number of vehicles that travel along optimal routing paths to serve clients at their required time windows. There are two kinds of vehicle routing optimization problems with time windows: one is that vehicles arriving in time windows that are not required by customers will be rejected, that is, vehicle routing optimization with hard time windows; the other is that vehicles arriving in time windows that are not required by customers will suffer penalty costs, that is, vehicle routing optimization with soft time windows.

In this paper, we focus on the problem of optimal vehicle routing with soft time windows. That is, finding a solution that uses a minimum number of vehicles along the optimal route to serve the customer at the time window that the customer needs, and if the vehicle arriving later than the required time will suffer penalty costs. Existing works in this area mostly leverage the ant colony optimization (ACO) algorithm [1]–[4], particle swarm optimization [5], and so on [6]. However, the traditional ACO algorithm may easily be trapped in local optimum due to a limited solution space [7], [8]. Several alternative algorithms have been proposed to address the local optimum problem. In [9], a simulated annealing ant colony algorithm (SAACO) is designed to conduct path planning for vehicle routing problems. In [10], an improved ant colony algorithm with a new pheromone update rule is proposed. In [11], a hybrid particle swarm optimization is proposed for emergency relief vehicle routing problems. Although these algorithms partially resolve the problem of local optimum, they mainly focus on the vehicle routing problem with hard time windows but not soft time windows.

The associate editor coordinating the review of this manuscript and approving it for publication was Yin Zhang.

In addition, in [12], a large-scale domain search is done for the vehicle routing optimization problem with soft time windows. In [13], scholars try to solve the vehicle routing optimization problem with soft time windows by using a unified tabu search algorithm. Later, simulated annealing ant colony algorithm (SAACO) appeared to solve the problem that ant colony algorithm is easy to fall into local optimum to a certain extent. The corresponding problem model of the Mouthuy's VNS algorithm is different from that of this paper. And we can know that SAACO algorithm is better than tabu search algorithm through literature from [14]. So we choose the SAACO algorithm to do the contrast experiment.

There are many related works. In [15], scholars have done some research on edge cloud communication. In the aspect of vehicle network, [16] and [17] has done some research on vehicle network communication, [18] and [19] has paid attention to the safety of vehicle network, and some big data analytics of cloud/IoT, such as [20]. In addition, [21] has discussed vehicle network data scheduling.

In this paper, we propose a novel ant colony optimization algorithm based on improved brainstorm optimization (referred to as IBSO-ACO), which enriches the solution space to achieve the global optimum for vehicle routing with soft time windows. We believe that this is the first work to leverage the brainstorm optimization to address the defect of local optimum for the ant colony algorithm. Our carefully-designed improved brainstorm optimization algorithm randomly generates and replaces the solutions obtained by the ant colony algorithm under certain conditions, then the pheromones for the ant colony algorithm are updated to provide a positive feedback for the next round of iteration.

In the experimental part, In the experimental part, we choose the SAACO algorithm, which is a representative of the improved ACO algorithm. We compare IBSO-ACO algorithm with a representative SAACO algorithm and traditional ACO algorithm. Finally, the proposed algorithm will converge to the approximate global optimal solution(because of combinatorial optimization) to the vehicle routing problem with soft time windows, and the result is better than which SAACO algorithm and the traditional ACO algorithm get.

Our contributions in this paper are as follows:

- An improved brainstorm algorithm is first designed to tackle the problem of local optimum in the traditional ant colony algorithm for vehicle routing problems with soft time windows.
- A novel improved brainstorm optimization algorithm is designed to enrich the diversity of solutions of the ant colony algorithm. The proposed iBSO-ACO algorithm can improve the shortcomings of the traditional ACO algorithm which is easy to fall into the local optimal solution.
- Extensive experiments have been conducted to demonstrate that our proposed algorithm outperforms benchmark algorithm in reducing the transportation cost at a higher convergence rate.

II. SYSTEM MODEL

In the vehicle routing problem (VRP) with time windows, if the vehicle does not arrive within the required time of the client, there are two options: 1) direct rejection, i.e., the VRP problem with hard time windows [22]–[25]; 2) adding penalties in the objective function, i.e., the VRP problem with soft time windows [26]–[29]. Note that VRP problems with soft time windows can be transformed into the one with hard time windows by setting service rejection as the penalty. Therefore, in this paper, we only consider VRP problems with soft time windows (referred to as VRPS).

We consider a transportation system consisting of N clients. Let $G = (V, E)$ [30] denote the transportation map, in which: 1) V is the set of all nodes including the distribution center node and the client nodes; in particular, v_0 represents the distribution center(only have one distribution center in this model), and v_i, $i \in [1, N]$ represents client; 2) E is the set of edges, denoting the section of roads that the vehicles travel. Let c_{ij} denote the total cost of traveling the route between v_i and v_j, and d_i denote the demand of client v_i. Let Q_k be the maximum load that vehicle k can bear. We use $x_{ijk} = 1$ to denote that vehicle k has visited the route between v_i and v_j, otherwise $x_{ijk} = 0$. The earliest and the latest time that client v_i requires the vehicle to arrive are t_i^e and t_i^l respectively. Let t_{ik} denote the time that vehicle k arrives at client v_i. If $t_{ik} < t_i^e$, the waiting cost per unit time is c_w. If $t_{ik} > t_i^l$, the penalty cost per unit time is c_p.

Our objective is to derive the route and the number of required vehicles (denoted by K) to serve all clients with the lowest total cost of transportation, which includes not only the cost of traveling and vehicle scheduling(which is proportional to the number of vehicles), but also the penalty cost. The objective function of our VRPS problem is:

$$\begin{aligned} \min \; & \sum_{k=1}^{K}\sum_{i=0}^{N}\sum_{j=0}^{N} c_{ij}x_{ijk} \\ & + c_w \sum_{i=1}^{N}\sum_{k=1}^{K}(\sum_{j=1}^{N} x_{ijk}) \times \max\{(t_i^e - t_{ik}), 0\} \\ & + c_p \sum_{i=1}^{N}\sum_{k=1}^{K}(\sum_{j=1}^{N} x_{ijk}) \times \max\{(t_{ik} - t_i^l), 0\}, \end{aligned} \tag{1}$$

in which c_{ij} is computed as:

$$c_{ij} = \begin{cases} c_{0k}K + c^s s_{ij}, & i = 0, \\ c^s s_{ij}, & i \neq 0, \end{cases} \tag{2}$$

in which c_{0k} is the scheduling cost at the distribution center; K is the number of vehicles required; c^s is the travel cost per unit distance; s_{ij} is the distance from client v_i to client v_j. Specifically, when $i = 0$, the more vehicles you need, the greater the total cost. We jointly consider minimizing the number of vehicles and the travelling distance of all vehicles.

Our objective function considers the cost of vehicle scheduling (which is related to the number of vehicles), the distance between vehicles (the distance after the vehicle is launched), and the time penalty cost (which is related to the early arrival time and the late arrival time). It is a multi-objective problem and minimizes the number of vehicles, the distance between vehicles and the time penalty cost.

The constraint [31] of our VRPS problem is:

$$\sum_{i=1}^{N} x_{i0k} = 1, \quad \forall i = 1, \cdots, N; \tag{3}$$

$$\sum_{i=0}^{N}\sum_{k=1}^{K} x_{ijk} = 1, \quad \forall j = 1, \cdots, N; \tag{4}$$

$$\sum_{j=0}^{N}\sum_{k=1}^{K} x_{ijk} = 1, \quad \forall i = 1, \cdots, N; \tag{5}$$

$$\sum_{i=0}^{N} d_i(\sum_{j=0}^{N} x_{ijk}) \le Q_k, \quad \forall k = 1, \cdots, K; \tag{6}$$

$$t_{ik} \ge 0, \quad \forall i = 1, \cdots, N, \ \forall k = 1, \cdots, K. \tag{7}$$

Constraint (3) means that each vehicle has to leave the distribution center v_0 first, and return to the distribution center v_0 after the service is completed. Constraints (4) and (5) restrain that each client is served by exactly one vehicle; Constraint (6) means that the load of a vehicle should be less than its maximum load capacity. Constraints (7) show that the arriving time should be non-negative.

III. IBSO-ACO ALGORITHM

In this section, we first introduce the traditional ACO algorithm. Then, we propose an improved brainstorming optimization (IBSO) algorithm to update the solutions obtained from the ACO algorithm, termed as IBSO-ACO. Finally, we demonstrate how to update the parameters in the IBSO-ACO algorithm to form a positive feedback loop.

A. THE BASIC ACO ALGORITHM

When ants try to find food, they leave pheromones on the paths that they have traveled. An ant will select the path to food based on the quantity of pheromones. The more pheromones left on a path, the more likely an ant chooses this path, thus more ants will gather on the path with the most pheromones to find the optimal path. Pheromones will volatilize over time. As shown in Fig. 1, node F is the anthill; node N is the food location; A, B, C are the intermediary location nodes; the solid triangles represent ants; and FC, FA, AC, CN, CB, BN are the paths that ants can. Consider a group of ants that travel from F to N for food. Initially, ants are randomly distributed on each path, as shown in the first stage. Then, more and more ants come to path $F \to C \to N$, as shown in the second stage. Finally, almost all ants gather on path $F \to C \to N$, as shown in the third stage, while nearly no ants stay on other paths.

Suppose there are N clients, and the pheromone on each edge(the line between the two clients is called edge) is initiated as 0, the collection of all edges that ants travel through is called a path. In ACO algorithm, assuming that there

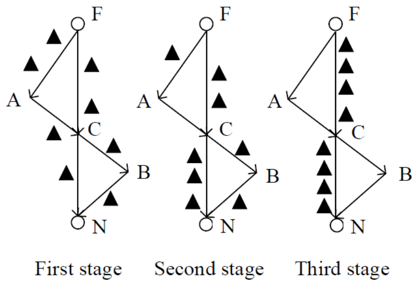

FIGURE 1. An illustrative example of the basic ACO algorithm.

are T iterations, pheromones are updated once each iteration path is completed. At round τ, the probability of ant m traveling from node i to node j can be represented as:

$$P_{ij}{}^{m}(\tau) = \begin{cases} \dfrac{\gamma_{ij}{}^{\alpha}(\tau)\eta_{ij}{}^{\beta}}{\sum_{j\in\mathcal{B}_m} \gamma_{ij}{}^{\alpha}(\tau)\eta_{ij}{}^{\beta}}, & \text{if } j \in \mathcal{B}_m, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

in which $\mathcal{B}_m$ is the set of nodes that ant m has not visited and can choose as the next node with a certain probability; $\gamma_{ij}(\tau)$ is the quantity of pheromones on edge e_{ij} between node i and node j at round τ; α is the influence of pheromones on the ant colony search; $\eta_{ij} = 1/s_{ij}$ is the visibility of edge e_{ij}, which is the inverse of the distance s_{ij}; β [32] is the influence of visibility on the ant colony search, and the greater β is, the more likely that ants will take the highly-visible path.

To begin with, all ants start from the anthill with a load of zero. Then, each ant iteratively selects the next node according to equation (8) until there is no node that has not been visited, i.e., $\mathcal{B}_m$ is empty. Upon choosing a candidate node according to equation (8), the ant will add the load of the node to its current load. If the resulting load is less than ant m's load limit, ant m will move to that node and remove it from the non-visiting set $\mathcal{B}_m$; otherwise, ant m will return to the anthill, where its load is reset to zero and it will continue to select the next node. The algorithm terminates when every ant has completed a round of search (the non-visiting set $\mathcal{B}_m$ for any ant is empty).

The ACO algorithm can be used to help solve the VRPS problem by using a cluster of ants(a total of M ants) to generate the routing solution. Each ant starts with a load of zero, and a load capacity of Q_1. The node selection sequence of the ant forms the routing path of the first vehicle. The load of the node is the demand of the client. Whenever the ant returns to the anthill due to overload, we deem that a new vehicle is required. The load of the ant is reset to zero, and the load capacity is set as that of the second vehicle. Such process continues until all nodes are covered by the ant, i.e., all client nodes are served by vehicles. The number of required vehicles equals the number of times that the ant returns to the anthill, and their corresponding paths can be obtained. With M ants running the ACO algorithm for one round of search, we can get M solutions for the VRPS problem.

Algorithm 1 The IBSO-ACO Algorithm

```
Require: Maximum number of iterations $T$, the transportation map $(V, E)$, the number of ants $M$.
Ensure: The optimal routing path.
1: if $\tau \leq T$ then
2:     for all ant $m \in [1, M]$ do
3:         The load $l_m = 0$;
4:         The non-visiting set $\mathcal{B}_m = V$;
5:     end for
6:     for all Ant $m \in [1, M]$ do
7:         $k = 1$.
8:         while $|\mathcal{B}_m| > 0$ do
9:             Select a node $v_i$ according to equation (8).
10:            Update the load $l_m = l_m + d_i$.
11:            if $l_m \leq Q_k$ then
12:                The next node is $v_i$.
13:                Remove node $v_i$ from the non-visiting set $\mathcal{B}_m$.
14:            else
15:                Return to $v_0$.
16:                $k = k + 1$.
17:                $l_m = 0$.
18:            end if
19:        end while
20:    end for
21:    The solution set of one round of search is $\{(S_m, C_m)\}_{m\in[1,M]}$.
22:    Update the solution set using the IBSO algorithm.
23:    Update the pheromones of all edges using the traditional ACO algorithm(according to equation (8)).
24: end if
```

B. UPDATE ACO SOLUTIONS WITH IBSO ALGORITHM

Although the traditional ACO algorithm has many nice properties, it usually suffers from the local optimum problem. Therefore, we propose an improved brainstorming optimization algorithm to enrich the solutions obtained by the ACO algorithm.

After running the ACO algorithm to complete one round of search for the VRPS problem, we can obtain M solutions. The solution of ant m is denoted as (S_m, C_m), in which S_m is the overall routing path, and C_m is the total cost. We design an improved BSO algorithm (IBSO) to update the solution set obtained by the ACO algorithm. The BSO algorithm [33] is a heuristic intelligent optimization algorithm based on the idea of brainstorming. The basic BSO algorithm works as follows:

- Randomly generate M solutions. Set the thresholds p_1, p_2, p_3, p_4.
- Divide the M solutions into several clusters using the k-means clustering algorithm. The center of each cluster is the optimal solution within the cluster;
- Randomly select a cluster, and generate a random number $r_1 \in (0, 1)$. If $r_1 < p_1$, generate a random solution to replace the center of the cluster;
- Generate a new solution:
  - Generate a random number $r_2 \in (0, 1)$. If $r_2 < p_2$, randomly select a cluster, then generate a random number $r_3 \in (0, 1)$. If $r_3 < p_3$, generate a new solution by adding random values to the center of the cluster; otherwise, randomly select a non-center solution from this cluster and add a random value to the solution to generate a new solution. If $r_2 > p_2$, randomly select two clusters, and produce a random number $r_4 \in (0, 1)$. If $r_4 < p_4$, combine the centers of the two clusters and add a random value to generate a new solution; otherwise, combine two randomly-selected non-center solutions of the two clusters and add a random value to generate a new solution.
  - The newly generated solution is compared with the original solution with the same index, and the better one is kept.

We leverage the clustering and randomization rationale of the basic BSO algorithm to achieve solution diversity and to avoid local optimum of the ACO algorithm. The initial solution set of all vehicles after ACO is $\{(S_m, C_m)\}_{m\in[1,M]}$. Our proposed IBSO algorithm updates the solution set as follows:

- Divide the solution set into cluster $\mathcal{A}$ and cluster $\mathcal{B}$. Cluster $\mathcal{A}$ contains all solutions with a cost lower than $(C_{\min} + C_{\max})/2$, and cluster $\mathcal{B}$ contains the remaining solutions. The optimal solution of cluster $\mathcal{A}$ is $(S^a_{\min}, C^a_{\min})$, which is also recorded as its cluster center, a represents cluster $\mathcal{A}$. The optimal solution of cluster $\mathcal{B}$ is $(S^b_{\min}, C^b_{\min})$, which is also recorded as its cluster center, b represents cluster $\mathcal{B}$.
- For cluster $\mathcal{A}$, replace each solution by a randomly-generated solution.
- For cluster $\mathcal{B}$, randomly generate a solution (S^{temp}, C^{temp}), then generate a random number $r \in (0, 1)$. In this algorithm, we set a probability p. If $r < p$, replace the cluster center $(S^b_{\min}, C^b_{\min})$ with (S^{temp}, C^{temp}) if $C^{temp} < C^b_{\min}$. If $r \geq p$, randomly select and replace a non-center solution (S^b_k, C^b_k) with (S^{temp}, C^{temp}) if $C^{temp} < C^b_k$. It should be noted that p is a probabilistic parameter. We have done a lot of work in choosing the parameters of probabilistic algorithm before choosing the value of p. We have chosen the parameter p which is relatively suitable for solving VRPS problem, so as to maximize the performance of IBSO-ACO algorithm.
- We regenerate the new solution by using the basic ACO algorithm.

For cluster $\mathcal{A}$, during the update, new solutions directly replace previous solutions. Since each initial solution in cluster $\mathcal{A}$ has a cost lower than $(C_{\min} + C_{max})/2$, the difference between solutions after updating is relatively large, which helps maintain the diversity and avoid local optimum to a certain extent. For cluster $\mathcal{B}$, a randomly-generated new solution will be accepted with a probability only if the cost of the new solution is smaller than that of the solution to be

TABLE 1. Performance comparison on standard experiments.

Optimization algorithm		Data set					
		R1-01	C1-01	RC1-03	R2-01	C2-01	RC2-05
IBSO-ACO	Average number of iterations for convergence	74	89	87	72	64	38
	Optimal cost	86892.40	58962.41	77535.66	70304.23	52813.15	69994.33
	Average optimal cost of 10 runs	90889.19	60330.68	81125.98	70974.80	56611.85	71455.90
SAACO	Average number of iterations for convergence	75	106	100	77	87	98
	Optimal cost	89424.88	59249.80	78370.46	70966.35	56895.82	71146.45
	Average optimal cost of 10 runs	91895.03	60377.75	82039.45	71794.42	57736.11	71576.54
ACO	Average number of iterations for convergence	107	97	89	96	73	65
	Optimal cost	91894.02	59724.20	80100.75	71540.12	57041.51	71491.08
	Average optimal cost of 10 runs	92928.94	61367.62	82184.05	71873.70	58033.05	72334.27

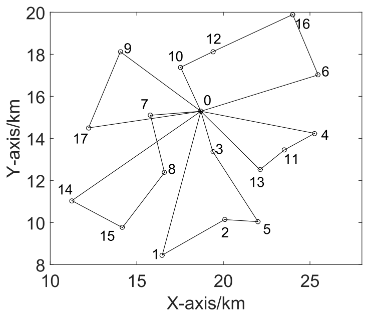

FIGURE 2. IBSO-ACO algorithm optimal solution.

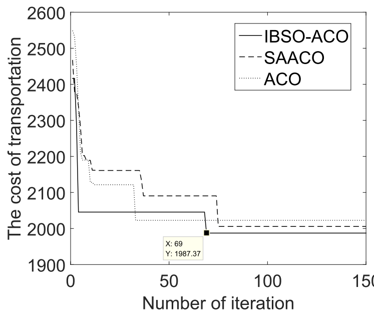

FIGURE 3. Performance comparison on small-scale experiment.

replaced (center or non-center). Since each initial solution in cluster $\mathcal{B}$ has a cost greater than $(C_{\min} + C_{max})/2$, it is more likely that high-cost solutions will be replaced by low-cost solutions. In this way, the proposed IBSO-ACO algorithm can improve the convergence rate while maintaining the diversity of solutions.

C. PHEROMONE UPDATE

After the solution set obtained by the ACO is updated by the IBSO algorithm, the pheromones need to be updated for the next iteration to give a positive feedback. The pheromone of round τ is $\gamma_{ij}(\tau)$, and we have the update rule as:

$$\gamma_{ij}(\tau + 1) = (1 - \rho)\gamma_{ij}(\tau) + \sum_{m=1}^{M} \Delta\gamma_{ij}^{m}(\tau), \quad (9)$$

in which ρ is the parameter for pheromone evaporation; $\Delta\gamma_{ij}{}^{k}(\tau)$ is the quantity of pheromones generated by vehicle (ant) k on edge e_{ij} at round τ. If vehicle k passes edge e_{ij} in round τ, $\Delta\gamma_{ij}{}^{k}(\tau) = P/L_k$, otherwise $\Delta\gamma_{ij}{}^{k}(\tau) = 0$. The constant P represents the total amount of information, whose value affects the positive feedback of the algorithm and thus affects the convergence rate of the algorithm. L_k represents the total length of path that ant m has traveled in this iteration. $\Delta\gamma_{ij}(0)$ are initiated as 0 (in order to better highlight the performance of IBSO-ACO algorithm, we set the initial pheromone to 0).

The updated solution set at iteration τ contains both the optimal solution of the ACO algorithm at iteration τ and the random solutions generated by the IBSO algorithm, which enriches the diversity of pheromone distribution. The more pheromones a path has, the greater the probability that each vehicle will choose this path in the next iteration. Since pheromones also exist on a small number of randomly-generated paths, they may be chosen by vehicles in the next iteration as well, which helps prevent local optimum. We will run ACO followed by IBSO for multiple iterations until the optimal cost converges or until the maximum number of iterations is hit. We summarize the IBSO-ACO algorithm in Alg. 1.

IV. EXPERIMENTS

We implement the proposed IBSO-ACO algorithm using MATLAB, running on a desktop with Intel Core i5-3230M, CPU 2.6GHz. Experiments are conducted on a small-scale 17 client nodes and on the standard test data of Solomon (100 client nodes) [34], respectively. We choose Solomon for our experiment because it is the most classic data set and it makes experiment more convincing, in this study, we selected a representative data set from each of the six types of Solomon data sets and carried out experiments on these six data sets. We compare the performance of the proposed IBSO-ACO algorithm with that of the basic ant colony algorithm (ACO) and the simulated annealing ant colony algorithm (SAACO). The parameters for the ACO algorithm are: $\alpha = 1, \beta = 5$, $\rho = 0.75, H = 100, M = 40, T = 150$. The parameters of SAACO are: the range of the temperature $\theta \in [\theta_{min}, \theta_{max}]$, in which the initial temperature of the system is $\theta_{\max} = 20$, the lowest temperature is $\theta_{\min} = 1$, the cooling coefficient is $r = 0.85$. The parameters of IBSO-ACO are: $p = 0.005$, the other parameters are the same as those of the ACO algorithm.

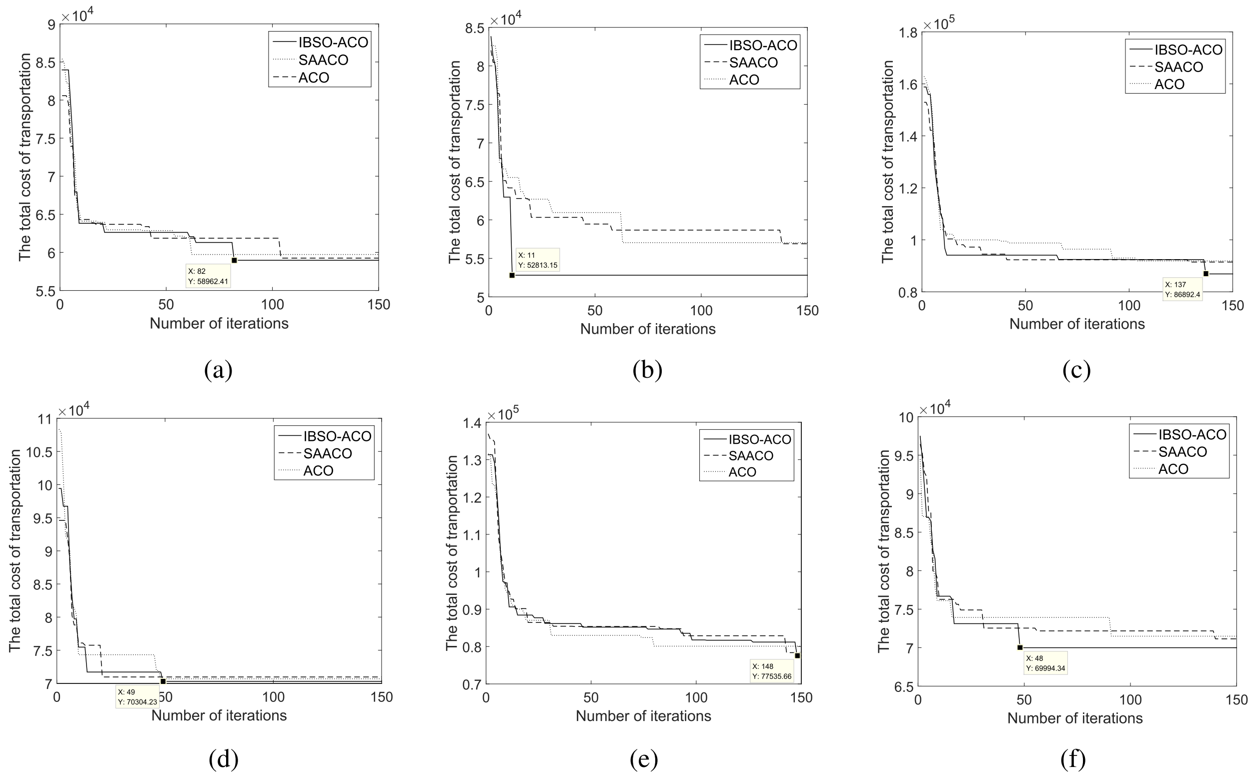

FIGURE 4. Convergence comparison. (a) C1-01. (b) C2-01. (c) R1-01. (d) R2-01. (e) RC1-03. (f) RC2-05.

TABLE 2. The client information.

Client	Coordinates	d_i	$[t^e, t^l]$	Service time
0	(18.70, 15.29)	-	-	-
1	(18.70,15.29)	0	[0,1000]	0
2	(16.47,8.45)	3.0	[5.0,19.5]	1.8
3	(20.07,10.14)	2.5	[4.1,14.9]	1.0
4	(19.39,13.37)	5.5	[1.5,10.9]	2.3
5	(25.27,14.24)	3.0	[6.0,19.6]	1.8
6	(22.00,10.04)	1.5	[6.5,14.8]	1.2
7	(25.47,17.02)	4.0	[9.9,19.8]	2.4
8	(15.79,15.10)	2.5	[10.1,24.2]	1.5
9	(16.60,12.38)	3.0	[15.5,27.1]	1.8
10	(14.05,18.12)	2.0	[2.5,19.9]	1.2
11	(17.53,17.38)	2.5	[3.2,15.6]	1.5
12	(23.52,13.45)	3.5	[2.0,20.3]	2.1
13	(19.41,18.13)	3.0	[2.3,15.5]	1.8
14	(22.11,12.51)	5.0	[15.2,24.5]	2.0
15	(11.25,11.04)	4.5	[11.1,28.8]	2.7
16	(14.17,9.76)	2.0	[12.6,24.0]	1.3
17	(24.00,19.89)	3.5	[8.4,18.5]	1.2
18	(12.21,14.50)	4.0	[10.1,23.2]	1.5

A. SMALL-SCALE EXPERIMENTS

The information of the 17 client nodes is shown in Table 2. Fig. 2 demonstrates the optimal paths obtained by the proposed IBSO-ACO algorithm: vehicle 1: 0 → 10 → 12 → 16 → 6 → 0; vehicle 2: 0 → 3 → 5 → 2 → 1 → 0;vehicle 3: ÂŽ0 → 13 → 11 → 4 → 0; vehicle 4: 0 → 7 → 8 → 15 → 14 → 0; vehicle 5: 0 → 9 → 17 → 0.

TABLE 3. Performance comparison on small-scale experiments.

	Convergence	Optimal cost	Average optimal cost after 20 runs
ACO	74	2022.77	2061.96
SAACO	87	2005.70	2059.12
IBSO-ACO	59	1987.37	2049.25

Fig. 3 compares the convergence rate and the optimal solution of the three algorithms. It is shown that the cost of the routing solution obtained by the IBSO-ACO algorithm is the smallest. The IBSO-ACO algorithm converges at iteration 69 with an optimal cost of 1987.37, which is 18.33 less than the optimal cost of SAACO, and 35.40 less than the optimal cost of ACO.

We run the three algorithms each 10 times for each data set, and show the average results in Table 3. The average number of iterations for IBSO-ACO to converge is 59, which is lower than that of ACO and SAACO, which confirms that IBSO-ACO has a fast convergence rate. The average optimal cost of IBSO-ACO after 20 runs is 2049.25, lower than that of SAACO (2059.12) and ACO (2061.96), which indicates that IBSO-ACO can yield relatively stable results within limited iterations. The experiment results have verified that IBSO-ACO has improved cost, convergence rate and stability.

B. STANDARD EXPERIMENT

We have chosen six datasets with different client distributions from the Solomon test data, that is, R1-01,

TABLE 4. The cost reduction of IBSO-ACO.

Data set	Cost reduction compared with SAACO(%)	Cost reduction compared with ACO(%)
R1-01	2.83	5.38
C1-01	4.85	1.27
RC1-03	1.06	3.20
R2-01	0.93	1.72
C2-01	7.17	7.14
RC2-05	1.61	2.09

C1-01, RC1-03, R2-01, C2-01 and RC2-05 from the test data set. We know that according to the distribution of points, the whole Solomon data set can be divided into six types, each of which has ten similar data sets, totaling 60. Solomon has six types of data sets, we select a representative data set from each kind of Solomon data set, and carry out experiments on these six different types of data sets.So we use six representative data sets to represent the results of 60 data sets.

We run the three algorithms for 10 times, and show the average results in Table 1. The average number of iterations for IBSO-ACO to converge is faster than that for SAACO and ACO. Compared with SAACO and ACO, IBSO-ACO can achieve a better optimal solution for all datasets because IBSO-ACO has a higher global search capability. Furthermore, the average optimal solution of IBSO-ACO after 10 runs is the closest to the final optimal solution, which indicates that IBSO-ACO can potentially achieve a more stable results than SAACO and ACO. Table 4 shows the cost reduction of IBSO-ACO relative to SAACO and ACO. It confirms that IBSO-ACO can greatly decrease the cost compared with SAACO and ACO, especially for clustering type C examples C1-01 and C2-01.

Fig. 4 compares the convergence rate of the three algorithms. For each dataset, the optimal solution obtained by IBSO-ACO is superior to those obtained by the other two algorithms. For C2-01 and RC2-05, the convergence rate of IBSO-ACO is much faster than the other two algorithms; while for R1-01, R2-01 and RC1-03, the convergence rate of IBSO-ACO is slower than the other two algorithms. This indicates that IBSO-ACO is more suitable for solving VRPS problems where the client node is more agglomerated.

Fig. 5 show the optimal solution of each iteration and the vehicle routing result obtained by IBSO-ACO on data set C1-01. As shown in Fig. 5, IBSO-ACO converges with 82 iterations and the optimal cost 58962.41. The required number of vehicles is 10. The optimal routing path obtained by the proposed IBSO-ACO algorithm is shown in Table 5.

Fig. 6 show the optimal solution of each iteration and the vehicle routing result obtained by IBSO-ACO on data set C2-01. As shown in Fig. 6, IBSO-ACO converges with 11 iterations and achieves the optimal cost 52813.15. The required number of vehicles is 10. The optimal routing path obtained by the proposed IBSO-ACO algorithm is shown in Table 6.

Fig. 7 show the optimal solution of each iteration and the vehicle routing result obtained by IBSO-ACO on data set R1-01. As shown in Fig. 7, IBSO-ACO converges with 137 iterations and the optimal cost 86892.40. The required number of vehicles is 8. The optimal routing path obtained by the proposed IBSO-ACO algorithm is shown in Table 7.

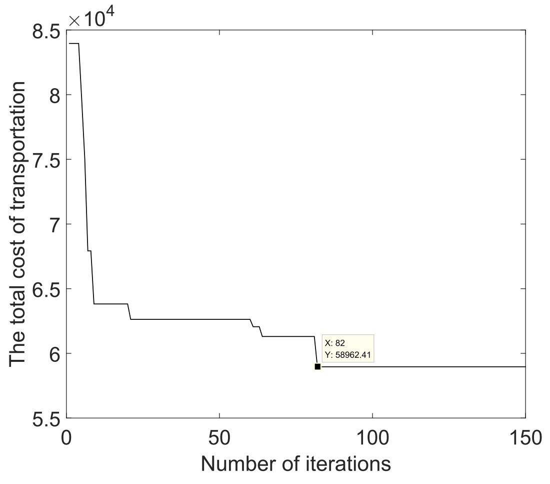

FIGURE 5. Global vs local optimum (C1-01).

TABLE 5. The routing path of IBSO-ACO(C1-01).

Vehicle number	Vehicle routing path
1	$0 \to 69 \to 66 \to 62 \to 61 \to 64 \to 68 \to 72 \to 74 \to 65 \to 63 \to 67 \to 0$
2	$0 \to 47 \to 49 \to 52 \to 50 \to 51 \to 48 \to 45 \to 44 \to 40 \to 41 \to 43 \to 46 \to 59 \to 55 \to 53 \to 0$
3	$0 \to 90 \to 89 \to 84 \to 83 \to 91 \to 85 \to 86 \to 88 \to 87 \to 10 \to 8 \to 11 \to 0$
4	$0 \to 20 \to 23 \to 26 \to 27 \to 25 \to 30 \to 29 \to 24 \to 22 \to 21 \to 28 \to 36 \to 0$
5	$0 \to 98 \to 100 \to 99 \to 96 \to 95 \to 94 \to 92 \to 97 \to 93 \to 0$
6	$0 \to 75 \to 5 \to 7 \to 3 \to 6 \to 9 \to 4 \to 1 \to 2 \to 12 \to 14 \to 0$
7	$0 \to 42 \to 31 \to 35 \to 38 \to 37 \to 39 \to 34 \to 32 \to 0$
8	$0 \to 19 \to 15 \to 13 \to 17 \to 18 \to 16 \to 33 \to 0$
9	$0 \to 80 \to 79 \to 77 \to 73 \to 70 \to 78 \to 76 \to 71 \to 81 \to 82 \to 0$
10	$0 \to 54 \to 56 \to 58 \to 60 \to 57 \to 0$

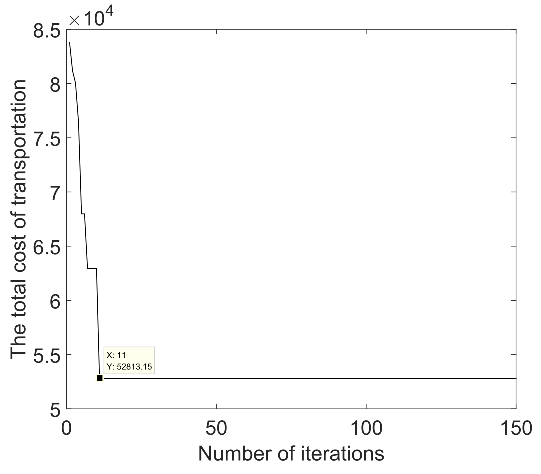

FIGURE 6. Global vs local optimum (C2-01).

TABLE 6. The routing path of IBSO-ACO(C2-01).

Vehicle number	Vehicle routing path
1	$0 \to 48 \to 69 \to 61 \to 64 \to 72 \to 66 \to 62 \to 74 \to 68 \to 65 \to 49 \to 55 \to 59 \to 40 \to 44 \to 0$
2	$0 \to 43 \to 41 \to 45 \to 50 \to 51 \to 52 \to 47 \to 42 \to 46 \to 60 \to 56 \to 58 \to 0$
3	$0 \to 21 \to 24 \to 27 \to 30 \to 29 \to 6 \to 36 \to 35 \to 37 \to 39 \to 31 \to 32 \to 0$
4	$0 \to 88 \to 86 \to 83 \to 77 \to 96 \to 87 \to 90 \to 91 \to 89 \to 4 \to 95 \to 100 \to 92 \to 0$
5	$0 \to 5 \to 75 \to 98 \to 99 \to 1 \to 2 \to 11 \to 9 \to 23 \to 26 \to 28 \to 34 \to 0$
6	$0 \to 20 \to 12 \to 14 \to 19 \to 18 \to 17 \to 13 \to 25 \to 22 \to 0$
7	$0 \to 67 \to 82 \to 84 \to 7 \to 3 \to 94 \to 8 \to 10 \to 63 \to 0$
8	$0 \to 81 \to 76 \to 79 \to 80 \to 73 \to 71 \to 70 \to 78 \to 85 \to 0$
9	$0 \to 93 \to 15 \to 16 \to 33 \to 38 \to 0$
10	$0 \to 97 \to 53 \to 54 \to 57 \to 0$

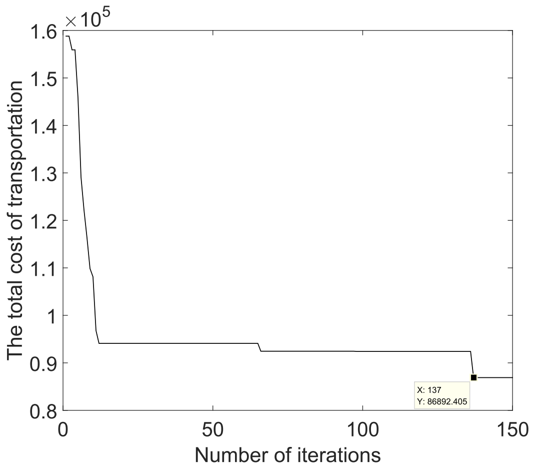

FIGURE 7. Global vs local optimum (R1-01).

TABLE 7. The routing path of IBSO-ACO(R1-01).

Vehicle number	Vehicle routing path
1	$0 \to 27 \to 52 \to 69 \to 70 \to 90 \to 63 \to 11 \to 64 \to 49 \to 47 \to 48 \to 82 \to 1 \to 0$
2	$0 \to 6 \to 92 \to 91 \to 17 \to 84 \to 60 \to 83 \to 18 \to 7 \to 88 \to 10 \to 32 \to 30 \to 20 \to 51 \to 78 \to 35 \to 71 \to 65 \to 0$
3	$0 \to 53 \to 40 \to 2 \to 57 \to 42 \to 43 \to 15 \to 41 \to 74 \to 56 \to 55 \to 25 \to 3 \to 77 \to 80 \to 24 \to 29 \to 34 \to 9 \to 81 \to 33 \to 0$
4	$0 \to 89 \to 99 \to 96 \to 37 \to 100 \to 98 \to 61 \to 16 \to 44 \to 38 \to 14 \to 93 \to 0$
5	$0 \to 28 \to 76 \to 50 \to 12 \to 26 \to 21 \to 73 \to 75 \to 22 \to 72 \to 4 \to 54 \to 0$
6	$0 \to 58 \to 13 \to 97 \to 95 \to 59 \to 94 \to 31 \to 62 \to 19 \to 36 \to 46 \to 0$
7	$0 \to 8 \to 45 \to 85 \to 39 \to 67 \to 23 \to 87 \to 0$
8	$0 \to 5 \to 86 \to 66 \to 68 \to 79 \to 0$

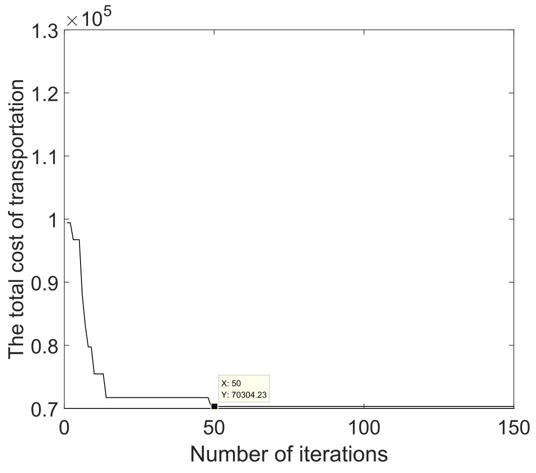

FIGURE 8. Global vs local optimum (R2-01).

TABLE 8. The routing path of IBSO-ACO(R2-01).

Vehicle number	Vehicle routing path
1	$0 \to 6 \to 99 \to 59 \to 92 \to 98 \to 91 \to 16 \to 86 \to 17 \to 84 \to 60 \to 18 \to 48 \to 46 \to 36 \to 64 \to 63 \to 0$
2	$0 \to 53 \to 40 \to 58 \to 26 \to 54 \to 55 \to 25 \to 24 \to 77 \to 79 \to 78 \to 3 \to 80 \to 29 \to 34 \to 35 \to 71 \to 0$
3	$0 \to 1 \to 69 \to 50 \to 76 \to 33 \to 51 \to 81 \to 9 \to 20 \to 70 \to 30 \to 10 \to 90 \to 32 \to 11 \to 0$
4	$0 \to 89 \to 94 \to 95 \to 97 \to 37 \to 42 \to 2 \to 74 \to 72 \to 21 \to 73 \to 22 \to 75 \to 41 \to 15 \to 0$
5	$0 \to 13 \to 87 \to 57 \to 100 \to 44 \to 38 \to 43 \to 14 \to 93 \to 4 \to 56 \to 0$
6	$0 \to 52 \to 5 \to 83 \to 8 \to 82 \to 7 \to 88 \to 62 \to 19 \to 49 \to 47 \to 45 \to 0$
7	$0 \to 28 \to 12 \to 68 \to 65 \to 66 \to 39 \to 67 \to 0$
8	$0 \to 27 \to 31 \to 85 \to 61 \to 96 \to 23 \to 0$

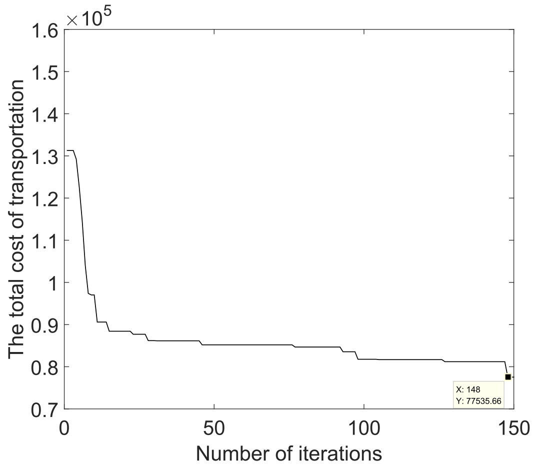

FIGURE 9. Global vs local optimum (RC1-03).

Fig. 8 show the optimal solution of each iteration and the vehicle routing result obtained by IBSO-ACO on data set R2-01. As shown in Fig. 8, IBSO-ACO converges with 50 iterations and the optimal cost 70304.23. The required number of vehicles is 8. The optimal routing path obtained by the proposed IBSO-ACO algorithm is shown in Table 8.

Fig. 9 show the optimal solution of each iteration and the vehicle routing result obtained by IBSO-ACO on data set RC1-03. As shown in Fig. 9, IBSO-ACO converges with

TABLE 9. **The routing path of IBSO-ACO(RC1-03).**

Vehicle number	Vehicle routing path
1	$0 \to 90 \to 52 \to 74 \to 77 \to 21 \to 48 \to 18 \to 49 \to 19 \to 20 \to 24 \to 83 \to 66 \to 64 \to 0$
2	$0 \to 80 \to 93 \to 62 \to 85 \to 63 \to 33 \to 32 \to 30 \to 28 \to 29 \to 31 \to 27 \to 26 \to 34 \to 0$
3	$0 \to 68 \to 61 \to 1 \to 3 \to 5 \to 45 \to 46 \to 8 \to 100 \to 55 \to 88 \to 60 \to 73 \to 14 \to 47 \to 0$
4	$0 \to 81 \to 54 \to 72 \to 39 \to 42 \to 44 \to 43 \to 37 \to 35 \to 41 \to 2 \to 0$
5	$0 \to 98 \to 4 \to 6 \to 7 \to 78 \to 79 \to 11 \to 0$
6	$0 \to 82 \to 53 \to 9 \to 13 \to 15 \to 16 \to 17 \to 10 \to 12 \to 97 \to 0$
7	$0 \to 65 \to 99 \to 87 \to 59 \to 75 \to 58 \to 25 \to 23 \to 57 \to 86 \to 0$
8	$0 \to 92 \to 91 \to 95 \to 67 \to 71 \to 96 \to 94 \to 50 \to 36 \to 0$
9	$0 \to 69 \to 70 \to 40 \to 38 \to 22 \to 76 \to 0$
10	$0 \to 56 \to 84 \to 51 \to 89 \to 0$

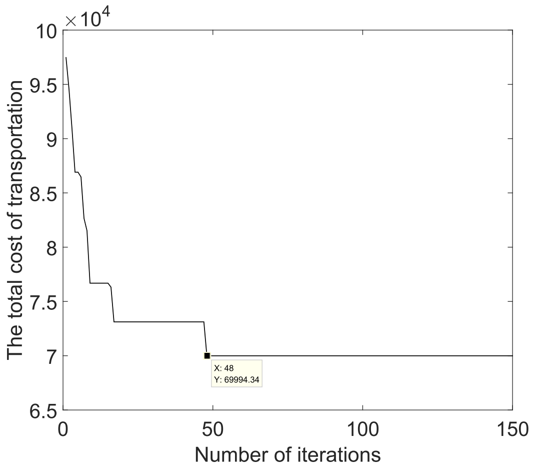

FIGURE 10. **Global vs local optimum (RC2-05).**

TABLE 10. **The routing path of IBSO-ACO(RC2-05).**

Vehicle number	Vehicle routing path
1	$0 \to 90 \to 66 \to 83 \to 57 \to 52 \to 86 \to 74 \to 77 \to 25 \to 21 \to 19 \to 18 \to 0$
2	$0 \to 91 \to 94 \to 96 \to 81 \to 93 \to 71 \to 54 \to 80 \to 95 \to 62 \to 50 \to 85 \to 63 \to 33 \to 0$
3	$0 \to 82 \to 53 \to 98 \to 60 \to 73 \to 78 \to 17 \to 47 \to 14 \to 12 \to 13 \to 9 \to 10 \to 0$
4	$0 \to 68 \to 61 \to 100 \to 1 \to 5 \to 8 \to 7 \to 2 \to 3 \to 46 \to 45 \to 6 \to 0$
5	$0 \to 92 \to 56 \to 64 \to 24 \to 48 \to 23 \to 20 \to 49 \to 22 \to 51 \to 84 \to 0$
6	$0 \to 72 \to 44 \to 42 \to 40 \to 38 \to 39 \to 37 \to 43 \to 35 \to 36 \to 0$
7	$0 \to 65 \to 99 \to 87 \to 59 \to 97 \to 75 \to 58 \to 15 \to 16 \to 0$
8	$0 \to 67 \to 32 \to 28 \to 29 \to 31 \to 30 \to 34 \to 26 \to 27 \to 89 \to 0$
9	$0 \to 88 \to 55 \to 70 \to 41 \to 69 \to 79 \to 4 \to 0$
10	$0 \to 76 \to 11 \to 0$

148 iterations and the optimal cost 77535.66. The required number of vehicles is 10. The optimal routing path obtained by the proposed IBSO-ACO algorithm is shown in Table 9.

Fig. 10 show the optimal solution of each iteration and the vehicle routing result obtained by IBSO-ACO on data set RC2-05. As shown in Fig. 10, IBSO-ACO converges with 48 iterations and the optimal cost of 69994.34. The required number of vehicles is 10. The optimal routing path obtained by the proposed IBSO-ACO algorithm is shown in Table 10.

V. CONCLUSION

In this paper, we have investigated the problem of vehicle routing with soft time window and proposed a novel algorithm based on ant colony optimization and brainstorm optimization. To address the issue that the traditional ant colony algorithm may fall into local optimal, we have carefully designed an improved brainstorm optimization algorithm to enrich the diversity of the set of solutions obtained by the ant colony algorithm. We have conducted extensive experiments to evaluate the performance of the proposed algorithm, and the results have verified the superiority of our proposed algorithm over traditional ant colony algorithm and the simulated annealing ant colony algorithm in that the total transport cost is reduced at a higher convergence rate.

REFERENCES

[1] K. El Bouyahyiouy and A. Bellabdaoui, "An ant colony optimization algorithm for solving the full truckload vehicle routing problem with profit," in *Proc. Int. Colloq. Logistics Supply Chain Manage.*, 2017, pp. 142–147.

[2] S. Joshi and S. Kaur, "Comparative analysis of two different heuristics for model of VRP," in *Proc. 2nd Int. Conf. Adv. Comput. Commun. Eng.*, May 2015, pp. 124–127.

[3] K. Thirugnanasambandam, R. Ramalingam, V. Thirumal, and S. Pothula, "Greedy based population seeding technique in ant colony optimization algorithm," in *Proc. Int. Conf. Commun. Electron. Syst. (ICCES)*, Oct. 2016, pp. 1–4.

[4] I. Niroomand, A. H. Khataie, and M. R. Galankashi, "Vehicle routing with time window for regional network services—Practical modelling approach," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Dec. 2014, pp. 903–907.

[5] M.-C. Chen, Y.-H. Hsiao, H. Reddy, and M. K. Tiwari, "A particle swarm optimization approach for route planning with cross-docking," in *Proc. 7th Int. Conf. Emerg. Trends Eng. Technol. (ICETET)*, Nov. 2015, pp. 1–6.

[6] G. Kim, Y. S. Ong, C. K. Heng, P. S. Tan, and N. A. Zhang, "City vehicle routing problem (City VRP): A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1654–1666, Aug. 2015.

[7] P. Kirci, "On the performance of tabu search algorithm for the vehicle routing problem with time windows," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud Workshops (Fi-CloudW)*, Aug. 2016, pp. 351–354.

[8] A. Abdo, S. Edelkamp, and M. Lawo, "Nested rollout policy adaptation for optimizing vehicle selection in complex VRPs," in *Proc. IEEE 41st Conf. Local Comput. Netw. Workshops (LCN Workshops)*, Nov. 2016, pp. 213–221.

[9] K. Liu and M. Zhang, "Path planning based on simulated annealing ant colony algorithm," in *Proc. 9th Int. Symp. Comput. Intell. Design*, vol. 2, 2016, pp. 461–466.

[10] M. Jun, T. Xingzhi, and X. Wenxia, "Study on VRP based on improved ant colony optimization and Internet of vehicles," in *Proc. IEEE Conf. Expo Transp. Electrific. Asia–Pacific*, 2014, pp. 1–6.

[11] Y. He, J. Wen, and M. Huang, "Study on emergency relief VRP based on clustering and PSO," in *Proc. Int. Conf. Comput. Intell. Secur.*, 2015, pp. 43–47.

[12] S. Mouthuy, F. Massen, Y. Deville, and P. Van Hentenryck, "A multistage very large-scale neighborhood search for the vehicle routing problem with soft time windows," *Transp. Sci.*, vol. 49, no. 2, pp. 223–238, 2015.

[13] Z. Fu, R. Eglese, and L. Y. O. Li, "A unified tabu search algorithm for vehicle routing problems with soft time windows," *J. Oper. Res. Soc.*, vol. 59, no. 5, pp. 663–673, 2008.

[14] L. I. Jian-Bin *et al.*, "Applied optimal strategies for warehouse picking routing in B2C," *Oper. Res. Manage. Sci.*, 2014.

[15] M. Chen, Y. Hao, L. Hu, M. S. Hossain, and A. Ghoneim, "Edge-CoCaCo: Toward joint optimization of computation, caching, and communication on edge cloud," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 21–27, Jun. 2018.

[16] A. Bazzi, B. M. Masini, A. Zanella, and A. Calisti, "Visible light communications as a complementary technology for the Internet of vehicles," *Comput. Commun.*, vol. 93, pp. 39–51, Nov. 2016.

[17] B. Liu *et al.*, "Infrastructure-assisted message dissemination for supporting heterogeneous driving patterns," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2865–2876, Oct. 2017.

[18] Y. Zhang, M. Chen, N. Guizani, D. Wu, and V. C. Leung, "SOVCAN: Safety-oriented vehicular controller area network," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 94–99, 2017.

[19] L. Wu, J. Wang, K. R. Choo, and D. He, "Secure key agreement and key protection for mobile device user authentication," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 319–330, Feb. 2019.

[20] M. Hwang and K. Chen, *Big-Data Analytics for Cloud, IoT and Cognitive Computing*. Hoboken, NJ, USA: Wiley, 2017.

[21] L. Wu, Y. Xia, Z. Wang, and H. Wang, "Be stable and fair: Robust data scheduling for vehicular networks," *IEEE Access*, vol. 6, pp. 32839–32849, 2018.

[22] A. B. Pratiwi, "A hybrid cat swarm optimization—Crow search algorithm for vehicle routing problem with time windows," in *Proc. 2nd Int. Conf. Inf. Technol., Inf. Syst. Electr. Eng. (ICITISEE)*, Nov. 2017, pp. 364–368.

[23] Y. Gao and C. Liu, "Hybrid ant colony algorithm for logistics distribution problem with time windows," in *Proc. 10th Int. Symp. Comput. Intell. Design (ISCID)*, Dec. 2017, pp. 289–291.

[24] S. Jung and B.-R.Moon, "A hybrid genetic algorithm for the vehicle routing problem with time windows," in *Proc. Int. Conf. Innov. Inf., Embedded Commun. Syst. (ICIIECS)*, Mar. 2015, pp. 1–4.

[25] M. Baranwal, P. M. Parekh, L. Marla, S. M. Salapaka, and C. L. Beck, "Vehicle routing problem with time windows: A deterministic annealing approach," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2016, pp. 790–795.

[26] X. Meng, J. Li, B. Qian, M. Zhou, and X. Dai, "Improved population-based incremental learning algorithm for vehicle routing problems with soft time windows," in *Proc. 11th IEEE Int. Conf. Netw., Sens. Control*, Apr. 2014, pp. 548–553.

[27] S. Iqbal and M. S. Rahman, "Vehicle routing problems with soft time windows," in *Proc. 7th Int. Conf. Elect. Comput. Eng.*, Dec. 2012, pp. 634–638.

[28] S. Belhaiza, R. M'Hallah, and G. B. Brahim, "A new hybrid genetic variable neighborhood search heuristic for the vehicle routing problem with multiple time windows," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1319–1326.

[29] D. Aggarwal, V. Kumar, and A. Girdhar, "Lagrangian relaxation for the vehicle routing problem with time windows," in *Proc. Int. Conf. Intell. Comput., Instrum. Control Technol. (ICICICT)*, Jul. 2017, pp. 1601–1606.

[30] A. Arvianto, D. S. Perkasa, W. Budiawan, P. W. Laksosno, and S. S. Aptadi, "Vehicle routing problem modelling to minimize a number of vehicle by considering heterogenous fleet vehicle," in *Proc. Joint Int. Conf. Electr. Veh. Technol. Ind., Mech., Elect. Chem. Eng. (ICEVT IMECE)*, Nov. 2015, pp. 380–388.

[31] Y. Zhou and J. Wang, "A local search-based multiobjective optimization algorithm for multiobjective vehicle routing problem with time windows," *IEEE Syst. J.*, vol. 9, no. 3, pp. 1100–1113, Sep. 2015.

[32] W. Shi, T. Weise, P. R. R. Chiong, and B. Çatay, "Hybrid PACO with enhanced pheromone initialization for solving the vehicle routing problem with time windows," in *Proc. IEEE Symp. Series Comput. Intell.*, Dec. 2015, pp. 1735–1742.

[33] H. Zhu and Y. Shi, "Brain storm optimization algorithms with k-medians clustering algorithms," in *Proc. Int. Conf. Adv. Comput. Intell.*, 2015, pp. 107–110.

[34] L. J. Pan, "Vehicle routing problem with time windows and its algorithm," Ph.D. dissertation, Dept. College Transp. Eng., Central South Univ., Changsha, China, 2012.

LIBING WU is currently a Professor with the School of Computer Science, Wuhan University. He is a Doctoral Supervisor, a Vice-Dean, and a CCF Director/Distinguished Member. In recent years, he has presided over the projects of the National Natural Science Foundation of China, the Innovative Group of Hubei Province, the Science and Technology Support Program of Hubei Province, the key projects and the top projects of the Natural Science Foundation of Hubei Province, the key scientific and technological research projects of Wuhan City, the subprojects of the National Science and Technology Support Program of the Ministry of China, and the special support of the China Postdoctoral Science Foundation, and research works. His main research areas are wireless networks, network management, and distributed computing. He is a member of the IEEE, the ACM, and the Internet, Network and Communication Committee of the China Computer Society. He is the Chairman of the YOCSEF Wuhan Sub-Forum 2012–2013.

ZHIJUAN HE received the M.Sc. degree in computer science from Wuhan University. Her research interest includes vehicle routing optimization.

YANJIAO CHEN received the B.E. degree in electronic engineering from Tsinghua University, in 2010, and the Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology, in 2015, under the supervision of Prof. Q. Zhang. In 2013, she was a Visiting Scholar with the Singapore University of Technology and Design, with Prof. L. Duan. From 2015 to 2016, she has been a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, under the supervision of Prof. B. Li. She has joined the Department of Computer Science, Wuhan University, as a Professor, in 2016.

DAN WU is currently an Associate Professor with the School of Computer Science, University of Windsor. His research interests include uncertain reasoning, machine learning, mobile robotics, multi-agent systems, and knowledge representation.

JIANQUN CUI received the Ph.D. degree in engineering from Wuhan University, in 2008. She was a Visiting Scholar with the Buffalo State University of New York, from 2017 to 2018. She presided over the National Natural Science Foundation of China, the Hubei Natural Science Foundation, the Wuhan Science and Technology Morning Project, and the Fundamental Research Business Fee Project of the Central University of Central China Normal University. She is currently a Professor, a Doctoral Tutor, a member of the CCF Network and Data Communication Committee, the Director of the Hubei Computer Society and the Internet of Things Engineering Department, and a Key Researcher with the Computer Network and Communication Research Institute, Central China Normal University.