A Hybrid BSO-ACO for Dynamic Vehicle Routing Problem on Real-World Road Networks

MINGDE LIU1, QI SONG1, QI ZHAO2, LING LI1, ZHIMING YANG1, AND YINGBIN ZHANG1

1China Telecom Research Institute, Guangzhou 510630, China
2Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
Corresponding author: Mingde Liu (rosandthree@gmail.com)

ABSTRACT The Dynamic Vehicle Routing Problem With Time Windows (DVRPTW) is an NP-hard problem, which has attracted a lot of attention in the past decades due to its many practical applications in logistics. In order to better describe the actual logistics distribution scenario, this paper studies the DVRPTW based on real road networks and proposes the hybrid BSO-ACO algorithm, which is a combination of Brain Storm Optimization (BSO), Ant Colony Optimization (ACO) and Neighborhood Search (2-opt, relocate, exchange). The algorithm 1) uses ACO to generate new individuals from the same cluster formed by BSO, and increases exploitation by ACO’s pheromone accumulation, 2) harnesses the 2-opt, relocate, and exchange to increase exploration to avoid the algorithm from falling into local optima. We construct a test set by extracting the real road networks in Panyu District, Guangzhou, China and compare the hybrid BSO-ACO algorithm with other algorithms on this test set. The computation experiments show the effectiveness and efficiency of the hybrid BSO-ACO algorithm.

INDEX TERMS Ant colony optimization, brain storm optimization, dynamic vehicle routing problem with time windows, road network.

I. INTRODUCTION

The vehicle routing problem (VRP) was first proposed by George Dantzig in 1959 [1], which is a classical combinatorial optimization problem with many real-life application scenarios, especially in the logistics industry. The research on VRPs is the key to enhancing the core competitiveness of the logistics industry [2]. It has received extensive attention from academia and industry in the past decades. Building an efficient logistics and distribution system and vehicle route system can improve logistics efficiency, save logistics and distribution costs, and enhance customer satisfaction. The purpose of VRPs is to guide vehicles to serve customers from the warehouse by planning the lowest cost vehicle routes and satisfying some constraints. The common constraints of this problem are the time window constraint (Vehicle Routing Problem with Time Windows, VRPTW), vehicle capacity constraint (capacity Vehicle Routing Problem), simultaneous pickup and delivery customer goods constraint (Vehicle Routing Problem with Pickup and Delivery, VRPPD), etc.

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

Traditional VRP is based on a customer-graph (i.e. a complete graph containing customers and a depot) [3], the customers are connected to each other by straight lines. The straight-line general Euclidean distance between customers is usually used in the calculation of distribution transportation costs. The result of the solution is the service order of customers, which lacks guidance for actual vehicle travel. Apart from the customer nodes considered in traditional VRPs, the actual logistics distribution also needs to consider in actual road networks. The cost of choosing different roads between customers is different, and the distribution center needs to reasonably arrange the order of customer service and the selection of roads between customers in the real scenario.

The VRP on road networks can consider road information more closely to guide distribution services, and there have been some studies in this area in recent years. Letchford et al. [4] studied road networks with multiple attributes, e.g., the quickest path between customers is not equal to the cheapest path. They studied VRP on a real road network and solved the path selection problem between customers on the road network by pricing problem (branch and price method, but without bring problem). Huang et al. [3] studied time-dependent
VRP, considering the effect of vehicle speed caused by road traffic conditions, and investigated the case of deterministic and stochastic traffic conditions respectively. They constructed a dataset based on the traffic network in Beijing and finally solved the problem by using the industrial solver (CPLEX). Boyac et al. [5] studied the simplification of road networks by linear Eulerian distances. They constructed a VRP test dataset based on an urban road network. Then they compared the solution results of the two representations under the road network and the linear Eulerian distance. The experiments showed that the Eulerian distance yields a solution similar to that of the road network, i.e., the linear Eulerian distance can simulate the road network very well. This study also pointed out that the Eulerian distance approximation for road networks with time windows may have the consequence that the time windows are not satisfied. Yao et al. [6] studied the consistent VRP. The customer based graph miss some road information and therefore cannot handle the path consistency problem properly. They studied this problem on road networks and solved it by using the brand price and cut algorithm.

Many exact algorithms used to deal with VRPs, which are guaranteed to find the optimal solution, such as column generation(CG) [7], [8], [9], integer linear programming [3], branch and bound algorithm [10]. However, since the VRPs belong to NP-hard problem, these exact algorithms can only solve small scale problems. Exact algorithms require a lot of time for solving medium or large scale VRPs. Therefore, researchers use heuristic algorithms and meta-heuristic algorithms to solve VRPs. Unlike exact algorithms, these two types of algorithms can find satisfactory solutions in a reasonable running time. The popularly used heuristic algorithms include the saving algorithm which merges vehicle routes [11], the nearest neighborhood algorithm in which the vehicle finds the nearest customer to serve [12], the 2-opt algorithm for optimizing intersecting parts of routes leading to detours [13], and the λ-exchange algorithm for vehicle route segment interaction, etc. Meta-heuristic algorithms for VRPs include: Tabu Search (TS) algorithm [14], [15], Variable Neighborhood Search(VNS) algorithm [16], [17], Genetic Algorithm (GA) [18], [19], Ant Colony Optimization (ACO) algorithm [20], [21], Particle Swarm Optimization (PSO) algorithm [22], Adaptive Large Neighborhood Search (ALNS) [23], [24], and Brain Storm Optimization(BO) [25], [26], etc.

In recent years, with the rapid development of mobile network technology, logistics companies can know more real-time information, such as the location of vehicles, changes in customer requests, new customer requests, customer cancellations, and so on. Therefore, dynamic VRPs have emerged to handle this real-time dynamic information. Dynamic VRPs need to handle dynamic customer requests and dynamically adjust the vehicle route in real time, and reduce the transportation cost.

Different from static VRPs, dynamic VRPs need to deal with dynamically changing customer demands in real time, and even take into account the road network traffic conditions in real time. As a real-time problem, the DVRP is essential to respond to dynamically changing demands on time, especially when the customer demands have service time constraints, i.e., dynamic VRPTW (DVRPTW). Gendreau et al. [27] first studied the DVRPTW inspired by courier service applications, proposing to use adaptive memory to store the current route information, vehicles do not know the next service customer until finish one, and they use a tabu search-based algorithm to solve the DVRPTW.

Chen and Xu [7] proposed a dynamic column generation algorithm to solve DVRPTW. Campbell et al. [28] considered the dynamic demand DVRPTW in a grocery delivery scenario, by finding a vehicle that serves additional customers to decide whether to accept a delivery request. Hong [29] decomposes DVRPTW into several static DVRPs by adding the emerging customer requests to the current solution and using a large neighborhood search algorithm to optimize the dynamic problem using the original routes information. Khouadjia et al. [30] used a hybrid algorithm that contains particle swarm optimization algorithm and variable neighborhood search approach for DVRP. Euchi et al. [31] studied the dynamic pickup and delivery problem, and the paper proposed an ant colony optimization algorithm combined with the 2-opt local search to solve the static subproblem after the transformation of the dynamic problem. Barkaoui et al. [32] proposed an adaptive genetic algorithm to solve DVRPTW. Shifeng et al. [33] used an adaptive large neighborhood search algorithm to reduce the number of vehicles in DVRPTW. Sabar et al. [34] proposed a population-based algorithm to deal with DVRP. They compared the algorithm extensively with other algorithms, and the experimental results showed that the algorithm has excellent performance.

In the actual DVRPTW, because the vehicle is driving on the road network, there may be some restrictions, such as road speed limit, no parking, u-turn restrictions, etc., and the customer graph-based VRP can not handle these situations. Therefore this paper focuses on the DVRPTW on road networks. We propose a Hybrid BSO-ACO algorithm to solve this problem by transforming the DVRPTW into a series of static problems. The main contributions of this paper are:

- The proposed algorithm uses the clustering idea of BSO and the pheromone accumulation idea of ACO to aggregate the dominant part of similar individuals. Then, individuals are perturbed by 2-opt, exchange, and relocate to avoid getting into local optima.
- We generate 120 sets of test data from the road networks of Guangzhou, China, and compare the Hybrid BSO-ACO algorithm with GVNS and ALNS algorithms. According to the experimental results, the average computation time of each algorithm is almost the same, but BSO-ACO’s average total distance is 3.14% and 2.01% shorter than ALNS and GVNS, respectively. These experimental results show that our algorithm can obtain better solutions for solving DVRPTW on road networks.
The rest of the paper is organized as follows. In Section II, the problem model of DVRPTW on road networks is presented. In Section III, we introduce the Hybrid BSO-ACO algorithm to solve DVRPTW on road networks. Experimental results and discussions are given in Section IV. Finally, the conclusion is provided in Section V.

II. PROBLEM DESCRIPTION

The section first presents the description of VRPTW on road networks, then gives the description and mathematical model of VRPTW, and the DVRPTW on road networks is introduced finally.

A. VRPTW

The VRPTW is defined on a complete graph \( G(C, E) \), where \( C = \{c_0, c_1, \ldots, c_N\} \) is the set of nodes in the graph, it is also the set of customer and depot. \( E = \{(c_i, c_j)|c_i, c_j \in C, i \neq j\} \) refers to edges. In general, \( c_0 \) is set as the depot and \( \{c_1, c_2, \ldots, c_N\} \) is \( N \) customers. The depot has a set of \( |K| \) vehicles. The capacity of the vehicles is \( Q \) and their speed is \( speed \). The vehicles depart from the depot \( c_0 \) to serve the customers and finally go back to the depot \( c_0 \). Each customer \( c_i \) has its demand information: the weight of goods to be delivered \( q_i \), the time \( s_t \) that the vehicle needs to serve that customer, and the acceptable service time window \([e_i, l_i] \), where \( e_i \) and \( l_i \) are the earliest and latest time that a customer \( c_i \) can receive vehicle delivery service, respectively. The vehicle needs to go between \( e_i \) and \( l_i \) to serve customer \( c_i \). If the vehicle reaches the customer \( c_i \) earlier than the \( e_i \) moment, it needs to wait \( w_i \) times. In addition, each customer can be served by only one vehicle. The transportation cost between customer \( c_i \) and customer \( c_j \) is \( cost_{ij} \), which is generally the Euclidean distance between customer nodes in the traditional VRPs. The problem’s constraints are: there is a set of customer demands that require vehicle delivery service, which contains information such as the weight demand of the goods to be delivered by the vehicle, the time window demand of the customer, and the maximum vehicle load constraint.

The traditional VRPTW is based on a complete graph of directly connected customer points. In contrast, real-world VRPTWs take place on road networks, where customers are connected by road networks rather than simply by straight lines. The road network-based VRPTW is defined on the road network graph \( G(V, C, E) \), where \( V = \{v_0, v_1, \ldots, v_m\} \) is the vertex set to represent road nodes, \( v_0 \) represents the depot, \( \{v_1, v_2, \ldots, v_m\} \) represent road nodes, \( C = \{c_1, c_2, \ldots, c_N\} \) is a subset of road nodes, which is used to represent customer nodes. \( E = \{(i, j)^p; i, j \in V, i \neq j, p = 1, \ldots, P\} \) is used to represent the paths between road nodes. The paths between road nodes generally do not contain duplicate road nodes. As shown in Fig. 1, the paths between customers \( a \) and \( b \) includes: \( a-2-b, a-6-b, a-2-6-b, a-6-2-b, a-6-0-13-4-7-3-b \), and so on. The distance (i.e. \( cost_{ij} \)) between road nodes is the length of the road, and the time (i.e. \( t_{ij} \)) spent by the vehicle is equal to the road distance divided by the vehicle speed. The mathematical model of VRPTW on road networks is as follows. An instance of VRPTW on road networks is shown in Fig. 1.

Objective function (1)–(8), as shown at the bottom of the next page.

The objective function (1) aims at minimizing the transportation cost of all vehicles, i.e., the total distance traveled. Equation (2) is represents binary variable used to indicate whether vehicle \( k \) uses path \((i, j)^p\). Equation 3 means the vehicle should start from and end in the depot. The constraint (4) is the capacity constraint of vehicles. The constraints (5-8) represent the time window constraints of vehicles.

An instance of VRPTW on road networks is shown in Fig. 1.

B. DYNAMIC VRPTW ON ROAD NETWORKS

The dynamic VRPTW can be described as follows [35]. A depot in a region of a logistics company performs distribution services to customers in the region. The depot plans the routes of vehicles to schedule them to serve customers. In the process of vehicles going to distribution after departure from the depot, the distribution center also receives two types of customer demands, namely customer new demand, and customer cancellation demand. The goal of optimization is to modify the original vehicle routes to serve the dynamically changing demand. If a new customer demand cannot be served with the original vehicles, a new vehicle is added from the depot to serve the customer. By dynamically adjusting routes to serve as many customers as possible while reducing distribution costs.

To better understand the DVRPTW on road networks, Fig. 2 shows an instance of DVRPTW on road networks. At time \( t_0 \), there are three vehicles (routes) departing from the depot to serve eight customers, and the distribution center plans the routes based on the current customer information. At time \( t_1 \), the distribution center receives two new customer demands and a customer cancellation, so the distribution center re-plans the routes, which cancels some routes, and arranges a vehicle to serve the new customer demand. One of the new customer demands (the red node) could not be served within the specified time window, so the customer demand was rejected. It is important to note that when a vehicle is on the road network, it cannot be operated randomly, and it may violate the traffic kneeling by going against the traffic on the road, so when there is a dynamic customer demand to change the vehicle’s path, we need to wait for the vehicle to reach the nearest road node before re-optimizing it. In this paper, the vehicle arriving at the nearest road node is called the virtual starting point. At time \( t_2 \), the vehicles have served all the customers.

The mathematical model of DVRPTW on road networks is as follows. Only the objective function of the DVRPTW on road networks and the constraints for vehicles to serve customers from virtual starting points are defined here, and the formulations for the other parts of the problem are the same as in the modeling part of the VRPTW on road networks.
### III. PROPOSED APPROACH

The common reoptimization method for DVRPs can be divided into two categories: periodic reoptimization and continuous reoptimization [35]. Periodic reoptimization divides each working day into several time slices of fixed time interval, at the beginning of the working day vehicle routes is optimized based on static customers, and then at the end of each time slice, the vehicle routes will reoptimization based on the currently updated customer information [36]. The continuous reoptimization is to reoptimize vehicle routes when there is dynamic data (i.e., new customer requests or customer cancellation requests) [37]. For the VRPTW, due to the time window constraint, it may not be able to serve the customers in time if it is processed by periodic reoptimization, so this paper uses the continuous processing strategy to solve the dynamic VRPTW on road networks. The procedure is as follows.

**Objective function (9)–(11)**, as shown at the bottom of the next page.

The objective function (9) is the cumulative sum of delivery cost in time slice \( t \). Equation (10) is the cumulative number of unserviceable demands for all time slices, which can be considered as a reflection of total customer satisfaction. Equation (11) indicates that the vehicles in each time slice must depart from the current virtual starting point or depot to serve the customer and return to the depot.

\[
\begin{align*}
\text{min} & \quad \sum_{k \in K} \sum_{(i,j)^p \in E} \text{cost}_{(i,j)^p} x_{(i,j)^p k} \\
\text{subject to :} & \quad x_{(i,j)^p k} = \begin{cases} 
1 & \text{if vehicle } k \text{ uses path } (i, j)^p \\
0 & \text{otherwise}
\end{cases} \\
& \quad \sum_{i \in V} \sum_{p \in E_{(i,0)}} x_{(i,0)^p k} = \sum_{j \in V} \sum_{p \in E_{(0,j)}} x_{(0,j)^p k} = 1 \quad (\forall k \in K) \\
& \quad \sum_{i \in C} \sum_{j \in C, j \neq i} x_{(i,j)^p k} \leq Q \quad (\forall k \in K) \\
& \quad t_i' = t_i + w_i + s_i \\
& \quad w_j = \begin{cases} 
\max\{e_j - t_j' - t_{ij}, 0\} & \text{if } j \in C \\
0 & \text{if } j \notin C
\end{cases} \quad (\forall i,j \in V) \\
& \quad t_i + s_i + t_{ij} + w_j \leq t_j \forall i \in V, j \in C, i \neq j \\
& \quad e_i \leq t_i + w_i \leq l_i \quad (\forall i \in C)
\end{align*}
\]
1) Constructing a solution by hybrid BSO-ACO according to the current static customer data;
2) When there is a new customer request or a canceled customer, execute the following steps
   a) Removing the canceled customer from the vehicle routes;
   b) Inserting the new customer into an existing vehicle route (or sends a new vehicle for service), and the unserviceable new customer is added to the reject service queue;
3) Re-optimizing vehicle routes again using the hybrid BSO-ACO;
4) Repeating the above process until there is no new or canceled request

A. NEIGHBORHOOD SEARCH
Neighborhood Search is a popular algorithm to solve VRPs [16], [17], [18], [19], [24], [38]. Neighborhood search refers to finding a better solution by exploring several solutions within the neighborhood of the current solution \( s \). In the VRPs, improved heuristic methods are often used to construct the neighborhood of the current solution. In this section, the 2-opt, exchange, and relocate operations are introduced.

2-opt [13] is an algorithm that is widely used for neighborhood construction. The algorithm was first proposed by Croes and used to solve the traveling salesman problem (TSP) [13]. The main idea of the algorithm is to eliminate the extra distance traveled by a detour. The algorithm then removes the intersecting edges in the routes and reverses the nodes between the intersecting edges. Unlike TSPs, which contain only a single route, in the VRPs, 2-opt has two operations, one acts within a single route, and the other acts between two routes. 2-opt’s two operations are shown in Fig. 3 and Fig. 4, respectively.

The exchange customers and relocate customers operations were first proposed by Savelsbergh [39], by which a neighborhood is constructed and sub-route segments in a route can be exchanged between routes, or relocate customers to other vehicles. The two operations of exchanging customers and relocating customers are shown in Fig. 5 and Fig. 6, respectively.

B. ANT COLONY SYSTEM
Ant Colony Optimization (ACO) [40] is a swarm intelligence algorithm that simulates the foraging behavior of ants. In this algorithm, ants leave pheromones on the routes they traveled. At the same time, the ants on the route with shorter total

\[
\text{min TD} = \sum_{i \in T} \sum_{k \in K} \sum_{(i,j) \in A_i^k} c_{L(i,j)} x_{(i,j)p_k} 
\]

\[
\text{min RN} = \sum_{i \in T} R_{N_i} 
\]

subject to:

\[
\sum_{i \in V} \sum_{p \in A_{i,j}} x_{(i,j)p_k} = \sum_{j \in V} \sum_{p \in A_{d_{j,i}}} x_{(d_{i,j})p_k} = 1 \quad (\forall k \in K) 
\]
distance travel more frequently, so the pheromone on the road will be more concentrated. This foraging mechanism relying on pheromones can help the ant colony find a shorter path from the nest to the food.

The process of solving the VRPs by the ACO is divided into route construction and pheromone updating. The route construction process is as follows. The ACO uses $M$ ants to search routes. Ant $k$ calculates the probability $p_{ij}^k$ corresponding to the $(i, j)$ route on each road node $i$ according to the Eq. (12), where $\tau_{ij}$ is the pheromone concentration on the road from node $i$ to node $j$, $\alpha$ is the parameter that controls the importance of pheromone, $\eta_{ij}$ denotes the degree of transfer expectation (usually set to $1/d_{ij}$), $\beta$ is the parameter used to control the influence of the degree of transfer expectation $\eta_{ij}$, and $J_k(i)$ is the set of nodes reachable by ant $k$ at node $i$.

$$p_{ij}^k = \frac{(\tau_{ij}^\alpha) \cdot (\eta_{ij}^\beta)}{\sum_{j \in J_k(i)} (\tau_{ik}^\alpha) \cdot (\eta_{ik}^\beta)} \quad j \in J_k(i)$$

After all ants have traversed all customers nodes, the pheromones of the edges between the nodes are updated according to the performance of each solution. The formula for the update is shown in Eq. (13), where $\rho$ is the evaporation coefficient of the pheromone and $\Delta \tau_{ij}^k$ is the pheromone secreted by ant $k$, defined by Eq. (14), and $L_k$ denotes the fitness value (usually the total distance traveled) of the solution found by ant $k$.

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

$$\Delta \tau_{ij}^k = \begin{cases} 1/L_k & \text{if } k \text{ use road}(i,j) \\ 0 & \text{otherwise} \end{cases}$$

C. BRAIN STORM OPTIMIZATION

Brain Storm Optimization (BSO), proposed by Shi [41] in 2011, is a swarm intelligence algorithm that simulates the process of brainstorming. The classical BSO algorithm uses the k-means algorithm to cluster solutions, which brings a lot of time cost, so Shi proposed a BSO algorithm based on objective space clustering [42]. The algorithm clusters solutions on the objective space. This method first sorts solutions by fitness values, and then divides solutions into two groups, one for the better elitists and one for the worse normals. The steps of the BSO in objective space are as follows.

1) Initialization phase: randomly generate $N$ individuals (solutions);
2) Evaluation phase: evaluate the fitness value of each solution;
3) Clustering phase: classify the individuals into two classes using a clustering algorithm, where the better $p_{elitists}$ are elitists and the rest are normals;
4) New solution phase generation phase: Generate random number $r \in (0, 1)$

Algorithm 1 Hybrid BSO-ACO Algorithm

1: initialize $n$ solutions by ACO
2: while not terminatd do
3: solutions $S$ are divided into two categories according to the fitness value: the better $p_{elitists}$ percentage are as enlist and the rest as normals;
4: if $\text{rand(0, 1)} < p_{elitists}$ then
5: if $\text{rand(0, 1)} < p_{one}$ then
6: randomly select $s_i$ from elitist
7: $s_i' \leftarrow \text{neighborhood_search}(s_i)$
8: else
9: randomly select $s_i$ and $s_j$ from normals
10: $s_i' \leftarrow \text{ACO}(s_i, s_j)$
11: end if
12: else
13: if $\text{rand(0, 1)} < p_{one}$ then
14: randomly select $s_i$ from normals
15: $s_i' \leftarrow \text{neighborhood_search}(s_i)$
16: else
17: randomly select $s_i$ and $s_j$ from normals
18: $s_i' \leftarrow \text{ACO}(s_i, s_j)$
19: end if
20: end if
21: if $s_i'$ is better than $s_i$ then
22: $s_i \leftarrow s_i'$
23: end if
24: end while
25: return $s_{best}$ from solutions $S$

| Algorithm 1 Hybrid BSO-ACO Algorithm |
|--------------------------------------|
| 1: initialize $n$ solutions by ACO |
| 2: while not terminatd do |
| 3: solutions $S$ are divided into two categories according to the fitness value: the better $p_{elitists}$ percentage are as enlist and the rest as normals; |
| 4: if $\text{rand(0, 1)} < p_{elitists}$ then |
| 5: if $\text{rand(0, 1)} < p_{one}$ then |
| 6: randomly select $s_i$ from elitist |
| 7: $s_i' \leftarrow \text{neighborhood_search}(s_i)$ |
| 8: else |
| 9: randomly select $s_i$ and $s_j$ from normals |
| 10: $s_i' \leftarrow \text{ACO}(s_i, s_j)$ |
| 11: end if |
| 12: else |
| 13: if $\text{rand(0, 1)} < p_{one}$ then |
| 14: randomly select $s_i$ from normals |
| 15: $s_i' \leftarrow \text{neighborhood_search}(s_i)$ |
| 16: else |
| 17: randomly select $s_i$ and $s_j$ from normals |
| 18: $s_i' \leftarrow \text{ACO}(s_i, s_j)$ |
| 19: end if |
| 20: end if |
| 21: if $s_i'$ is better than $s_i$ then |
| 22: $s_i \leftarrow s_i'$ |
| 23: end if |
| 24: end while |
| 25: return $s_{best}$ from solutions $S$ |

| TABLE 1. Parameters setting. |
|-------------------------------|
| Parameter | Value | Parameter | Value |
| $M$ | 20 | $p_{elitists}$ | 0.1 |
| $\beta$ | 2 | $p_{one}$ | 0.2 |
| $p_{one}$ | 0.6 | $\text{max}_i$ | 150 |
| $\rho$ | 0.1 | $\alpha$ | 1 |

- a) If $r < p_{elitists}$, generate random number $r_{one} \in (0, 1)$.
  i) If $r_{one} < p_{one}$, generate a new individual based on a randomly selected individual from the elitists
  ii) If $r_{one} \geq p_{one}$, generate a new individual based on two randomly selected individuals from elitists

- b) If $r \geq p_{elitists}$, generate random number $r_{one} \in (0, 1)$.
  i) If $r_{one} < p_{one}$, generate a new individual based on a randomly selected individual from normals
  ii) If $r_{one} \geq p_{one}$, generate a new individual based on two randomly selected individuals from normals
5) Selection phase: compare the newly generated solution with solutions having the same index and keep the better solution.

6) If \( N \) new solutions are generated, go to step 7; otherwise, go to step 4.

7) Terminate the program if the maximum number of iterations is reached, otherwise, go to step 2).

**D. HYBRID BSO-ACO**

Because the ACO relies on the accumulation of pheromones, it often falls into a local optimum. Therefore, this paper proposes a hybrid algorithm based on BSO and ACO. By using the BSO as the core framework, the ACO and neighborhood search algorithm are combined with each other to solve the drawback that the ACO is prone to fall into local optimum. The hybrid algorithm uses the BSO as the core framework, which consists of three main parts: 1) clustering, which classifies all individuals (solutions) into different classes; 2) learning the advantages of two different solutions by the ACO to generate a new solution; and 3) perturbing a single solution by neighborhood search to expand the search range to improve the quality of the solution. In part 2) hybrid BSO-ACO first zeroes the pheromone of ACO, then initializes the pheromone of ACO by two individuals (solutions) in the same class in the BSO algorithm, and finally obtains the new solution by the ACO algorithm. Because the clustering operation in the BSO brings similar individuals together, the ACO in part 2) can learn common properties in the same class to improve the convergence speed of the algorithm. In addition, this hybrid algorithm uses neighborhood search for solution perturbation to improve the solution quality by exploring different regions within the solution space. More details about the hybrid population intelligence algorithm are in the algorithm 1.

**IV. EXPERIMENTS AND DISCUSSIONS**

**A. BENCHMARK**

In our experiments, we use real road networks in Panyu District, Guangzhou, China from OpenStreetMap. The road networks contain a total of 13916 road nodes and 32,439 road edges. Similar to the classic Solomon benchmark (VRPTW), we generate 30 sets of test data sets based on this road network graph, including C1, C2, R1, R2, RC1, RC2. All instances contain 100 customers, respectively. The customers in R instances are obtained by randomly selecting nodes on the road network, while the customers in C instances are clustered. Half of the customers in the RC are randomly distributed, and the other half are clustered. In class 2 instances (i.e., R2, C2, and RC2), the scheduling horizon is longer since the high capacity of vehicles, and the time window is relaxed. We select 10%, 30%, 50%, 70% of the data from each instance as dynamic data, and 1/3 of the dynamic data is customer cancellation, 2/3 is customer new demand. More information can be obtained at https://github.com/lmingde/dvrptw-road-network. To validate the performance of our algorithm, we compare our...
algorithm with the state-of-art-algorithms, which used in DVRPTW : GVNS [43], and ALNS [33]. We compare algorithms base on the total distance.

**B. EXPERIMENT SETUP**

The proposed algorithm was programmed in the Python programming language, and all our experiments were executed
on a machine with the following configuration: Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz with 64GB RAM.

C. RESULT ANALYSIS

In this paper, we choose to use the total vehicle distance as the optimization objective, and choose the evaluation function (15) to represent the gap (Gap) between the hybrid algorithm and the other algorithms. A negative value of gap indicates that the hybrid algorithm is better than other algorithms, and a positive value of gap indicates that the hybrid algorithm is inferior to other algorithms. To ensure the statistical significance of the comparison experimental results, each algorithm is run 10 times for each test case, and each algorithm is limited to 2 minutes. The averaged results of the 10 comparison experiments are shown in Table 2, where “Instance” represents the test case, “D” represents the dynamic rate, “NV” represents the number of vehicles, “TD” represents the total distance traveled, “RN” represents the number of refuse customers, “GVNS-Gap” represents the difference between the hybrid BSO-ACO algorithm and ACO, and “ALNS-Gap” represents the difference between the hybrid BSO-ACO algorithm and ALNS.

\[
\text{Gap} = \frac{TD_{ours} - TD_{others}}{TD_{ours}} \quad (15)
\]

where \(TD_{ours}\) is the total distance of the Hybrid BSO-ACO algorithm, and \(TD_{others}\) is the total distance of other algorithms.

Table 2 shows the comparison of the results of the three methods. It can be seen that the average value of the number of vehicles in our method is 13.36, which is better than GVNS (14.28) and ALNS (13.70). The average value of total distance in our method is 443778.17, which is better than GVNS (450341.04) and ALNS (456805.79). The Hybrid BSO-ACO algorithm outperforms the GVNS for most instances. The total distance of Hybrid BSO-ACO is on average 2.01% lower than that of GVNS. In the 124 instances in the table, our method obtains 16 optimal (66%) and comparable values for denial of service. These results means that, under the condition of lower rate of refuse customers, our method can use fewer vehicles and lower total distance.

We can see that the RN is very close to 0 in the class 2 instances (i.e., C2, R2, RC2), i.e., almost all customer requests are served by the vehicles. This is due to the larger capacity of vehicles in the class 2 instances and the more relaxed time window in these instances. The larger capacity of vehicles the more customers can be served, and therefore the vehicle routes are longer in this type of problem. As shown in Table 3, the hybrid BSO-ACO algorithm performs significantly better in class 2 instances.

Fig. 7 is our method solution of C201 (10% dynamic rate) with 10% dynamic customers. The red triangle is the new customer node, the green triangle is the canceled customer node, the square is the completed customer node, the star is the depot, and the rest of the nodes are the customer nodes need to be served. Table 4 is the vehicle routes of C201 (10% dynamic rate) by our method at each time slice, where the bold parts are the nodes where the vehicles have been driven, the green parts are the customers that need to be canceled, and the red parts are the new customers.
REFERENCES

[1] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” Manage. Sci., vol. 6, no. 1, pp. 80–91, 1959.

[2] J. Caceres-Cruz, P. Arias, D. Guimarams, D. Riera, and A. A. Juan, “Rich vehicle routing problem: Survey,” ACM Comput. Surv., vol. 47, no. 2, pp. 1–28, 2014.

[3] Y. Huang, L. Zhao, T. Van Woensel, and J.-P. Gross, “Time-dependent vehicle routing problem with path flexibility,” Transp. Res. B: Methodol., vol. 95, pp. 169–195, Jan. 2017.

[4] A. N. Letchford, S. D. Nasiri, and A. Oukil, “Pricing routines for vehicle routing with time windows on road networks,” Comput. Oper. Res., vol. 51, pp. 331–337, Nov. 2014.

[5] B. Boyaci, T. H. Dang, and A. N. Letchford, “Vehicle routing on road networks: How good is Euclidean approximation?” Comput. Oper. Res., vol. 129, May 2021, Art. no. 105197.

[6] Y. Yao, T. Van Woensel, L. P. Veelenturf, and P. Mo, “The consistent vehicle routing problem considering path consistency in a road network,” Transp. Res. B: Methodol., vol. 153, pp. 21–44, Nov. 2021.

[7] Z.-L. Chen and H. Xu, “Dynamic column generation for dynamic vehicle routing with time windows,” Transp. Sci., vol. 40, no. 1, pp. 74–88, Feb. 2006.

[8] E. Choi and D.-W. Tcha, “A column generation approach to the heterogeneity fleet vehicle routing problem,” Comput. Oper. Res., vol. 34, no. 7, pp. 2080–2095, Jul. 2007.

[9] M. Fink, G. Desaulniers, M. Frey, F. Kiermaier, R. Kolisch, and F. Soumis, “Column generation for vehicle routing problems with multiple synchronization constraints,” Eur. J. Oper. Res., vol. 272, no. 2, pp. 699–711, 2019.

[10] S. Ropke and D. Pisinger, “A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with time windows,” J. Heuristics, vol. 18, no. 3, pp. 335–346, 2012.

[11] N. A. Kyriakakis, M. Marinaki, and Y. Marinakis, “A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem,” Comput. Oper. Res., vol. 34, no. 2, pp. 151–163, Feb. 2012.

[12] M. Liu, Y. Shen, Q. Zhao, and Y. Shi, “A hybrid BSO-ACS algorithm for vehicle routing problem with time windows on road networks,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jul. 2020, pp. 1–8.

[13] S. Poikonen, B. L. Golden, and E. Wasil, “A branch-and-bound algorithm for the pickup and delivery problem with time windows,” 1989.

[14] M. Liu et al.: Hybrid BSO-ACO for Dynamic Vehicle Routing Problem on Real-World Road Networks
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, Jan. 2004.

Y. Shi, “Brain storm optimization algorithm,” in *Advances in Swarm Intelligence* (Lecture Notes in Computer Science), vol. 6728, Y. Tan, Y. Shi, Y. Chai, and G. Wang, Eds. Berlin, Germany: Springer, 2011, doi: 10.1007/978-3-642-21515-5_36.

Y. Shi, “Brain storm optimization algorithm in objective space,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 1227–1234.

J. de Armas and B. Melián-Batista, “Variable neighborhood search for a dynamic rich vehicle routing problem with time windows,” *Comput. Ind. Eng.*, vol. 85, pp. 120–131, Jul. 2015.

MINGDE LIU received the B.E. degree from the Hebei University of Technology, China, in 2018, and the M.S. degree from the Harbin Institute of Technology, China, in 2020. He is a Researcher with the China Telecom Research Institute, Guangzhou, China. His research interests include swarm intelligence, evolutionary computation, artificial intelligence, machine learning, and dynamic optimization.

QI SONG was born in Jiangxi, China, in 1992. He received the B.E. degree from Nanchang University, China, in 2014, and the M.S. degree from the University of Electronic Science and Technology of China, China, in 2017. He is a Researcher with the China Telecom Research Institute, Guangzhou, China. His research interests include artificial intelligence and machine learning.

QI ZHAO received the Ph.D. degree in management science and engineering from the Beijing University of Technology, Beijing, China, in 2019. He was a Joint Ph.D. Student in computer science at the University of New South Wales, Canberra, Australia, from 2017 to 2018. He is a Research Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology. His research interests include evolutionary computation, automated algorithm design, and operations research.

LING LI received the B.S. degree in computer science from Hunan University, Changsha, China, in 1996, and the M.S. degree in computer engineering from the Guangdong University of Technology, Guangzhou, China, in 1999. From 1999 to 2022, he was a Researcher with the China Telecom Research Institute, Guangzhou. His research interests include modeling methodology and algorithm analysis and design.

ZHIMING YANG received the B.E. degree from the Huazhong University of Science and Technology, China, in 1995, and the M.S. degree from the South China University of Technology, China, in 1998. More than 20 years of experience in research and development, construction, and operation of large-scale intelligent operating systems. He is currently a Senior Researcher with the China Telecom Research Institute, Guangzhou, China. With Long-Term Research Development Experience, he has published a monograph “Technology and Practice of Operation and Maintenance Support System”. His research interests include intelligent networks, intelligent operation and maintenance, and machine learning.

YINGBIN ZHANG received the B.E. degree from the Sun Yat-sen University of Technology, China, in 2007, and the M.S. degree from the South China University of Technology, China, in 2010. He is a Researcher with the China Telecom Research Institute, Guangzhou, China. He is the Director of the Product and Service Research and Development Center, Cloud and Network Collection and Control Technology Development Department. His research interests include OSS planning and architecture design, artificial intelligence, machine learning, and dynamic optimization.