IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 29, 2020, accepted March 22, 2020, date of publication March 31, 2020, date of current version June 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984660

A Hybrid Swarm Intelligence Algorithm for Vehicle Routing Problem With Time Windows

YANG SHEN[1], MINGDE LIU[1], JIAN YANG[1], YUHUI SHI[1], (Fellow, IEEE), AND MARTIN MIDDENDORF[2]

[1]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
[2]Department of Computer Science, Leipzig University, 04009 Leipzig, Germany

Corresponding author: Yuhui Shi (shiyh@sustech.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFC0804002, in part by the National Science Foundation of China under Grant 61761136008, in part by the Shenzhen Peacock Plan under Grant KQTD2016112514355531, and in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X386.

ABSTRACT The Vehicle Routing Problem with Time Windows (VRPTW) has drawn considerable attention in the last decades. The objective of VRPTW is to find the optimal set of routes for a fleet of vehicles in order to serve a given set of customers within capacity and time window constraints. As a combinatorial optimization problem, VRPTW is proved NP-hard and is best solved by heuristics. In this paper, a hybrid swarm intelligence algorithm by hybridizing Ant Colony System (ACS) and Brain Storm Optimization (BSO) algorithm is proposed, to solve VRPTW with the objective of minimizing the total distance. In the BSO procedure, both inter-route and intra-route improvement heuristics are introduced. Experiments are conducted on Solomon's 56 instances with 100 customers benchmark, the results show that 42 out of 56 optimal solutions (18 best and 24 competitive solutions) are obtained, which illustrates the effectiveness of the proposed algorithm.

INDEX TERMS Ant colony system, brain storm optimization, heuristics, swarm intelligence, vehicle routing problem with time windows.

I. INTRODUCTION

In recent years, logistics has been playing an important role in many areas, such as economy, industry and environment, etc. The Vehicle Routing Problem (VRP) is a logistics problem and has drawn considerable attention in the last decades. VRP has many real-world applications in industry, seeking optimal solutions can make real-world logistics more efficient, reduce transportation cost and satisfy customer requests better, etc. According to the 2019 third-party logistics study,[1] reducing transportation cost is still the top challenge.

VRP is a combinatorial optimization problem seeking to find the optimal set of routes for a fleet of vehicles in order to serve a given set of customers. In fact, VRP is a generic name given to a whole class of problems, the basic VRP makes assumptions such as there is only one depot, the fleet vehicles are homogeneous, one route per vehicle, etc. Researchers eliminate these assumptions by regarding them as constraints, which results in many variations of traditional VRP, such as Capacitated Vehicle Routing Problem (CVRP) [1], Vehicle Routing Problem with Time Windows (VRPTW) [2], Dynamic Vehicle Routing Problem (DVRP) [3], Vehicle Routing Problem with Pickup and Delivery (VRPPD) [4], etc. In this paper, we address VRPTW, aiming to minimize the number of vehicles (NV) first, and then the total distance (TD). A solution for VRPTW is feasible if the set of routes satisfy the constraints, i.e., all vehicle capacities are not exceeded, and all customers are served within the given time windows. A typical example of VRPTW is shown in Fig. 1, in which each customer node has its location and a certain service time window, and three vehicles depart from the depot to service customer requests.

Determining the optimal solution to VRP is NP-hard [5], current VRP algorithms can be divided into two main

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

[1]https://www.supplychain247.com/paper/2019_third_party_logistics_study_the_state_of_logistics_outsourcing

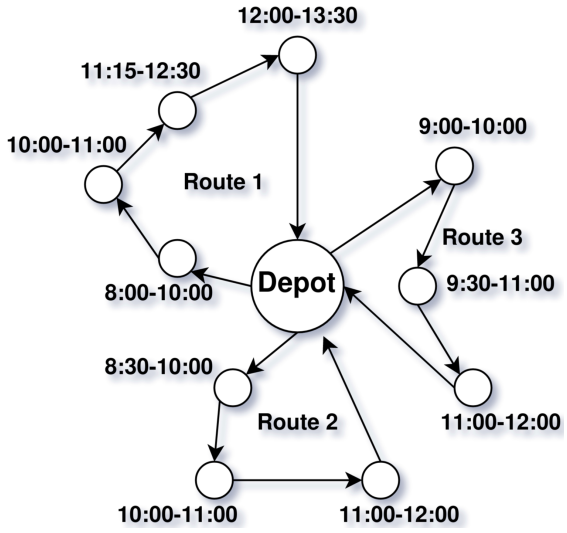

FIGURE 1. A typical example of VRPTW.

categories: exact algorithms and heuristic algorithms [6]. Exact algorithms [7], [8] usually solve small scale VRP, with the size of the problem increases, the computational time of exact algorithms grows exponentially. Golden *et al.* [9] point out that exact algorithms do not work well on VRP with more than 50 customers.

When dealing with large scale VRP, heuristic algorithms can obtain near-optimal solutions within acceptable computational time, which is more suitable for practical applications [6]. Heuristics can be divided into constructive heuristics and improvement heuristics according to their functions [6]. Constructive heuristics aim to find feasible solutions, the most popular ones are saving heuristic [10], nearest neighbor algorithm [11], insertion heuristic [12], sweep algorithm [13], cluster-first route-second algorithm [14], and route-first cluster-second algorithm [15], etc. Improvement heuristics includes 1) intra-route improvement heuristics such as 2-opt [16], Or-opt exchanges [17], 2-opt* [18] and 4-opt* exchanges [19], etc.; and 2) inter-route heuristics such as λ-interchange [20], cyclic exchanges [21], edge exchange schemes [22], ejection chains [23]–[25], very Large Neighbor Search (VLNS) [26], Adaptive Large Neighbor Search (ALNS) [27], etc. In addition, many other optimization techniques have also been applied in recent research on VRPTW, such as lagrangian relaxation [28], [29], integer programming [30], etc.

Apart from classic heuristics, many metaheuristics have been proposed for VRP in recent years. These proposed metaheuristics are mainly based on local search, population search, and learning mechanisms. Local searches include simulated annealing (SA) [31]–[33], deterministic annealing [1], tabu search (TS) [31], etc. Population searches are mostly evolutionary algorithms and swarm intelligence algorithms, which include adaptive memory procedures [34], Genetic Algorithm (GA) [35], Ant Colony Optimization (ACO) [36], [37], Ant Colony Systems (ACS) [38], Particle Swarm Optimization (PSO) algorithm [39], Brain Storm Optimization (BSO) algorithm [40], [41], firefly algorithm [42], etc. Methods of learning mechanisms are mainly neural networks [43].

Many comparison studies [44]–[46] have analyzed the impact of different heuristics and metaheuristics for VRP, the conclusions drawn from those studies showed that no single heuristic or metaheuristic could exceed others in all the cases, and certain cases require dynamic heuristic analysis to determine which heuristics to use according to their features. The heuristic analysis also illustrated that hybridization allows enhancing the strengths and compensating the weaknesses of two or more methods, with the aim of generating better solutions by combining the key elements of competing methodologies. In this paper, we further explore and implement multiple heuristics including ACS, BSO, 2-opt and λ-interchange to achieve near-optimal solutions for VRPTW. The peripheral frame of the algorithm is ACS, in which after setting the initial information, ants begin to construct routes and update pheromones locally. When all the ants have constructed their solutions, the best solution found by all the ants is sent to BSO for further optimization. In the modified BSO procedure, 2-opt heuristic is performed for intra-route improvement if one route is selected, and λ-interchange are performed for inter-route improvement if two routes are selected. The further optimization strategy based on BSO and the improvement heuristics is performed not only enhance the search in the solution space, but also avoid local optimum of ACS.

It is worth to mention that Wu *et al.* [41] proposed a brainstorming-based ant colony optimization algorithm named IBSO-ACO to solve VRP with soft time windows. In the IBSO-ACO method, an improved BSO was designed and combined with the ACO algorithm. The main differences between their work and ours are: 1) A penalty cost is added to their objective function if the constraints of the time window is violated, while we focus on VRP with hard time windows, i.e., time window constraints must be satisfied by all vehicles; 2) The algorithm proposed by us hybridized ACS instead of the classic ACO in order to balance exploration and exploitation better due to the state transition rule in the ACS [47]; 3) The global pheromone update in the ACS also makes the search more directed; 4) In the BSO procedure, we applied a different clustering scheme which clusters the population according to the geographical coordinates of customers in different routes, while the IBSO-ACO clusters the population according to the cost; 5) The IBSO-ACO was performed at the solution level, i.e., it maintains a population of solutions, and generates new solution randomly, which is very time consuming and will probably lead to infeasible solutions. However, the proposed algorithm applied different heuristics such as 2-opt and λ-interchange to generate new solutions, which is more effective and more efficient.

The main contributions of this paper are:

- A hybrid ACS-BSO algorithm is proposed, in which BSO is used to further optimize the solution and to avoid local optimum compared to classic ACS.

- Both intra-route and inter-route improvement are considered in the BSO procedure.
- 56 instances of VRPTW with 100 customers are evaluated to demonstrate the effectiveness of the proposed algorithm.

The rest of this paper is organized as follows. Section II describes the definition and mathematical model of VRPTW. Section III first introduces classic ACS, BSO, 2-opt and λ-interchange algorithms, and then proposes the hybrid ACS-BSO algorithm. Section IV evaluates ACS and the proposed algorithm. Section V concludes the paper.

II. PROBLEM DEFINITION AND MODELING

The VRPTW can be defined as a directed complete graph $G(V, E)$, where $V = \{v_0, v_1, \ldots, v_n\}$ is the vertex set, $E = \{(v_i, v_j)|v_i, v_j \in V,\ i \neq j\}$ is the edge set. Normally, v_0 is set as the depot, and $\{v_1, v_2, \ldots, v_n\}$ are N customers. A set of $|K|$ homogenous vehicles with the same capacity Q depart from depot v_0. Each customer v_i has a demand of capacity q_i and a service time window $[e_i, l_i]$, where e_i is the earliest time at which service for customer v_i may start, and l_i is the latest time at which service may start. Thus, a vehicle must wait if it arrives at customer v_i before e_i, and it must arrive before l_i. Each customer request also has a service time s_i, and each customer in the network requires to be serviced by one vehicle only once. The travel cost c_{ij} between vertices i and j is represented in proportion to Euclidean distance between them. In the 100 customer instances of Solomon's VRPTW benchmark, the vehicle speed is set as the unit, i.e., the time cost t_{ij} is equal to c_{ij}. The mathematical model of VRPTW is defined as follows [46].

Parameters description:

K	the set of all vehicles
V	the set of all customers
N	total number of customers
Q	maximum vehicle capacity
c_{ij}	distance cost from vertex i to vertex j
t_{ij}	travel time from vertex i to vertex j
q_i	demand of customer at vertex v_i
e_i	earliest arrival time at vertex v_i
l_i	latest arrival time at vertex v_i
s_i	service time at vertex v_i
t_i	arrival time at vertex v_i
w_i	wait time at vertex v_i

Objective function:

$$\min \mathrm{TD} = \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ijk} \tag{1}$$

$$\text{subject to: } x_{ijk} = \begin{cases} 1 & \text{if vehicle } k \text{ travels from } v_i \text{ to } v_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\sum_{i\in V} x_{i0k} = \sum_{j\in V} x_{0jk} = 1 \quad (\forall k \in K) \tag{3}$$

$$\sum_{j\in V, j\neq i} x_{ijk} = \sum_{j\in V, j\neq i} x_{jik} \leq 1 \quad (\forall i \in V,\ \forall k \in K) \tag{4}$$

$$\sum_{k\in K}\sum_{i\in V, i\neq j} x_{ijk} = 1 \quad (\forall j \in V) \tag{5}$$

$$\sum_{k\in K}\sum_{j\in V, j\neq i} x_{ijk} = 1 \quad (\forall i \in V) \tag{6}$$

$$\sum_{i\in V} q_i \sum_{j\in V, j\neq i} x_{ijk} \leq Q \quad (\forall k \in K) \tag{7}$$

$$w_j = max\{e_j - t_i - t_{ij},\ 0\} \quad (\forall i, j \in V, i \neq j) \tag{8}$$

$$t_i + s_i + t_{ij} + w_i \leq t_j \quad (\forall i, j \in V, i \neq j) \tag{9}$$

$$e_i \leq t_i + w_i \leq l_i \quad (\forall i \in V) \tag{10}$$

where TD is the total distance of all vehicles in Eq. (1). Eqs. (3)-(4) illustrate that there are maximum $|K|$ vehicles used to serve customers. Eqs. (5)-(6) ensure that each customer is serviced by one vehicle only once. The vehicle capacity constraint is specified by Eq. (7). The time windows constraints are defined by Eqs. (8)-(10). A solution is feasible if all the constraints are satisfied. The route can be represented as a concatenation of customers, and the solution is represented as a list of routes, a typical solution for a VRP with 10 customers is shown in Fig. 2, which has two routes: 0-5-2-1-6-3-0; 0-4-8-7-10-9-0.

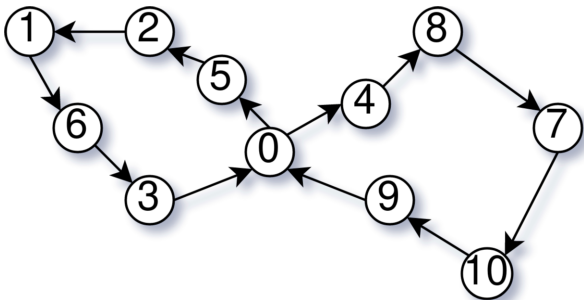

FIGURE 2. A solution for VRP with 10 customers (vertex 0 as depot).

III. PROPOSED HYBRID ACS-BSO ALGORITHM

In this section, we first introduce the ACS, improvement heuristics 2-opt and λ-interchange (local search), and BSO algorithm. Then, the hybrid ACS-BSO algorithm is proposed. The breadth search of solutions is ensured via swarm intelligence algorithm due to its population based feature, while the depth search of solutions is achieved by the local search heuristics. Therefore, hybridization of swarm intelligence algorithm and local search leads to both breadth and depth search of solutions.

A. ANT COLONY SYSTEM

Ant Colony Optimization (ACO) [48] is first proposed by Dorigo, which is inspired by ant behavior of leaving pheromones to direct each other to food while exploring the environment. When a colony of ants have different routes to reach the food, those who travel the shorter route go back and forth to the depot more frequently and leave more pheromones. Ants choose route according to the density

of pheromones, i.e., the more pheromones left on a route, the more likely ants choose this route. At the same time, pheromones also evaporate over time.

In ACO, the state transition rule, i.e., the probability of ant k moves from customer i to j is defined in Eq. (11):

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij}^{\alpha}) \cdot (\eta_{ij}^{\beta})}{\sum_{k \in J_k(i)} (\tau_{ik}^{\alpha}) \cdot (\eta_{ik}^{\beta})} & \text{if } j \in J_k(i) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where τ_{ij} is the pheromone deposited for transition from customer i to j, η is the desirability of state transition (normally set as $1/d_{ij}$, where d_{ij} is the distance between customer i and j), $\beta \geq 1$ is a parameter which controls the relative influence of η_{ij}, and $J_k(i)$ is the feasible set of customers that remain to be visited by ant k.

When a solution is found by the ant colony, pheromones along the edges are updated according to Eq. (12):

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \sum_{k} \Delta \tau_{ij}^{k} \tag{12}$$

where $0 < \rho < 1$ is the pheromone evaporation coefficient, and $\Delta \tau_{ij}^{k}$ is the pheromone deposited by ant k, which is defined by Eq. (13):

$$\Delta \tau_{ij}^{k} = \begin{cases} 1/L_k & \text{if edge } (i, j) \in \text{ant } k\text{'s route} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where L_k is the length of the route traveled by ant k.

ACS [47] is a variation of ACO algorithm, which differs from ACO in three main aspects:

1) a probability parameter is added to the state transition rule to balance exploration and exploitation
2) a local pheromone update rule is applied when ants are constructing routes
3) a global pheromone update is applied only to the edges in the best route

In ACS, the state transition rule is defined by Eq. (14):

$$s = \begin{cases} \arg\max_{j \in J_k(i)} \tau_{ij} \cdot \eta_{ij}^{\beta} & \text{if } q \leq q_0 \\ S & \text{otherwise} \end{cases} \tag{14}$$

where $0 \leq q \leq 1$ is a uniformly distributed random number, $0 \leq q_0 \leq 1$ is a parameter of probability, which allows ants to focus more on exploitation when $q \leq q_0$ and focus more on exploration otherwise. S is the state transition probability from Eq. (11).

In ACS, the local pheromone update is performed by Eq. (15):

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta \tau_{ij} \tag{15}$$

The global pheromone update is performed by Eq. (16):

$$\tau_{ij} \leftarrow (1 - \alpha) \cdot \tau_{ij} + \alpha \cdot \Delta \tau_{ij} \tag{16}$$

where

$$\Delta \tau_{ij} = \begin{cases} 1/L_{gbest} & \text{if edge } (i, j) \in \text{global best route} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where α is the pheromone evaporation rate, and L_{gbest} is the length of the global best route. The global pheromone update is performed when all the ants have completed the tours, and only the ant which constructed the shortest tour is allowed to deposit pheromone, which makes the search more directed. The pseudocode of the ACS algorithm is shown in Alg. 1.

B. 2-OPT AND λ-INTERCHANGE

1) INTRA-ROUTE IMPROVEMENT WITH 2-OPT

In optimization, 2-opt [16] is a widely used local search algorithm first proposed by Croes for solving TSP. The main idea of 2-opt algorithm is to reverse a subset of the route itself, as shown in Fig. 3, in which 2-opt is applied to modify a single route: the original route is 0-4-2-1-3-5-6-7-0, after performing 2-opt algorithm, the order of the sub-route 3-5 is reversed, and the new route is 0-4-2-1-5-3-6-7-0.

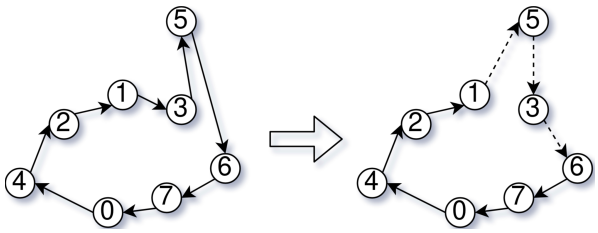

FIGURE 3. 2-opt for intra-route improvement.

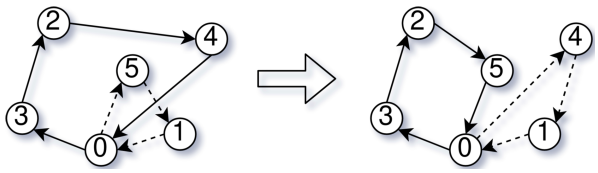

FIGURE 4. λ-interchange for inter-route improvement with operator (1,1).

2) INTER-ROUTE IMPROVEMENT WITH λ-INTERCHANGE

λ-interchange [20] is first proposed by Osman *et al.* It is an improvement heuristic which can interchange customer nodes between routes, the parameter λ is the maximum number of customers that would be interchanged. There are $2^{\lambda+1}$ interchange operators in total, due to the computational cost, normally $\lambda \leq 3$. In this paper, we only consider $\lambda = 2$, i.e., there are eight possible interchange operators which are (0,1), (1,0), (1,1), (0,2), (2,0), (1,2), (2,1) and (2,2). Operator (1,2) means that for a chosen pair of routes (R_p, R_q), one customer node from R_p is shifted to R_q and two customer nodes from R_q are shifted to R_p. After the interchange of customer nodes, only feasible and improved solutions will be accepted. An example of λ-interchange with operator (1,1) is shown in Fig. 4, in which the original route 1 is 0-3-2-4-0, and the route 2 is 0-5-1-0, after interchange with operator (1,1), customer node 4 is shifted from route 1 to route 2, and customer node 5 is shifted from route 2 to route 1,

Algorithm 1 ACS Algorithm

Input: $G(V, E)$, M: Set of ants
function Set_Init_Positions
  for $\forall k \in M$ **do**
    let r_{k1} be the starting vertex for ant k
    $J_k(r_{k1}) \leftarrow V - r_{k1}$ ▷ $J_k(r)$: set of unvisited vertices for ant k
    $r_k \leftarrow r_{k1}$ ▷ r_k: current vertex where ant k locates
  end for
end function
function Construct_Routes
  for $i := 1$ to $|V| - 1$ **do**
    for $\forall k \in M$ **do**
      if $q \leq q_0$ **then** ▷ exploitation
        choose next vertex s_k according to Eq. (14)
      else ▷ exploration
        choose next vertex s_k according to Eq. (11)
      end if
      add edge (r_k, s_k) to $Route_k$
      $r_k \leftarrow s_k$
    end for
  end for
  for $\forall k \in M$ **do**
    add edge (r_k, r_{k1}) to $Route_k$
  end for
end function
function Local_Pheromone_Update
  compute $L_k \forall k \in M$ ▷ L_k: length of $Route_k$
  update pheromones according to Eq. (15)
end function
function Global_Pheromone_Update
  compute L_{gbest} ▷ L_{gbest}: global best route
  update pheromones according to Eqs. (16)-(17)
end function
function Main
  for $\forall edge(r, s) \in E$ **do**
    $\tau_{r,s} \leftarrow \tau_0$
    $\eta_{r,s} \leftarrow 1/c_{r,s}$ ▷ $C_{r,s}$: distance between r and s
  end for
  loop ▷ at this level each loop is called an iteration
    Set_Init_Positions
    loop ▷ at this level each loop is called a step
      Construct_Routes
      Local_Pheromone_Update
    end loop
    Global_Pheromone_Update
  end loop
end function

the obtained new route 1 is 0-3-2-5-0, and new route 2 is 0-4-1-0.

There are two selection strategies for selecting candidate solutions S' from $N_\lambda(S)$, where $N_\lambda(S)$ is the neighborhood solutions of current solution S.

1) Best-Improve (BI) strategy goes over all solutions S' in $N_\lambda(S)$ and selects the one which results in maximum decrease in cost.
2) First-Improve (FI) strategy accepts the first solution S' in $N_\lambda(S)$ which results in a decrease in cost.

Since BI strategy usually takes too much computational time than FI strategy, in this paper, 2-interchange with FI strategy is implemented to make the algorithm more efficient, i.e., the algorithm accepts the first improved solution and runs the next iteration.

C. BRAIN STORM OPTIMIZATION

Brain Storm Optimization (BSO) [49], [50] was first introduced in 2011, which is inspired by the human brainstorming process, and has been widely and successfully used to solve a lot of optimization problems [51], [52]. The procedure of classic BSO algorithm is described as follows.

1) Randomly generate N individuals / solutions, initialize parameters p_1, p_2, p_3, p_4;
2) Clustering: Cluster N solutions into M clusters, and mark the best solution in each cluster as the cluster center;
3) Evaluate N solutions according to fitness function;
4) Replacing: Generate a random number $r_1 \in (0, 1)$, if $r < p_1$, randomly select a cluster center, and randomly generate a solution to replace it;
5) Generating: Generate a random number $r_2 \in (0, 1)$, if $r_2 < p_2$, randomly select a cluster and generate a random number $r_3 \in (0, 1)$. If $r_3 < p_3$, generate a new solution by adding random values to the selected cluster center, otherwise, generate a new solution by adding random values to a random solution in selected cluster; If $r_2 \geq p_2$, randomly select two clusters, and generate a random number $r_4 \in (0, 1)$, if $r_4 < p_4$, then combine two cluster centers and add random values to generate a new solution, otherwise, combine two random solutions in selected clusters and add random values to generate a new solution;
6) Selecting: Evaluate the new generated solution, and compare it to the existing solution with the same index, the better one is kept and recorded;
7) If N new solutions have been generated, go to step 8; otherwise, go to step 5;
8) Terminate the procedure if the maximum number of iterations is reached; otherwise, go to step 2.

To apply the classic BSO to VPRTW, the new solution generation operation in step 5) performs at the solution level, which is very time consuming and will probably lead to infeasible solutions. To make the process more efficient, we modify the classic BSO algorithm to optimize VRPTW solutions at the route level. First, we divide routes into two clusters A and B according to their coordinates, i.e., the geographical coordinates of customers in the routes. Other different clustering strategies can also be applied since clustering in BSO is only for simulating the problem owners to pick up better solutions they believe in the brainstorming process.

After clustering, randomization rationale from BSO is performed to enhance solution diversity and to avoid local optimum of ACS. The algorithm has four different ways of generating new solutions: 1) perform 2-opt on a cluster center; 2) perform 2-opt on a random route in the cluster; 3) perform 2-interchange on two cluster centers; 4) perform 2-interchange on two random routes in two different clusters. More details are shown in Alg. 2.

Algorithm 2 Modified BSO for VRPTW

Input: solution S (i.e., NV, routes) as initial solution
Output: new_solution S'
for $i := 1$ to NV **do**
  compute cost for routes in S
end for
while not termination **do**
  perform route clustering on S
  find centers for each cluster
  for $i := 1$ to NV **do**
    if $rand(0, 1) < p_1$ **then**
      randomly pick a cluster C_j
      if $rand(0, 1) < p_2$ **then**
        $nr_i \leftarrow$ 2-opt(c_j) ▷ c_j: center of C_j
      else
        $nr_i \leftarrow$ 2-opt(r_j) ▷ r_j: random route in C_j
      end if
    else
      randomly pick two clusters C_j, C_k
      if $rand(0, 1) < p_3$ **then**
        $nr_i \leftarrow$ 2-interchange(c_j, c_k)
      else
        $nr_i \leftarrow$ 2-interchange(r_j, r_k)
      end if
    end if
    $S' \leftarrow$ update S with nr_i
    if S' outperforms S **then**
      $S \leftarrow S'$
    end if
  end for
end while
return S

D. PROPOSED HYBRID ACS-BSO ALGORITHM

The proposed hybrid ACS-BSO algorithm combines population based method and local search, the overall procedure is described in Alg. 3.

The first step is to initialize the parameters for ACS and BSO algorithms. After initialization, the outer-loop of ACS starts. In the inner-loop of the proposed algorithm, the initial solution is constructed in two ways, either picking the nearest neighbor as the next customer or picking a new customer randomly. In this case, the diversity of the solution is ensured, and different initial solutions can also help to avoid local optimum. When the initial solution is constructed, ant actions described in Section III-A is performed. Since local

Algorithm 3 Hybrid ACS-BSO Algorithm

1: Initialize parameters
2: **while** not termination **do** ▷ each loop is an iteration
3: set ants' initial positions
4: **while** not termination **do** ▷ each loop is a step
5: construct solution
6: perform local pheromone update
7: send current best solution to BSO
8: further optimize by BSO with local search
9: **end while**
10: perform global pheromone update
11: **end while**

pheromone update can't ensure the quality of the solution, current best solution is then sent to BSO to get further optimization. Besides, further intra-route and inter-route optimization can also improve the diversity of solutions. Global pheromones of ants are updated after BSO. The proposed algorithm will output the best solution found if the condition of termination is satisfied, i.e., either the maximum number of iterations is achieved, or the solution is not improved after a certain number of iterations.

IV. EXPERIMENTS AND DISCUSSIONS

For our experiments, we choose the 56 instances of Solomon's benchmark with 100 customers, which is most widely used for evaluation. The benchmark has six sets of problems: C1, C2, R1, R2, RC1, and RC2. "C" stands for clustered, which means that the geographical coordinates of customers are clustered in problem sets C1 and C2. "R" represents random, which means the benchmark data are randomly generated (uniformly distributed) in problem sets R1 and R2. And "RC" means a mix of random and clustered. In problem sets 1 (i.e., R1, C1, and RC1), the capacity of the vehicle is small, and the time windows are narrow, thus more vehicles are required to service the customers, and fewer customers will be serviced by the same vehicle. The number of vehicles required are normally larger than 10 for problem sets 1. In contrast, problem sets R2, C2 and RC2 have wide time windows and permit more customers per route, the number of vehicles required are much fewer.

A. EXPERIMENT SETUP

The parameters for ACS are set as follows: $M = 30$, $\alpha = 1$, $\beta = 2$, $\rho = 0.1$, $q_0 = 0.1$, $max\_iter = 500$. All the parameters were tuned to balance the quality of solutions and the computational cost. Although the maximum number of iterations of ACS is set as 500, the proposed algorithm terminates earlier before achieving the maximum number of iterations except for a few complicate instances.

BSO is used to further improve the solutions obtained by ACS, the probability parameters for BSO are set as: $p_1 = 0.3$, $p_2 = 0.4$, $p_3 = 0.5$. Since it is nested in the loop of ACS, the number of maximum iterations for BSO is set as 15, i.e., for each current best solution obtained by ACS, BSO

TABLE 1. Results for Solomon's 56 instances with 100 customers.

Instances	BKS		ACS		Ours (best NV)		Ours (best TD)		Cost Reduction (%)	Gap (%)
	NV	TD	NV	TD	BNV	TD	NV	BTD		
C101	10	828.94 [55]	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0.00	0.00
C102	10	828.94 [55]	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0.00	0.00
C103	10	828.06 [55]	**10**	**828.06**	**10**	**828.06**	**10**	**828.06**	0.00	0.00
C104	10	824.78 [55]	10	831.27	**10**	**828.78**	**10**	**828.78**	-0.30	0.00
C105	10	828.94 [55]	**10**	**828.94**	**10**	**824.94**	**10**	**824.94**	0.00	0.00
C106	10	828.94 [55]	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0.00	0.00
C107	10	828.94 [55]	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0.00	0.00
C108	10	828.94 [55]	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0.00	0.00
C109	10	828.94 [55]	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0.00	0.00
C201	3	591.56 [55]	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**	0.00	0.00
C202	3	591.56 [55]	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**	0.00	0.00
C203	3	591.17 [55]	**3**	**591.17**	**3**	**591.17**	**3**	**591.17**	0.00	0.00
C204	3	590.60 [55]	3	593.93	**3**	**590.60**	**3**	**590.60**	-0.56	0.00
C205	3	588.88 [55]	**3**	**588.88**	**3**	**588.88**	**3**	**588.88**	0.00	0.00
C206	3	588.49 [55]	**3**	**588.49**	**3**	**588.49**	**3**	**588.49**	0.00	0.00
C207	3	588.29 [55]	**3**	**588.29**	**3**	**588.29**	**3**	**588.29**	0.00	0.00
C208	3	588.32 [55]	**3**	**588.32**	**3**	**588.32**	**3**	**588.32**	0.00	0.00
R101	20	1642.88 [55]	21	1682.05	**19**	**1671.16**	**20**	**1642.88**	-2.38	0.00
R102	18	1472.62 [56]	19	1523.67	**17**	**1504.60**	18	1479.55	-2.98	0.47
R103	14	1213.62 [56]	15	1257.44	14	1245.86	15	1225.31	-2.62	0.95
R104	11	976.61 [55]	11	1040.22	11	1010.73	12	1002.62	-3.75	2.59
R105	15	1360.78 [55]	16	1417.47	15	1366.05	15	1365.66	-3.79	0.36
R106	13	1240.47 [57]	14	1276.37	13	1288.84	13	1249.51	-2.15	0.72
R107	11	1073.34 [57]	13	1137.02	11	1101.56	12	1091.21	-4.20	1.64
R108	10	947.55 [57]	11	982.88	10	974.17	10	960.23	-2.36	1.32
R109	13	1151.84 [57]	15	1246.26	**12**	**1165.71**	**12**	**1165.71**	-6.91	1.19
R110	12	1072.41 [55]	13	1168.85	**11**	**1090.92**	**11**	**1090.92**	-7.14	1.70
R111	12	1053.50 [55]	13	1121.77	**11**	**1148.14**	12	1063.69	-5.46	0.96
R112	10	953.63 [55]	11	1036.72	10	1004.53	11	976.28	-6.19	2.32
R201	8	1147.8 [58]	**5**	**1249.04**	**4**	**1336.05**	8	1161.20	-7.56	1.15
R202	8	1034.35 [55]	**5**	**1148.50**	**4**	**1128.05**	**6**	**1058.83**	-8.47	2.31
R203	6	874.87 [55]	6	947.16	**3**	**1020.10**	**5**	**883.42**	-7.22	0.97
R204	5	735.8 [58]	5	776.58	**3**	**834.92**	**4**	**756.93**	-2.60	2.79
R205	5	954.16 [58]	6	1035.89	**3**	**1105.38**	6	978.47	-5.87	2.48
R206	5	879.89 [55]	5	981.96	**3**	**949.11**	**4**	**906.27**	-8.35	2.91
R207	4	799.86 [57]	4	869.00	4	812.35	4	812.35	-6.97	1.54
R208	4	705.45 [59]	4	745.34	**2**	**940.30**	**3**	**725.05**	-2.80	2.70
R209	5	859.39 [55]	5	879.52	**3**	**1046.73**	5	879.01	-0.06	2.23
R210	5	910.7 [59]	5	968.34	**3**	**1069.26**	7	923.43	-4.86	1.38
R211	4	755.96 [57]	4	820.37	**3**	**836.36**	4	776.17	-5.69	2.60
RC101	15	1623.58 [34]	17	1684.39	16	1643.78	16	1643.78	-2.47	1.23
RC102	14	1461.23 [55]	15	1523.39	14	1464.63	14	1464.63	-4.01	0.23
RC103	11	1261.67 [60]	12	1370.38	11	1275.64	11	1275.65	-7.43	1.10
RC104	10	1135.48 [61]	12	1193.83	10	1156.92	10	1156.92	-3.19	1.85
RC105	16	1518.58 [55]	17	1635.69	**14**	**1609.68**	16	1535.78	-6.51	1.12
RC106	13	1371.69 [53]	15	1461.73	13	1378.45	13	1378.45	-6.04	0.49
RC107	12	1212.83 [57]	14	1385.64	**11**	**1318.69**	12	1216.65	-13.89	0.31
RC108	11	1117.53 [56]	11	1168.31	11	1134.85	11	1134.28	-3.00	1.48
RC201	9	1265.56 [55]	**8**	**1325.47**	**4**	**1514.41**	**8**	**1284.71**	-3.17	1.49
RC202	8	1095.64 [55]	8	1141.42	**4**	**1326.71**	**7**	**1127.02**	-1.28	2.78
RC203	5	928.51 [57]	7	1012.38	**3**	**1166.91**	6	943.13	-7.34	1.55
RC204	4	786.38 [58]	6	851.87	**3**	**929.94**	4	807.71	-5.47	2.64
RC205	7	1157.55 [58]	8	1204.78	**4**	**1360.91**	8	1170.98	-2.89	1.15
RC206	7	1054.61 [55]	7	1137.24	**3**	**1237.21**	7	1093.64	-3.99	3.57
RC207	6	966.08 [55]	7	1021.88	**4**	**1039.59**	6	986.70	-3.57	2.09
RC208	4	779.31 [57]	7	862.36	**3**	**910.59**	4	785.60	-9.77	0.80

will further optimize the solution by using either 2-opt or 2-interchange method with first improvement strategy.

The proposed algorithm was programmed in Python, and all our experiments were conducted on an Intel Xeon E5-2650 CPU@2.30GHz PC with 16GB RAM.

B. RESULT ANALYSIS

The best results obtained by classic ACS and the proposed algorithm are shown in Table 1, as well as the Best Known Solutions (BKS) found by other researches so far. In Table 1, "NV" represents the number of vehicles, "TD" means total distance, "BNV" and "BTD" stands for best number of vehicles and best total distance, respectively. Although the objective is to minimize the total distance, researches also focus on minimizing the number of vehicles used as well. Tan *et al.* points out that all the instances in problem sets C1 and C2 have positively correlating objectives, and many instances in problem sets R1, R2, RC1, and RC2 have

TABLE 2. Average results of total distance over 10 runs for each problem set.

Problem Sets	BKS	ACS	Hybrid ACS-BSO	Avg. Cost Reduction (%)	Avg. Gap (%)
C1	828.38	829.10	828.38	-0.09	0.00
C2	589.86	590.28	589.86	-0.07	0.00
R1	1179.94	1240.89	1192.80	-4.03	1.08
R2	878.02	947.43	896.47	-5.68	2.06
RC1	1337.82	1427.92	1350.77	-5.71	0.96
RC2	1004.20	1069.67	1024.94	-4.37	2.02

TABLE 3. Solution to C104.

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

conflicting objectives [53], i.e., a multiobjective optimization problem. We list two columns of the best results found, one with minimum NV and the other with minimum TD. To see how much hybridization of different algorithms cam improve the solutions, we computed the cost reduction between our solutions and solutions obtained by classic ACS. To compare with the BKS, the solutions which have fewer NV or smaller TD are highlighted with bold fonts, the gap (i.e., percentage deviation) between the BTD and the BKS is also computed. The cost reduction [41] and the gap [54] are computed according to Eq. (18) and Eq. (19).

$$Cost\ Reduction = \frac{TD_{ours} - TD_{ACS}}{TD_{ours}} \tag{18}$$

$$Gap = \frac{TD_{ours} - TD_{BKS}}{TD_{ours}} \tag{19}$$

It can be observed from Table 1 that for problem sets C1 and C2, all the best known solutions are found by the proposed algorithm. Classic ACS fails to find two optimal solutions for instances C104 and C204. For problem sets R and RC, more vehicles are used in R1 and RC1 because they have tight time windows, 5/12 and 2/8 solutions with fewer NV or smaller TD are found by the proposed algorithm, all the cost reductions of total distance are further optimized by BSO ranging from -7.14% to -2.15% for problem set R1, and -13.89% to -2.47% for problem set RC1. For problem sets R2 and RC2, almost all solutions with fewer number of vehicles (10/11 and 8/8 respectively) are found by the proposed algorithm, and the total distance of all instances are also optimized by the BSO algorithm. For all instances in problem sets R and RC, all the solutions obtained by the proposed algorithm are better than classic ACS. The proposed algorithm found competitive solutions for 42 out of 56 instances, including all instances in type C and 18 out of 19 instances of problem sets R2 and RC2.

To get an overview of different problem sets, the average cost reduction and gap of each problem sets are also computed. For all the 56 benchmark instances, the experiments were run for 10 times, and the average total distance of BKS, ACS and the proposed hybrid ACS-BSO algorithm, as well as the average cost reduction and gap of each problem sets are shown in Table 2. In Table 2, for problem sets C1 and C2, the proposed algorithm has a slight improvement over classic ACS, and there is no gap since all the optimal solutions are obtained. For problem sets R1, R2, RC1 and RC2, the average cost reduction are much larger than C1 and C2 problem sets, which are -4.03%, -5.68%, -5.71% and -4.37%, respectively. Besides, the average gaps are relatively small, which are 1.08% and 0.96% for problem sets R1 and RC1, 2.06% and 2.02% for problem sets R2 and RC2. The average gaps are higher for R2 and RC2 problem sets is that most solutions obtained have fewer vehicles being used to service the customers, thus result in larger total distance.

For problem sets C1 and C2, the proposed algorithm can find optimal solutions in less than 30 seconds. Thus, in the aspect of convergence speed, the proposed algorithm is very efficient.

C. CASE STUDY

Two heuristics were used in the proposed algorithm, which are 2-opt and 2-interchange. Theoretically, a sequence of operations of 2-opt could get stuck on the local optimum, while a sequence of operations of 2-interchange can move current solution to anywhere in the solution space [46]. However, 2-interchange operation is very time consuming, while 2-opt is simple and effective. Thus, both heuristics were taken

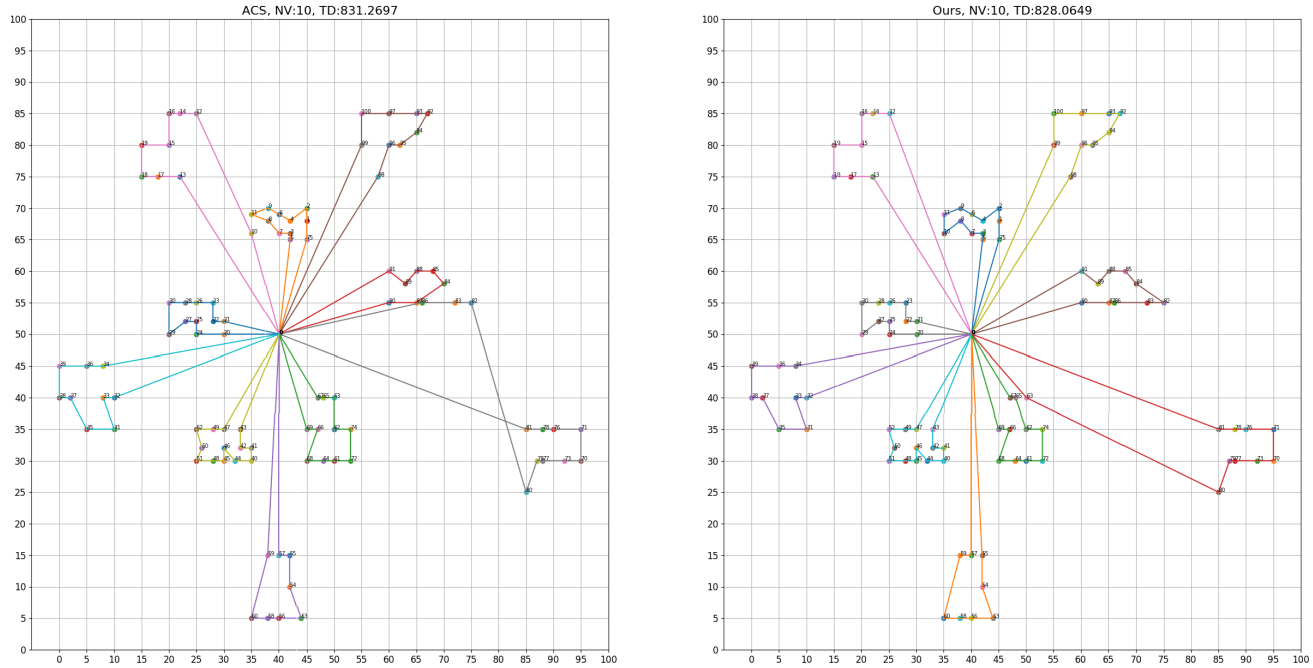

FIGURE 5. Solution to C104 (Left: ACS, Right: Ours).

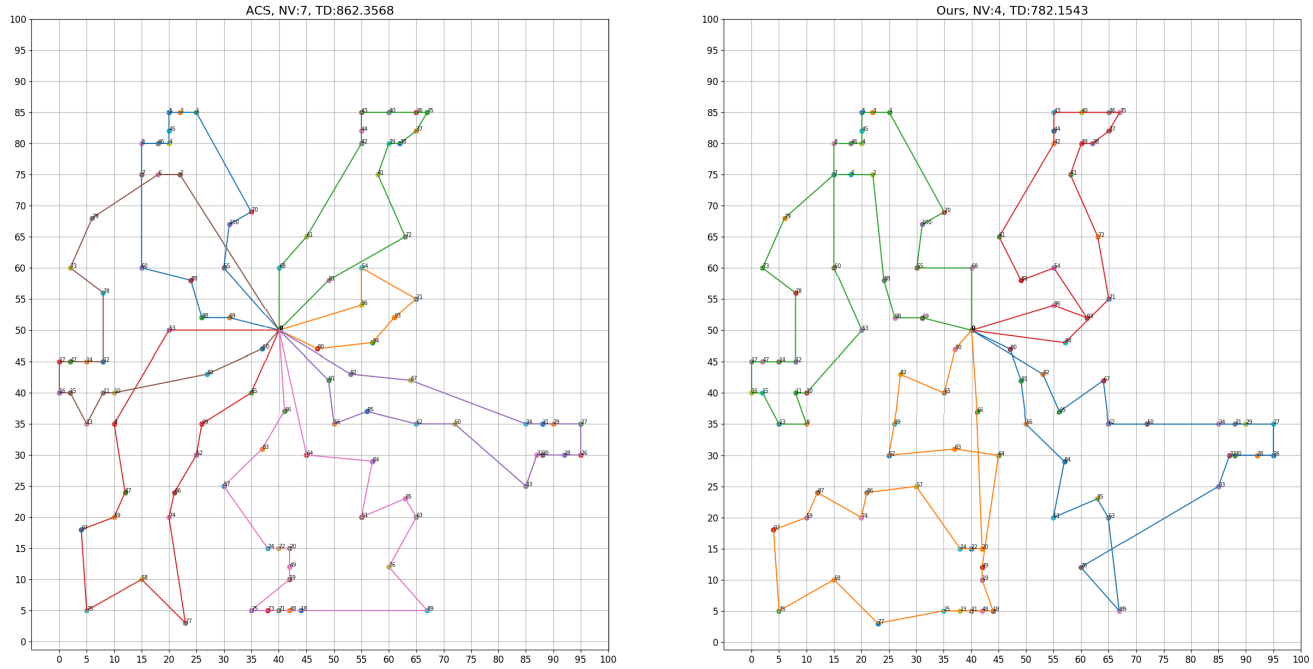

FIGURE 6. Solution to RC208 (Left: ACS, Right: Ours).

to balance the computational cost of the algorithm and the quality of the solutions.

For further analysis of the heuristics used, two instances were chosen, which are C104 and RC208. The solutions to instance C104 are very similar to each other, while the solutions to instance RC208 are quite different. The solutions are shown in Fig. 5 and Fig. 6, and the detailed routes are shown in Table 3 and Table 4.

It can be observed from Table 3 that most edges are the same between the solutions obtained by ACS and the proposed algorithm. Taking vehicle 2 and 7 as an example, the two routes of ACS are 0-5-3-7-8-11-9-6-4-2-1-75-0 and

TABLE 4. Solution to RC208.

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

0-13-17-18-19-15-16-14-12-10-0, which can be transferred to 0-5-3-7-8-10-11-9-6-4-2-1-75-0 and 0-13-17-18-19-15-16-14-12-0 by taking one 2-interchange operation with the operator (0,1). In addition, for vehicle 5, the routes of ACS is 0-57-55-54-53-56-58-60-59-0, while the routes of the proposed algorithm is 0-55-54-53-56-58-60-59-57-0, the only difference is the position of customer node 57. Apparently, it would take a lot of 2-opt operations to mutate the route 5 of ACS to the route 5 of the proposed algorithm, but only two 2-interchange operations to make such mutation.

For instance RC208, it can be observed from Fig. 6 that the number of vehicles is reduced from 7 to 4 by the proposed algorithm, and the total distance is reduced from 862.36 to 782.15. The detailed routes are shown in Table 4. There are many differences between the two solutions, but there also exists many short fragments in two solutions. Firstly, the intra-route improvement by 2-opt would change the position of the customer nodes in the routes, but 2-opt operation is not able to reduce the number of vehicles. Secondly, in Table 4, many routes in the solution to ACS are short, i.e., there are not many customer nodes in the routes. In addition, the inter-route improvement by 2-interchange with operator (0, x) or (x, 0) is able to reduce the number of vehicles. Thus, the number of vehicles is reduced from 7 to 4. However, to optimize the solution to ACS to the the solution obtained by the proposed algorithm, a lot of 2-interchange operations have to be taken. Therefore, any single heuristic has its limitations, it is essential to perform multiple heuristics.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a hybrid swarm intelligence algorithm for solving the VRPTW. The ACS, BSO, as well as 2-opt and λ-interchange local search heuristics, were illustrated. The proposed algorithm uses ACS to optimize the solution first, and then performs BSO with local search for further optimization. Experiments on Solomon's benchmark with 100 customers showed that the proposed algorithm can achieve competitive results comparing to the best known solutions obtained by many other different methods. In addition, although classic ACS method can achieve good quality solutions for VRPTW, hybridization of different algorithms can highly improve those solutions achieved from one single classical method. We think a successful strategy must consider two aspects: i) ''breadth'' via population based method, ii) ''depth'' via local search. Our experimental results obtained very competitive solutions with regard to the best known solutions, a total of 42 out of 56 optimal solutions (18 best and 24 competitive solutions) were found.

Many successful VRP metaheuristics use either local search or large neighborhood search (LNS). The main idea of LNS is ''destory and repair'', and LNS usually requires very large computational cost to explore the search space better. In this paper, the ACS and the BSO algorithms were used to explore the search space, which are more efficient due to their population-based properties. In addition, the 2-opt and λ-interchange local search methods were applied, which are simple and effective.

The new solution generation operations of BSO in the proposed algorithm have been performed at the route level. According to our observation from the near-optimal solutions, most fragments (i.e., edges in routes) are the same as the best known solutions. In this case, cross over at solution level to inherit good fragments is another possible way to improve the convergence speed and produce high quality solutions. Further research could also focus on solving other variants of the VRP and related problems.

ACKNOWLEDGMENT

(Yang Shen, Mingde Liu, and Jian Yang contributed equally to this work.)

REFERENCES

[1] F. Li, B. Golden, and E. Wasil, ''Very large-scale vehicle routing: New test problems, algorithms, and results,'' *Comput. Oper. Res.*, vol. 32, no. 5, pp. 1165–1179, May 2005.

[2] J. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis, ''VRP with time windows,'' in *The Vehicle Routing Problem* (SIAM Monographs on Discrete Mathematics and Applications), vol. 9. Philadelphia, PA, USA: SIAM, 2002, pp. 157–193.

[3] D. M. Chitty and M. L. Hernandez, ''A hybrid ant colony optimisation technique for dynamic vehicle routing,'' in *Proc. Genetic Evol. Comput. Conf.* Berlin, Germany: Springer, 2004, pp. 48–59.

[4] I. Gribkovskaia, O. Halskau, and K. N. B. Myklebost, ''Models for pick-up and deliveries from depots with lasso solutions,'' in *NOFOMA2001, Collaboration in Logistics: Connecting Islands Using Information Technology*. Goteborg, Sweden: Department of Transportation and Logistics, Chalmers Univ. of Technology, Reykjavik, Iceland, 2001, pp. 279–293.

[5] J. K. Lenstra and A. H. G. R. Kan, ''Complexity of vehicle routing and scheduling problems,'' *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[6] N. A. El-Sherbeny, "Vehicle routing with time windows: An overview of exact, heuristic and Metaheuristic methods," *J. King Saud Univ.-Sci.*, vol. 22, no. 3, pp. 123–131, Jul. 2010.
[7] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 3, pp. 345–358, Jun. 1992.
[8] R. Baldacci, N. Christofides, and A. Mingozzi, "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts," *Math. Program.*, vol. 115, no. 2, pp. 351–385, Oct. 2008.
[9] B. L. Golden, S. Raghavan, and E. A. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges*, vol. 43. New York, NY, USA: Springer, 2008.
[10] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Oper. Res.*, vol. 12, no. 4, pp. 568–581, 1964.
[11] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "An analysis of several heuristics for the traveling salesman problem," *SIAM J. Comput.*, vol. 6, no. 3, pp. 563–581, 1977.
[12] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, Apr. 1987.
[13] B. E. Gillett and L. R. Miller, "A heuristic algorithm for the vehicle-dispatch problem," *Oper. Res.*, vol. 22, no. 2, pp. 340–349, Apr. 1974.
[14] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981.
[15] J. E. Beasley, "Route first—Cluster second methods for vehicle routing," *Omega*, vol. 11, no. 4, pp. 403–408, 1983.
[16] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Res.*, vol. 6, no. 6, pp. 791–812, Dec. 1958.
[17] I. Or, "Traveling salesman type combinatorial problems and their relation to the logistics of regional blood banking," Ph.D. dissertation, Northwestern Univ., Evanston, IL, USA, 1977.
[18] J.-Y. Potvin and J.-M. Rousseau, "An exchange heuristic for routeing problems with time windows," *J. Oper. Res. Soc.*, vol. 46, no. 12, pp. 1433–1446, Dec. 1995.
[19] J. Renaud, F. F. Boctor, and G. Laporte, "A fast composite heuristic for the symmetric traveling salesman problem," *INFORMS J. Comput.*, vol. 8, no. 2, pp. 134–143, May 1996.
[20] I. H. Osman and N. Christofides, "Capacitated clustering problems by hybrid simulated annealing and tabu search," *Int. Trans. Oper. Res.*, vol. 1, no. 3, pp. 317–336, Jul. 1994.
[21] P. M. Thompson and H. N. Psaraftis, "Cyclic transfer algorithm for multivehicle routing and scheduling problems," *Oper. Res.*, vol. 41, no. 5, pp. 935–946, Oct. 1993.
[22] G. A. P. Kindervater and M. W. P. Savelsbergh, "Vehicle routing: handling edge exchanges," in *Local Search in Combinatorial Optimzation*. Princeton, NJ, USA: Princeton Univ. Press, 1997, pp. 337–360.
[23] J. Xu and J. P. Kelly, "A network flow-based tabu search heuristic for the vehicle routing problem," *Transp. Sci.*, vol. 30, no. 4, pp. 379–393, Nov. 1996.
[24] C. Rego and C. Roucairol, "A parallel tabu search algorithm using ejection chains for the vehicle routing problem," in *Meta-Heuristics*. Boston, MA, USA: Springer, 1996, pp. 661–675.
[25] C. Rego, "A subpath ejection method for the vehicle routing problem," *Manage. Sci.*, vol. 44, no. 10, pp. 1447–1459, Oct. 1998.
[26] O. Ergun, J. B. Orlin, and A. Steele-Feldman, "Creating very large scale neighborhoods out of smaller ones by compounding moves: A study on the vehicle routing problem," *SSRN Electron. J.*, 2002.
[27] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transp. Sci.*, vol. 40, no. 4, pp. 455–472, Nov. 2006.
[28] 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.
[29] V. Kumar, M. S. Bhangu, and R. Anand, "Lagrangian relaxation for distribution networks with cross-docking centre," *Int. J. Intell. Syst. Technol. Appl.*, vol. 18, nos. 1–2, p. 52, 2019.
[30] D. Aggarwal and V. Kumar, "Mixed integer programming for vehicle routing problem with time windows," *Int. J. Intell. Syst. Technol. Appl.*, vol. 18, nos. 1–2, p. 4, 2019.
[31] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Ann. Oper. Res.*, vol. 41, no. 4, pp. 421–451, Dec. 1993.
[32] W.-C. Chiang and R. A. Russell, "Simulated annealing metaheuristics for the vehicle routing problem with time windows," *Ann. Oper. Res.*, vol. 63, no. 1, pp. 3–27, Feb. 1996.
[33] V. F. Yu, S.-W. Lin, W. Lee, and C.-J. Ting, "A simulated annealing heuristic for the capacitated location routing problem," *Comput. Ind. Eng.*, vol. 58, no. 2, pp. 288–299, Mar. 2010.
[34] Y. Rochat and É. D. Taillard, "Probabilistic diversification and intensification in local search for vehicle routing," *J. Heuristics*, vol. 1, no. 1, pp. 147–167, Sep. 1995.
[35] E. Alba and B. Dorronsoro, "Solving the vehicle routing problem by using cellular genetic algorithms," in *Proc. 4th Eur. Conf. Evol. Comput. Combinat. Optim.* (Lecture Notes in Computer Science), vol. 3004. Coimbra, Portugal: Springer, Apr. 2004, pp. 11–20.
[36] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Adv. Eng. Inform.*, vol. 18, no. 1, pp. 41–48, 2004.
[37] B. Yu, Z.-Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routing problem," *Eur. J. Oper. Res.*, vol. 196, no. 1, pp. 171–176, 2009.
[38] M. Reimann, K. Doerner, and R. F. Hartl, "D-ants: Savings based ants divide and conquer the vehicle routing problem," *Comput. Oper. Res.*, vol. 31, no. 4, pp. 563–591, Apr. 2004.
[39] T. J. Ai and V. Kachitvichyanukul, "A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery," *Comput. Oper. Res.*, vol. 36, no. 5, pp. 1693–1702, May 2009.
[40] L. Ke, "A brain storm optimization approach for the cumulative capacitated vehicle routing problem," *Memetic Comput.*, vol. 10, no. 4, pp. 411–421, Dec. 2018.
[41] L. Wu, Z. He, Y. Chen, D. Wu, and J. Cui, "Brainstorming-based ant colony optimization for vehicle routing with soft time windows," *IEEE Access*, vol. 7, pp. 19643–19652, 2019.
[42] D. Aggarwal and V. Kumar, "Performance evaluation of distance metrics on Firefly Algorithm for VRP with time windows," *Int. J. Inf. Technol.*, pp. 1–8, Nov. 2019.
[43] H. Ghaziri, "Solving routing problems by a self-organizing map," in *Artificial Neural Network*. Amsterdam, The Netherlands: North-Holland, 1991, pp. 829–834.
[44] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *Int. Trans. Oper. Res.*, vol. 7, nos. 4–5, pp. 285–300, Sep. 2000.
[45] A. V. Breedam, "Comparing descent heuristics and metaheuristics for the vehicle routing problem," *Comput. Oper. Res.*, vol. 28, no. 4, pp. 289–315, Apr. 2001.
[46] K. C. Tan, L. H. Lee, Q. L. Zhu, and K. Ou, "Heuristic methods for vehicle routing problem with time windows," *Artif. Intell. Eng.*, vol. 15, no. 3, pp. 281–295, Jul. 2001.
[47] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[48] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Milan, Italy, 1992.
[49] Y. Shi, "Brain storm optimization algorithm," in *Proc. Int. Conf. Swarm Intell.* Berlin, Germany: Springer, 2011, pp. 303–309.
[50] Y. Shi, "An optimization algorithm based on brainstorming process," in *Emerging Research on Swarm Intelligence and Algorithm Optimization*. Hershey, PA, USA: IGI Global, 2015, pp. 1–35.
[51] S. Cheng, J. Chen, X. Lei, and Y. Shi, "Locating multiple optima via brain storm optimization algorithms," *IEEE Access*, vol. 6, pp. 17039–17049, 2018.
[52] Z. Jia, H. Duan, and Y. Shi, "Hybrid brain storm optimisation and simulated annealing algorithm for continuous optimisation problems," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 2, p. 109, 2016.
[53] K. C. Tan, Y. H. Chew, and L. H. Lee, "A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows," *Comput. Optim. Appl.*, vol. 34, no. 1, pp. 115–151, May 2006.
[54] D. Bustos Coral, M. Oliveira Santos, C. Toledo, and L. Fernando Nino, "Clustering-based search in a memetic algorithm for the vehicle routing problem with time windows," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.
[55] S. Jung and B.-R. Moon, "A hybrid genetic algorithm for the vehicle routing problem with time windows," in *Proc. 4th Annu. Conf. Genetic Evol. Comput.* San Mateo, CA, USA: Morgan Kaufmann, 2002, pp. 1309–1316.

[56] G. B. Alvarenga, G. R. Mateus, and G. de Tomi, "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows," *Comput. Oper. Res.*, vol. 34, no. 6, pp. 1561–1584, Jun. 2007.
[57] J. Berger, M. Salois, and R. Begin, "A hybrid genetic algorithm for the vehicle routing problem with time windows," in *Proc. Adv. 12th Biennial Conf. Can. Soc. Comput. Stud. Intell. Artif. Intell. (AI)* (Lecture Notes in Computer Science), vol. 1418. Vancouver, BC, Canada: Springer, Jun. 1998, pp. 114–127.
[58] H. C. B. de Oliveira and G. C. Vasconcelos, "A hybrid search method for the vehicle routing problem with time windows," *Ann. Operations Res.*, vol. 180, no. 1, pp. 125–144, Nov. 2010.
[59] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "Time-window relaxations in vehicle routing heuristics," *J. Heuristics*, vol. 21, no. 3, pp. 329–358, 2015.
[60] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in *Proc. 4th Int. Conf. Princ. Pract. Constraint Program.* (Lecture Notes in Computer Science), vol. 1520. Pisa, Italy: Springer, Oct. 1998, pp. 417–431.
[61] J.-F. Cordeau, G. Laporte, and A. Mercier, "A unified tabu search heuristic for vehicle routing problems with time windows," *J. Oper. Res. Soc.*, vol. 52, no. 8, pp. 928–936, Aug. 2001.

YANG SHEN received the B.E. degree from Northwestern Polytechnical University, in 2013. His research interests include swarm intelligence, evolutionary computation, artificial intelligence, multiobjective optimization, and so on.

MINGDE LIU received the B.E. degree from the Hebei University of Technology, in 2018. His research interests include swarm intelligence, evolutionary computation, artificial intelligence, machine learning, dynamic optimization, and so on.

JIAN YANG received the B.S. and M.S. degrees from the Department of Control Science and Engineering, Harbin Institute of Technology, China, in 2005 and 2007, respectively, and the Ph.D. degree from the Department of Mechanical Engineering, Harbin Institute of Technology, Shenzhen, in 2019. He is currently a Research Fellow with the Department of Computer Science and Engineering, SUSTech Artificial Intelligence Institute (SAINT), Southern University of Science and Technology (SUSTech). His research interests include swarm intelligence, robotics, computational intelligence, intelligent cybernetics, and bio-inspired systems.

YUHUI SHI (Fellow, IEEE) received the Ph.D. degree in electronic engineering from Southeast University, Nanjing, China, in 1992. He is currently the Chair Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. Before joining the Southern University of Science and Technology, he was with Electronic Data Systems Corporation, Indianapolis, IN, USA. His main research interests include the areas of computational intelligence techniques (including swarm intelligence) and their applications. He is the Editor-in-Chief of the *International Journal of Swarm Intelligence Research.*

MARTIN MIDDENDORF received the Diploma degree in mathematics and the Dr. rer. nat. degree from Leibniz University Hannover, Germany, in 1988 and 1992, respectively, and the Professorial Habilitation degree from the University of Karlsruhe, Germany, in 1998. He has worked at the Technical University of Dortmund, Germany, and Leibniz University Hannover, as a Visiting Professor of computer science. He was a Professor of computer science with the Catholic University of Eichstaett-Ingolstadt, Germany. He is currently a Professor with the Swarm Intelligence and Complex Systems Group, Leipzig University, Germany. His research interests include algorithms from nature, bioinformatics, and swarm intelligence.