IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 September 2023, accepted 23 October 2023, date of publication 30 October 2023, date of current version 3 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3328404

RESEARCH ARTICLE

A Hybrid Brain Storm Optimization Algorithm for Dynamic Vehicle Routing Problem With Time Windows

MINGDE LIU[1], QI ZHAO[2], QI SONG[1], AND YINGBIN ZHANG[1]
[1]China Telecom Research Institute, Guangzhou 510630, China
[2]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
Corresponding author: Mingde Liu (rosandthree@gmail.com)

ABSTRACT The vehicle routing problem (VRP) holds significant applications in logistics and distribution scenarios. This paper presents a hybrid brain storm optimization (BSO) algorithm for solving the dynamic vehicle routing problem with time windows (DVRPTW). The proposed hybrid BSO algorithm effectively addresses the dynamic emergence of new customers and minimizes the number of unserved customers by utilizing the repeated insertion algorithm. Furthermore, the algorithm uses BSO clustering operations to classify vehicle routes and facilitates mutual learning within and between classes through λ-interchange. The intra-class similarity expedites solution convergence, while the inter-class difference expands the search space to avoid local optima. Finally, the quality of the solution is enhanced through the application of the 2-opt operation. To evaluate its performance, we compare the proposed algorithm with state-of-the-art algorithms using Lackner's benchmark. The experimental results demonstrate that our algorithm significantly reduces the number of unserved customers.

INDEX TERMS Brain storm optimization, dynamic vehicle routing problem with time windows, repeated insertion.

I. INTRODUCTION

The vehicle routing problem (VRP) is a classical combinatorial optimization problem that was first proposed by George Dantzig in 1959 [1]. The VRP aims to find the optimal routes for a fleet of vehicles to service a set of customers, subject to a variety of constraints, such as capacity limitations (capacity vehicle routing problem, CVRP) and time windows (vehicle routing problem with time windows, VRPTW). Because its practical applications in transportation, distribution, and delivery planning, the VRP holds immense significance in the fields of operations research and logistics. The study of VRP is the key to building smart logistics systems which can lead to reduced transforming costs, improved customer satisfaction, and increased profits.

The VRP has been extensively investigated, resulting in the development of various solution methods, including exact methods, heuristic methods, and metaheuristic methods. Exact methods aim to find the optimal solution for the VRP and typically based on mathematical programming techniques such as integer programming [2], dynamic programming [3], branch-and-bound [4], and branch-and-price [5]. While exact methods guarantee the discovery of the optimal solution, they are computationally intensive and primarily suitable for small-scale problem instances. On the other hand, heuristic methods are approximate algorithms designed to find high-quality solutions to the VRP within a reasonable amount of time. Heuristics often employ constructive approaches, such as nearest neighbor insertion [6] and the saving algorithm [7], or local improvement approaches, such as 2-opt [8], chain-exchange [9], and λ-interchange [10]. Although heuristics are faster than exact methods, they do not guarantee optimality. Lastly, metaheuristic methods are high-level search algorithms that utilize heuristic information to guide the exploration of the search space. These methods are often population-based and incorporate local or global search strategies. Commonly used metaheuristic methods for VRPs include genetic algorithms

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

(GA) [11], ant colony optimization (ACO) [12], [13], tabu search (TS) [14], simulated annealing (SA) [10], and brain storm optimization (BSO) [15], [16].

In contrast to the VRP, which assumes a fixed set of customer requests and predetermined vehicle routes, the dynamic vehicle routing problem (DVRP) allows for real-time adjustments to vehicle routes to accommodate dynamically changing customer requests or traffic conditions. This flexibility enables more efficient and cost-effective goods delivery, as routes can be optimized in response to changing circumstances. DVRPs are more realistic and practical in modern world, where information technology has facilitated easy access to real-time information. By enabling logistics companies to adapt vehicle routes based on changing circumstances, they can effectively reduce transportation costs, enhance customer satisfaction, and improve overall delivery efficiency.

The DVRP is a more complex problem than the VRP since DVRP needs to deal with real-time, dynamically changing customer demands and road network traffic conditions. An important variant of DVRP is the dynamic vehicle routing problem with time windows (DVRPTW), which adds the constraint of time windows to the problem. This means that customers have specific time periods during which they can be serviced. The DVRPTW must schedule the delivery of goods within these time windows while also minimizing transportation costs. Gendreau et al. [17] first studied the DVRPTW inspired by courier service applications, proposing a tabu search heuristic adapted for the dynamic case and implemented on a parallel platform. Chen and Xu [18] presented a column-generation-based dynamic approach for the problem, generating single-vehicle trips in real-time using existing columns. The approach outperforms an insertion-based heuristic on most test problems. Hong [19] proposed a solving strategy and algorithm for the DVRPTW. The problem is decomposed into a series of static VRPTW using an event-trigger mechanism. An improved large neighborhood search algorithm is used to solve the static problem and efficiently merge the latest requirements into the current solution. Computational results demonstrate the effectiveness of the proposed method on various test problems. Uchi et al. [20] presented an ACO algorithm based on the 2-opt local search to solve the DVRP with dynamic pickup and delivery. Pillac et al. [21] proposed a fast-reoptimization approach based on parallel adaptive large neighborhood search (pALNS) to tackle the DVRPTW. Computational results show that it achieves state-of-the-art results in total distances objective and improves upon previous approaches by up to 12%. Silva Junior et al. [22] proposes a hybrid algorithm for solving the DVRPTW. The hybrid algorithm combines multiple ant colony systems with a random variable neighborhood descent, achieving competitive results in minimizing the number of unserved customers compared to the state-of-the-art. Sabar et al. [23] presented a population-based approach to tackle the DVRP. The approach combines a local search algorithm with evolutionary operators, utilizing a population of solutions and a quality-and-diversity strategy to retain promising solutions. Computational results showed the approach has excellent performance.

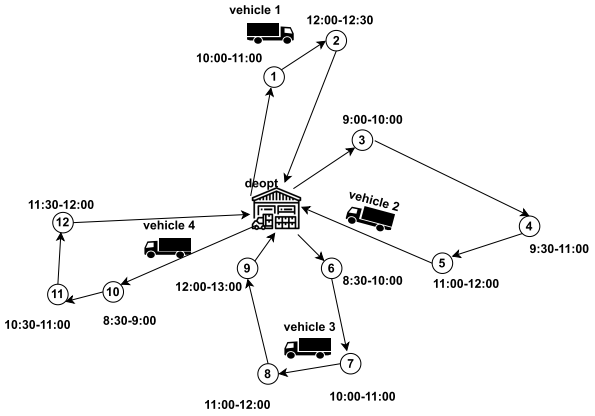

FIGURE 1. An instance of VRPTW.

The majority of research on DVRPTW has primarily concentrated on minimizing transport costs, specifically the total distance traveled by the vehicles, while often neglecting the goal of serving as many customers as possible. Consequently, customer satisfaction has not been adequately addressed. Based on this, this paper aims to minimize the number of unserved customers in a DVRPTW scenario. The main contributions of this study are as follows:

- A hybrid BSO algorithm is proposed, which serves as many customers as possible by repeated insertion algorithm and uses the BSO algorithm to optimize the updated vehicle routes.
- 280 DVRPTW instances are used for comparison experiments and by comparing the proposed algorithm with state-of-the-art approaches. The experimental results demonstrate the excellent effectiveness of the proposed algorithm.

II. PROBLEM DESCRIPTION

In this section, we first introduce the VRPTW, followed by a presentation of a corresponding mathematical model. Subsequently, we introduce the DVRTW and provide its formal problem definition.

A. VPRTW

The VRPTW is a significant variant of the VRP. It entails determining the optimal set of routes for a fleet of vehicles intended to serve a group of customers within specified time windows. The objective is to minimize the total distance traveled by vehicles, concurrently satisfying time window constraints and vehicle capacity limits. The VRPTW can be defined on a complete graph $G(C, E)$, where $C = \{c_0, c_1, \ldots, c_n\}$ denotes a set of customers and c_0 represents the depot. $E = \{(c_i, c_j)|c_i, c_j \in C, i \neq j\}$ signifies the edges connecting customers. In the context of VRPTW, the depot hosts K vehicles, each with a capacity of Q and a

speed of v. Every customer is associated with a service time window, denoted as $[e_i, l_i]$, and a demand q_i that indicates the weight of goods to be transported. When a vehicle k is scheduled to serve a customer, it is crucial to ascertain that the vehicle's capacity Q is not less than the total of the q_i values corresponding to the customers being served. Vehicle k must service customer c_i within the time window $[e_i, l_i]$. If the vehicle arrives before e_i, it must wait until e_i, wherein the waiting time is w_i. The mathematical model for VRPTW is defined as follows:

Parameters description:

K	the set of vehicles
C	the set of customers
v	the speed of each vehicle
Q	the capacity of each vehicle
c_{ij}	the distance cost between c_i and c_j
t_{ij}	the travel time between c_i and c_j
q_i	the demand of c_i
e_i	the earliest time c_i can be serviced
l_i	the latest time c_i can be serviced
s_i	the time required for servicing c_i
t_i	the time for the vehicle to arrive at c_i
w_i	the waiting time of a vehicle at c_i

Objective function:

$$\min_{x} \sum_{k\in K}\sum_{i\in C}\sum_{j\in C} c_{ij}x_{ijk} \tag{1}$$

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

$$\sum_{i\in C} x_{i0k} = \sum_{j\in C} x_{0jk} = \begin{cases} 1 & \text{if vehicle } k \text{ used} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

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

$$\sum_{i\in C} q_i \sum_{j\in C, j\neq i} x_{ijk} \leq Q \quad (\forall k \in K) \tag{5}$$

$$w_j = max\{e_j - t_i - t_{ij},\ 0\} \quad (\forall i, j \in C, i \neq j) \tag{6}$$

$$e_i \leq t_i + w_i \leq l_i \quad (\forall i \in C) \tag{7}$$

The objective function (1) is the total distance traveled by vehicles. Eq. (2) denotes a binary variable indicating whether the vehicle k travels from customer i to customer j. Constraint (3) represents the vehicle needing to depart from the depot c_0 and return to the depot after completing the service. Constraint (4) means that each customer can be serviced only once by one vehicle. Constraint (5) indicates the maximum vehicle capacity limit, where $t_{ij} = c_{ij}/v$. Constraint (6) is the wait time of the vehicle at customer j. Constraint (7) is the time window constraint for customer i.

Figure 1 provides an example of the VRPTW, Where each customer has an associated time window and four vehicles are dispatched from the depot, with each vehicle serving its customers within their requested time windows.

B. DVRPTW

The DVRPTW can be characterized as follows. A logistics company's distribution center schedules vehicles for customer service. However, while these vehicles are en route, new customer demands may emerge. Consequently, the distribution center must either re-schedule the existing vehicles or deploy additional ones to accommodate these newly emerging customer requests. The aim is to dynamically adjust vehicle routes to serve as many customers as possible while minimizing transportation costs.

Given the constraints of the customer service window, there is a risk that new customers in DVRPTW may not be served. Therefore, in DVRPTW, it is also necessary to serve as many customers as possible, i.e. mini the number of unserved customers. Let the number of unserved customers at each moment t be UC_t, and the set of all moments in a day be T, then the objective to be optimized is the total number of unserved customers in a day, as shown in Eq. (8).

$$\min UC = \sum_{t\in T} UC_t \tag{8}$$

Figure 2 illustrates an instance of DVRPTW. At 08:00, twelve customers require service and the distribution center has scheduled four vehicles to serve these customers from the depot. However, at 09:30, an additional customer (Customer 13) emerges, and the distribution center adjusts the route of Vehicle 2 to accommodate this customer. At 10:30, another customer (Customer 14) appears, prompting the distribution center to schedule a newly available Vehicle 5 to serve this customer, taking into consideration the service time window and the current availability of vehicles in transit. By 14:00, all vehicles have completed their service rounds and returned to the depot.

III. PROPOSED APPROACH

This section initially presents the repeated insertion algorithm, followed by an exploration of the 2-opt and λ-interchange algorithms, which are frequently used in VRPs. Finally, we elaborate on the BSO algorithm and the hybrid BSO algorithm.

A. REPEATED INSERTION ALGORITHM

When a new customer needs to be served by vehicles, the intuitive way is to insert the customer into the current vehicle routes or to arrange a new vehicle to serve the customer. But there exists a failure scenario: the new customer cannot be inserted into the current vehicle routes and the newly arranged vehicle cannot serve the customer in time too. In this case, if the vehicles on the delivery route are allowed to abandon some customers to serve the new customers, and the abandoned customers are served by the new vehicles. This approach may solve the problem that the insertion algorithm cannot handle. Based on this idea, this paper proposes a repeated insertion algorithm to reduce the number

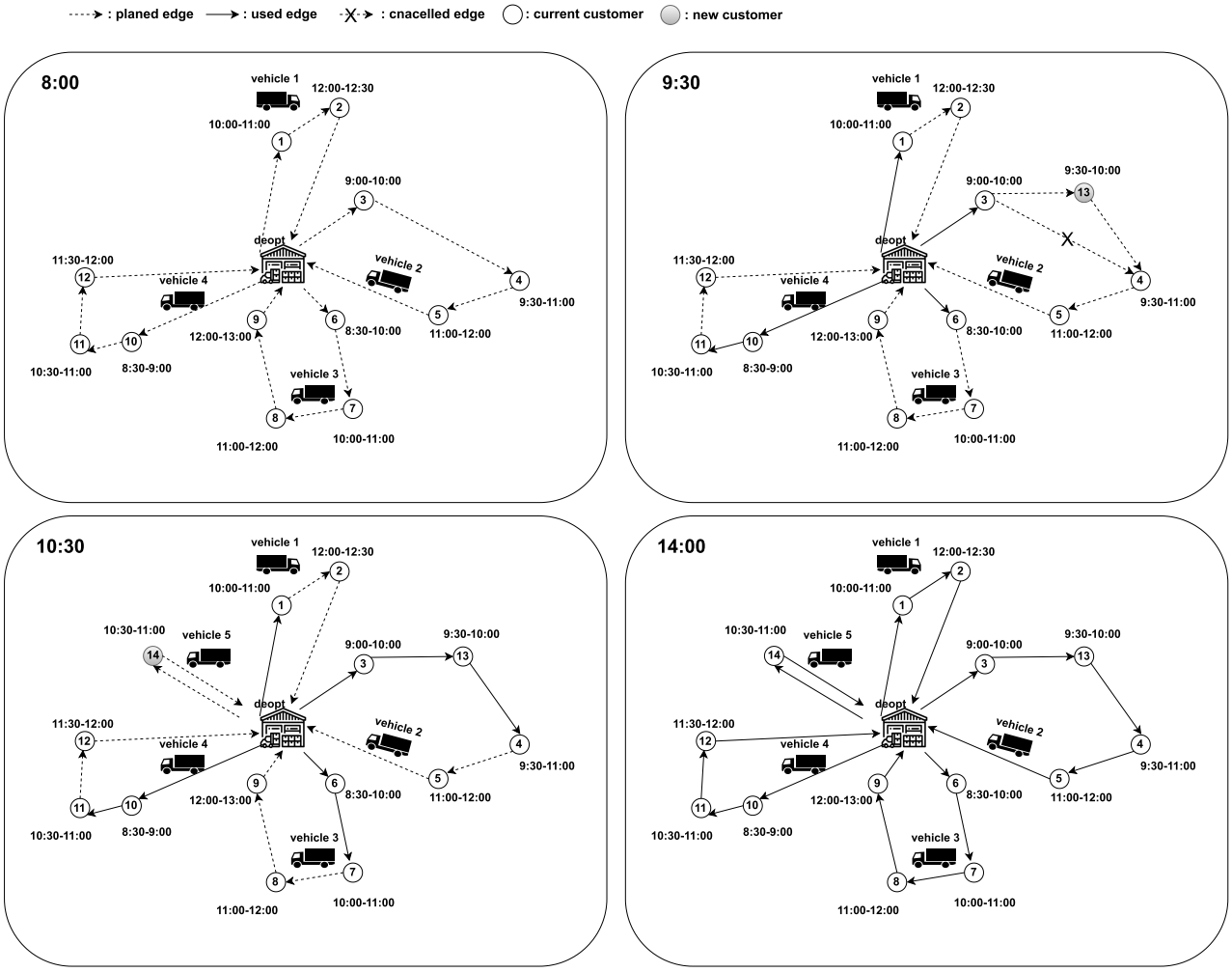

FIGURE 2. An instance of DVRPTW.

of unserved customers. The procedure of the algorithm is as follows.

1) Constructs the set of customers which need be served $C' = c$ when a new customer c appears.
2) Randomly select a customer demand c' from C'.
3) Try to insert c' into vehicle routes R or arrange a new vehicle to serve that customer.
4) If the customer c' cannot be served, a route r is randomly selected from current routes R.
   a) Calculate the urgency $u_i = l_i - t_i$ of all customers c_i in r by the arrival time of the vehicle t_i and the latest service time l_i of the customer c_i.
   b) Select the most urgent customers c_j (i.e., the smallest u_j) to remove from route r and move c_j to C', Then the vehicle has more time to serve other customers.
5) Stops the algorithm when C' is empty or the maximum number of iterations is reached, otherwise goto step 2).

B. 2-OPT AND λ-INTERCHANGE

The 2-opt algorithm, a prevalent local search method in combinatorial optimization, was initially applied to the traveling salesman problem (TSP) [8]. It operates by evaluating pairs of edges in the existing solution and implementing swaps to determine whether these alterations result in a shorter route. Due to its superior performance, this algorithm is extensively utilized in VRPs. A depiction of the 2-opt operation can be seen in Figure 3.

The λ-interchange algorithm [10] is a renowned neighborhood search strategy employed for addressing VRPs. The operation of the λ-interchange method entails selecting a pair of non-adjacent edges within a solution and interchanging them to generate an alternative feasible solution. This interchange holds the potential to enhance the solution by diminishing the total distance traveled. The λ parameter in the λ-interchange method regulates the magnitude of the neighborhood search, i.e. the length of the edge. In this study, the λ parameter is confined to a maximum value of 2, signifying that subroutes with an utmost length of 2 (equivalent to a maximum of two customers) can be interchanged. A visual representation of the λ-interchange procedure is depicted in figure 4.

C. BRAIN STORM OPTIMIZATION

Brain Storm Optimization (BSO) [24], [25] is a swarm intelligence algorithm that is inspired by the way a group

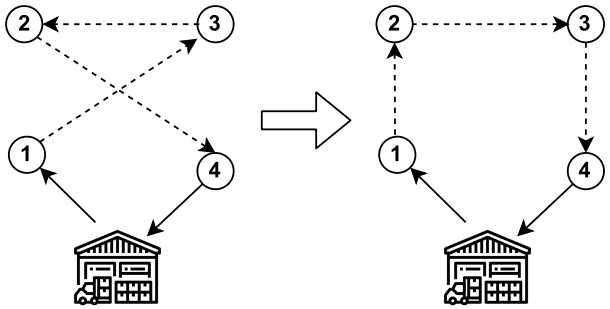

FIGURE 3. 2-opt.

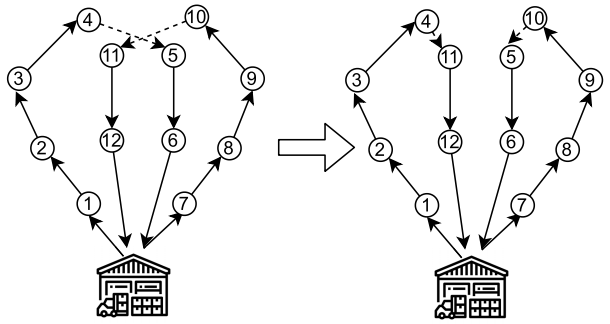

FIGURE 4. λ-interchange.

of people brainstorm to generate new ideas. Like other swarm intelligence algorithms, BSO uses a population-based approach, where a group of solutions is used to explore the solution space. The main idea of the algorithm is to divide the population into several groups by clustering. Individuals within a group learn each other's strengths to find excellent solutions quickly, and individuals in different groups learn each other's different characteristics to increase the randomness of the overall population in order to detect a wider solution space and thus jump out of the local optimum. BSO has been widely and successfully used to solve optimization problems [26], [27], [28]. The procedure of the BSO is as fellow.

1) Initialization: randomly generate n potential solutions (individuals), evaluate the n individuals, and set parameters $p_{one}, p_{cluster}$.
2) Clustering: n individuals are divided into m groups by clustering algorithm.
3) New individuals' generation: Generate random number $r_{one} \in (0, 1)$.
   a) If $r_{one} < p_{one}$, generate the random number $r_{cluster} \in (0, 1)$.
      i) If $r_{cluster} < p_{cluster}$, randomly select a group, and generate a new individual from the random two individuals in the group.
      ii) If $r_{cluster} \geq p_{cluster}$, randomly select two groups, randomly select one individual from each of these two groups, and generate new individuals from these two individuals.
   b) If $r_{one} \geq p_{one}$, randomly select an individual to generate a new individual.
4) Selection: the newly generated individuals are compared with the old individuals with the same index and the better ones are retained.
5) Terminate the program if the maximum number of iterations is reached, otherwise jump to step 2).

The BSO algorithm optimizes the solution as an individual, which corresponds to the set of vehicle routes in VRPs. The operation of generating solutions at this level is very time consuming and may generate infeasible solutions. Therefore, to apply the BSO algorithm in VRPs, this paper treats the vehicle route as an individual. The routes are first clustered into m groups based on the location of the route by using a clustering algorithm. The location of the route is obtained by computing the average of the geographical coordinates of all customer nodes. In the individual generation phase, new individuals are generated in three different ways: 1) using the λ-exchange algorithm to optimize two random routes in one group; 2) using λ-exchange for routes in two groups, and 3) using 2-op for a random route. The modified BSO algorithm uses clustering to classify geographically close routes into one group. The optimization of routes is accelerated by exchanging customers within groups during the generation of new routes. Customer nodes are exchanged between groups to try to avoid getting trapped in a local optimal solution by interacting with customers at a distance. The 2-opt algorithm is applied inside the routes to further optimize them. More details of the algorithm can be found in Alg. 1.

Algorithm 1 Modified BSO for VRPTW

```
Input: solution s (i.e. routes)
Output: new_solution s'
while not termination do
    cluster routes into m groups G = g_1, g_2, g_3, ..., g_m
    if rand(0, 1) < p_one then
        if rand(0, 1) < p_cluster then
            randomly select g_k from G
            randomly select r_i and r_j from g_k
            (r_i', r_j') ← λ-exhange(r_i, r_j)
        else
            randomly select g_k, g_l from G
            randomly select r_i from g_k and r_j from g_l
            (r_i', r_j') ← λ-exhange(r_i, r_j)
        end if
        if (r_i', r_j') is better than (r_i, r_j) then
            (r_i, r_j)←(r_i', r_j')
        end if
    else
        randmly select r_i from s
        r_i' ← 2-opt(s)
        if r_i' is better than r_i then
            r_i ← r_i'
        end if
    end if
end while
return s'
```

D. PROPOSED HYBRID BSO ALGORITHM

DVRPs require re-optimization of vehicle routes based on existing route information to serve new customers. There are two categories of re-optimization methods: periodic re-optimization and continuous re-optimization [29]. When a new customer appears, the continuous re-optimization strategy generates a new solution to serve the new customer [30]. In contrast, the periodic processing policy divides each workday into time slices, and then the strategy re-optimizes routes at the end of the time slice to serve the new customers within the time slice [31]. Compared to the periodic re-optimization strategy, the continuous processing strategy is more time-consuming because it is more frequent. However, since DVRPTW needs to consider the time window constraint, it is necessary to have a continuous re-optimization strategy to prevent customers from being unserved since delayed processing. Therefore, in this paper, we use the continuous re-optimization strategy to solve the DVRPTW.

The hybrid BSO algorithm proposed in this paper uses a continuous optimization strategy to solve the DVRPTW. In the beginning, the distribution center already has some static customers before the delivery vehicle serves the customer, so the distribution center uses the modified BSO algorithm to plan the vehicle routes and then dispatches the vehicle to serve the customer. New customers will appear when the vehicles are dispatched to serve the customers. When a new customer appears, a repeated insertion algorithm is used to insert the new customer into the current vehicle routes or to schedule a new vehicle to serve the customer. After processing the new customer, a modified BSO algorithm is used to optimize the changed vehicle routes. Details of the algorithm can be found in Alg. 2

IV. EXPERIMENTS AND DISCUSSIONS

This section evaluates the performance of the hybrid BSO algorithm in solving the DVRPTW through a series of comparative experiments.

A. BENCHMARK

For our experiments, we use Lackner's benchmark [32], which is widely used in the DVRPTW. It is modified from solomon benchmark [33]. The Solomon benchmark has a total of 56 instances, with six types (C1, R1, RC1, C2, R2, RC2), each containing 8-12 instances. Each instance in Solomon's benchmark contains 100 customers, each with location information, time window, and weight information. The customers in the type R (i.e. R1, R2) are geographically randomly distributed, while the customers in type C (i.e. C1, C2) are geographically aggregated (i.e. customers are distributed around some centroids) and the customers in type RC (i.e. RC1, RC2) is half randomly distributed and half aggregated. Compared to type 1 (i.e. R1, C1, RC1), the vehicles in type 2 have a larger capacity and the vehicles can serve more customers. Therefore, the vehicle routes in type 2 are longer. Lackner modifies some of the customers in Solomon's data into dynamically occurring customers, with five values of dynamic degree (i.e. 10%, 30%, 50%, 70%, 90%). Thus Lackner's benchmark has a total of 280 instances.

Algorithm 2 Hybrid BSO Algorithm

```
initialize solution s
s ← modified_BSO
while a new customer c_i appears do
    if c_i can be served by repeated insertion algorithm then
        s ← repeated_insertion(c_i)
        s' ←modified_BSO(s)
    else
        add c_i to unserved queue
    end if
end while
return s
```

B. EXPERIMENT SETUP

The algorithm was implemented using the Python programming language, and the experiments were executed on a system equipped with an Intel(R) Xeon(R) Gold 6248R CPU @ 3.00 GHz, and 64GB of RAM, running the Windows operating system.

The parameters in the experiment are set according to table 1, where p_{one} is used to control whether the algorithm learns combinations of individuals to accelerate convergence or perturbation of a single individual to expand the search range in BSO. Meanwhile $p_{cluster}$ is used to control whether the combinations of individuals learn each other in a single cluster or in different clusters. The number of clusters is chosen to be 4, considering that in the benchmark the number of customers is 100, and most of the time less than 20 vehicles are needed to serve the customers. These parameters are the maximum number of iterations to be run. All parameters are tuned to balance the solution quality and computational cost.

TABLE 1. Parameters setting.

Parameter	Value
p_{one}	0.7
$p_{cluster}$	0.6
the number of cluster	4
maximum number of iterations in the repeated insertion algorithm	100
maximum number of iterations in the modified BSO algorithm	200

C. COMPARISON WITH THE STATE OF ART APPROACHES

In this paper, the number of unserved customers is chosen as the optimization objective, which means serving as many customers as possible. To verify the performance of the hybrid BSO algorithm proposed in this problem, we compare the results of our algorithm in Lackner's benchmark with the results of published state-of-the-art research. These recent works are: Hong [19], Pillac et al. [21], Chen et al. [34] and Silva Junior et al.'s work [22]. Since the results given in these studies are the average of the results of each instance

TABLE 2. Comparison of the result obtained by hybrid BSO with state of art for DVRPTW.

Type	D	Hong [19]		Pillac et al. [21]		Chen et al. [34]		Silva Junior et al. [22]		Hybrid BSO	
		TD	UC	TD	UC	TD	UC	TD	UC	TD	UC
C1	10	895.77	0.22	**850.60**	0.11	1102.80	0.11	903.73	**0.00**	893.33	**0.00**
	30	962.08	0.33	**874.90**	0.11	1126.84	0.11	949.06	**0.00**	937.89	**0.00**
	50	1001.18	0.22	**903.40**	0.11	1096.58	0.11	986.60	**0.00**	981.63	**0.00**
	70	1031.68	0.22	**919.10**	0.11	1088.38	0.11	1001.14	**0.00**	1006.24	**0.00**
	90	1039.77	0.22	**929.90**	0.11	974.41	0.11	1052.80	**0.00**	1038.16	**0.00**
C2	10	**594.67**	**0.00**	597.20	**0.00**	624.06	**0.00**	598.85	**0.00**	601.90	**0.00**
	30	651.42	**0.00**	**597.60**	**0.00**	658.09	**0.00**	607.38	**0.00**	600.47	**0.00**
	50	604.98	**0.00**	**604.00**	**0.00**	650.70	**0.00**	644.24	**0.00**	607.52	**0.00**
	70	636.47	**0.00**	**619.20**	**0.00**	652.63	**0.00**	630.85	**0.00**	624.06	**0.00**
	90	636.79	**0.00**	**625.70**	**0.00**	668.45	**0.00**	657.79	**0.00**	631.14	**0.00**
R1	10	1257.08	0.17	**1197.40**	0.25	1309.10	0.58	1267.44	0.08	1293.90	**0.05**
	30	1286.63	0.58	**1212.90**	0.80	1310.23	1.50	1304.79	0.08	1331.44	**0.05**
	50	1295.81	0.67	**1225.00**	1.25	1313.35	2.17	1358.13	0.08	1395.05	**0.06**
	70	1331.34	1.75	**1237.30**	1.71	1298.86	2.83	1346.81	0.33	1423.67	**0.16**
	90	1335.94	2.33	**1230.10**	2.55	1270.20	3.92	1427.12	0.25	1434.39	**0.15**
R2	10	950.00	0.09	**893.00**	**0.00**	1080.50	**0.00**	996.97	**0.00**	942.05	**0.00**
	30	985.59	**0.00**	**915.60**	**0.00**	1087.52	**0.00**	1012.13	**0.00**	977.29	**0.00**
	50	1016.52	**0.00**	**948.60**	**0.00**	1086.09	**0.00**	1051.65	**0.00**	1017.01	**0.00**
	70	1032.04	0.09	**967.70**	**0.00**	1088.97	**0.00**	1051.02	**0.00**	1046.37	**0.00**
	90	1047.82	0.09	**981.70**	**0.00**	1076.77	**0.00**	1063.76	**0.00**	1049.18	**0.00**
RC1	10	1436.23	1.13	**1389.40**	0.04	1473.69	0.25	1465.01	**0.00**	1472.47	**0.00**
	30	1492.22	1.13	**1421.50**	0.28	1484.89	0.63	1508.58	0.13	1516.26	**0.12**
	50	1514.72	1.38	**1463.40**	0.23	1520.47	1.38	1573.11	**0.00**	1581.12	**0.00**
	70	1511.29	1.88	**1470.10**	0.58	1510.21	1.75	1599.18	**0.13**	1607.33	0.15
	90	1513.94	2.00	**1495.50**	0.51	1501.76	1.75	1605.36	0.25	1653.54	**0.13**
RC2	10	1103.30	**0.00**	**1024.40**	**0.00**	1253.11	**0.00**	1159.53	**0.00**	1065.44	**0.00**
	30	1166.04	0.25	**1053.10**	**0.00**	1238.82	**0.00**	1196.34	**0.00**	1102.43	**0.00**
	50	1190.54	0.13	**1060.50**	**0.00**	1260.66	**0.00**	1214.24	**0.00**	1150.42	**0.00**
	70	1239.46	**0.00**	**1091.40**	**0.00**	1261.81	**0.00**	1218.06	**0.00**	1204.26	**0.00**
	90	1257.19	0.13	**1130.30**	**0.00**	1264.94	**0.00**	1259.86	**0.00**	1236.28	**0.00**
SV		33018.51	15.01	**30930.50**	8.75	34334.89	17.31	33711.53	1.33	33422.24	**0.87**
GAP (%)		1.22	-94.20	8.06	-90.06	-1.82	-92.32	-0.86	-34.59		

TABLE 3. Summary of Comparison result between hybrid BSO and the state of art approaches.

Type	Hong [19]		Pillac et al. [21]		Chen et al. [34]		Silva Junior et al. [22]		Hybrid BSO	
	TD	UC	TD	UC	TD	UC	TD	UC	TD	UC
C1	4930.48	1.21	4477.90	0.55	5389.01	0.55	4893.33	0.00	4857.25	0.00
C2	3124.33	0.00	3043.70	0.00	3253.93	0.00	3139.11	0.00	3065.09	0.00
R1	6506.80	5.50	6102.70	6.56	6501.74	11.00	6704.29	0.82	6878.45	0.47
R2	5031.97	0.27	4706.60	0.00	5419.85	0.00	5175.53	0.00	5031.90	0.00
RC1	7468.40	7.52	7239.90	1.64	7491.02	5.76	7751.24	0.51	7830.72	0.40
RC2	5956.53	0.51	5359.70	0.00	6279.34	0.00	6048.03	0.00	5758.83	0.00

TABLE 4. Robustness of the results obtained by Hybrid BSO algorithm.

Type	Best		Average		Worst	
	TD	UC	TD	UC	TD	UC
C1	4781.33	0.00	4857.25	0.00	4915.70	0.00
C2	3060.50	0.00	3065.09	0.00	3111.39	0.00
R1	6984.28	0.40	6878.45	0.47	6861.23	0.51
R2	5014.45	0.00	5031.90	0.00	5139.50	0.00
RC1	7853.89	0.36	7830.72	0.40	7737.26	0.43
RC2	5660.39	0.00	5758.83	0.00	5889.73	0.00

run, the results of the proposed algorithm in this paper are also taken as the average of the results of the hybrid BSO algorithm runs. In this paper, the hybrid BSO algorithm is run ten times on each instance, with each run limited to 2 minutes. The algorithm runtime limit takes into account the setting of the maximum number of iterations parameter of the algorithm. With this runtime limit, most of the instances terminate prematurely, except for a few complex instances.

The comparison results are shown in table 2, where Type is the instance type, D is the percentage of dynamic customers, TD represents the total distance, UC represents the number of unserved customers, SV represents the sum of the column, and GAP represents the difference between the hybrid BSO algorithm and the other algorithms, calculated as shown in equation 9. The best part of the comparison result is bolded in the table.

$$GAP = \frac{SV_{ours} - SV_{others}}{SV_{others}} \tag{9}$$

In the table 2, the number of customers unserved by the hybrid BSO method totals 0.87, which is the smallest of all methods. In addition, the hybrid BSO finds nearly all minimum UC in all 30 sets of instances, only slightly larger than the results of Junior et al. [22] for one type of instances. In terms of GAP, the hybrid BSO significantly reduces the value of the UC, with a maximum reduction of 94.2%. Because more customers are served, the hybrid BSO theoretically requires more travel distance, and therefore the hybrid BSO does not have an advantage in terms of the total distances target. The table 2 shows that the results of Pillac et al. [21] are excellent in terms of the total distance objective, finding the smallest total distance (96.6%) on 29 of

TABLE 5. Comparison result of repeated insertion algorithm.

Type	D	Hybrid BSO without Repeated Insertion		Hybrid BSO	
		TD	UC	TD	UC
C1	10	885.33	0.11	893.33	0.00
	30	907.89	0.22	937.89	0.00
	50	961.63	0.11	981.63	0.00
	70	987.24	0.22	1006.24	0.00
	90	1008.16	0.11	1038.16	0.00
C2	10	601.90	0.00	601.90	0.00
	30	600.47	0.00	600.47	0.00
	50	607.52	0.00	607.52	0.00
	70	624.06	0.00	624.06	0.00
	90	631.14	0.00	631.14	0.00
R1	10	1231.90	0.28	1293.90	0.05
	30	1281.44	0.68	1331.44	0.05
	50	1285.05	0.98	1395.05	0.06
	70	1301.67	1.13	1423.67	0.16
	90	1344.39	1.25	1434.39	0.15
R2	10	942.05	0.00	942.05	0.00
	30	977.29	0.00	977.29	0.00
	50	1017.01	0.00	1017.01	0.00
	70	1046.37	0.00	1046.37	0.00
	90	1049.18	0.00	1049.18	0.00
RC1	10	1472.47	0.00	1472.47	0.00
	30	1471.26	0.31	1516.26	0.12
	50	1581.12	0.00	1581.12	0.00
	70	1573.33	0.23	1607.33	0.15
	90	1581.54	0.33	1653.54	0.13
RC2	10	1065.44	0.00	1065.44	0.00
	30	1102.43	0.00	1102.43	0.00
	50	1150.42	0.00	1150.42	0.00
	70	1204.26	0.00	1204.26	0.00
	90	1236.28	0.00	1236.28	0.00
TV		32730.24	5.96	33422.24	0.87
GAP (%)		2.11	-85.40		

the 30 sets of instances. Table 3 is the average value calculated by instance type. From table 3, it can be seen that the TD of the hybrid BSO is only slightly higher than that of Pillac et al. [21] when the UC is 0. This indicates that the hybrid BSO algorithm exhibits strong performance characteristics with respect to optimizing vehicle transportation costs.

D. ANALYSIS OF HYBRID BSO

Based on the comparison with other recent works, the hybrid BSO algorithm outperforms other algorithms. In order to examine the robustness of the algorithm, table 4 shows the best, average, and worst results of the hybrid BSO algorithm under different benchmark types. The UC is 0 under the worst result in C1, C2, R2, and RC2, i.e., there is no unserved customers. At this point the TD's will be considered as a target, so the smallest TD's are under the Best result. In R1 and RC1, at this point Best has the smallest UC but the largest TD, in contrast Worst has the largest UC but the smallest TD. This is due to the fact that the UC is larger and the vehicle does not need to travel longer distances to service more customers.

To analyze the effect of the repeated insertion algorithm in the hybrid BSO algorithm. In this paper, the hybrid BSO algorithm is compared with the hybrid BSO without the repeated insertion algorithm on the Lackner's benchmark. The hybrid BSO algorithm without the repeated insertion algorithm uses the insertion algorithm, i.e., when a new customer appears, it tries to insert it into the existing vehicle routes, or a new vehicle is scheduled to serve the customer.

Table 5 shows that when the vehicle capacity is large (i.e. instances of type C2, R2, RC2), all customers can be served using the insertion algorithm. When the vehicle capacity is small (i.e., C1, R1, RC1), the insertion algorithm does not serve all customers, and the repeated insertion algorithm significantly reduces the number of unserved customers. At the same time, the repeated insertion algorithm serves more customers and results in a increase in the total distance. It can also be seen that the average UC of the two algorithms is smaller for geographically concentrated instances of customers (i.e. C1, C2) than for geographically random instances (i.e. R1, R2). The table 1 shows that the UC of hybrid BSO algorithm is generally smaller than the UC of hybrid BSO algorithm without repeated insertion algorithm, which indicates that the repeat insertion algorithm is able to insert as much new customer as possible into the current vehicle routes, thus serving more customers and increasing customer satisfaction.

V. CONCLUSION

In this paper, the focus is on the analysis of the DVRPTW, with the number of unserved customers being the objective for optimization. A hybrid BSO algorithm is introduced, which reformulates the DVRPTW as a sequence of static problems contingent upon the dynamic presented of customers. The hybrid BSO algorithm employs a repeated insertion algorithm to accommodate newly arising customers into pre-existing vehicle routes or to allocate new vehicles for servicing the customers and consequent route updates. This repeated insertion algorithm aims to minimize the number of unserved customers by iteratively attempting customer insertions into vehicle routes. Following customer insertion, the hybrid BSO algorithm updates the vehicle routes and commences optimization to minimize transportation costs. Initially, the BSO clustering operation is utilized to segment vehicle routes into groups, which is followed by the implementation of a λ-interchange mechanism to facilitate mutual learning within and across groups. The homogeneity within groups expedites the convergence of solutions, while heterogeneity across groups enlarges the search space to evade local optima. The solution's quality is further refined through a 2-opt operation. A comparative analysis is conducted against state-of-the-art methodologies using a renowned benchmark dataset. Experimental outcomes reveal that, in one instance (1/30), the number of unserved customers obtained through the proposed approach is marginally higher compared to the results of Silva Junior et al. [22], with an enhancement ranging from 94.2% to 34.59% compare with alternative methods.

REFERENCES

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manage. Sci.*, vol. 6, no. 1, pp. 80–91, Oct. 1959.
[2] C. Çetinkaya, I. Karaoglan, and H. Gökçen, "Two-stage vehicle routing problem with arc time windows: A mixed integer programming formulation and a heuristic approach," *Eur. J. Oper. Res.*, vol. 230, no. 3, pp. 539–550, Nov. 2013.
[3] C. Novoa and R. Storer, "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands," *Eur. J. Oper. Res.*, vol. 196, no. 2, pp. 509–515, Jul. 2009.

[4] S. Poikonen, B. Golden, and E. A. Wasil, "A branch-and-bound approach to the traveling salesman problem with a drone," *INFORMS J. Comput.*, vol. 31, no. 2, pp. 335–346, Apr. 2019.
[5] M. Dell'Amico, G. Righini, and M. Salani, "A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection," *Transp. Sci.*, vol. 40, no. 2, pp. 235–247, May 2006.
[6] 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, Sep. 1977.
[7] 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, Aug. 1964.
[8] G. A. Croes, "A method for solving traveling-salesman problems," *Oper. Res.*, vol. 6, no. 6, pp. 791–812, Dec. 1958.
[9] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Oper. Res.*, vol. 21, no. 2, pp. 498–516, Apr. 1973.
[10] I. Osman, "Capacitated clustering problems by hybrid simulated annealing and Tabu search," *Int. Trans. Oper. Res.*, vol. 1, no. 3, pp. 317–336, Jul. 1994.
[11] B. M. Baker and M. A. Ayechew, "A genetic algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 30, no. 5, pp. 787–800, Apr. 2003.
[12] 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, Jul. 2009.
[13] 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.
[14] M. Gmira, M. Gendreau, A. Lodi, and J.-Y. Potvin, "Tabu search for the time-dependent vehicle routing problem with time windows on a road network," *Eur. J. Oper. Res.*, vol. 288, no. 1, pp. 129–140, Jan. 2021.
[15] 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.
[16] 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.
[17] M. Gendreau, F. Guertin, J.-Y. Potvin, and É. Taillard, "Parallel Tabu search for real-time vehicle routing and dispatching," *Transp. Sci.*, vol. 33, no. 4, pp. 381–390, Nov. 1999.
[18] 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.
[19] L. Hong, "An improved LNS algorithm for real-time vehicle routing problem with time windows," *Comput. Oper. Res.*, vol. 39, no. 2, pp. 151–163, Feb. 2012.
[20] J. Euchi, A. Yassine, and H. Chabchoub, "The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach," *Swarm Evol. Comput.*, vol. 21, pp. 41–53, Apr. 2015.
[21] V. Pillac, C. Guéret, and A. Medaglia, "A fast re-optimization approach for dynamic vehicle routing," Ph.D. dissertation, École des Mines de Nantes, 20123. [Online]. Available: https://hal.science/hal-00739782/file/Pillac_DVRP_tech.pdf
[22] O. S. da Silva Júnior, J. E. Leal, and M. Reimann, "A multiple ant colony system with random variable neighborhood descent for the dynamic vehicle routing problem with time windows," *Soft Comput.*, vol. 25, no. 4, pp. 2935–2948, Feb. 2021.
[23] N. R. Sabar, S. L. Goh, A. Turky, and G. Kendall, "Population-based iterated local search approach for dynamic vehicle routing problems," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 2933–2943, Oct. 2022.
[24] Y. Shi, "Brain storm optimization algorithm," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2011, pp. 303–309.
[25] Y. Shi, "Brain storm optimization algorithm in objective space," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 1227–1234.
[26] C. Narmatha, S. M. Eljack, A. A. R. M. Tuka, S. Manimurugan, and M. Mustafa, "A hybrid fuzzy brain-storm optimization algorithm for the classification of brain tumor MRI images," *J. Ambient Intell. Humanized Comput.*, vol. 2020, pp. 1–9, Aug. 2020.
[27] F. Pourpanah, Y. Shi, C. P. Lim, Q. Hao, and C. J. Tan, "Feature selection based on brain storm optimization for data classification," *Appl. Soft Comput.*, vol. 80, pp. 761–775, Jul. 2019.
[28] M. Liu, Q. Song, Q. Zhao, L. Li, Z. Yang, and Y. Zhang, "A hybrid BSO-ACO for dynamic vehicle routing problem on real-world road networks," *IEEE Access*, vol. 10, pp. 118302–118312, 2022.
[29] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *Eur. J. Oper. Res.*, vol. 225, no. 1, pp. 1–11, Feb. 2013.
[30] É. D. Taillard, L. M. Gambardella, M. Gendreau, and J.-Y. Potvin, "Adaptive memory programming: A unified view of metaheuristics," *Eur. J. Oper. Res.*, vol. 135, no. 1, pp. 1–16, Nov. 2001.
[31] P. Kilby, P. Prosser, and P. Shaw, "Dynamic VRPs: A study of scenarios," Univ. Strathclyde, Glasgow, U.K., Tech. Rep., 1, 1998.
[32] A. Lackner, "Selection meta-heuristics for dynamic vehicle routing problem," in *Göttinger Wirtschaftsinformatik, Heraus-geber* (Dynamische Tourenplanung mit ausgewählten Metaheuristiken), vol. 47, J. Biethahn and M. Schumann, Eds., 2004.
[33] 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.
[34] S. Chen, R. Chen, G.-G. Wang, J. Gao, and A. K. Sangaiah, "An adaptive large neighborhood search heuristic for dynamic vehicle routing problems," *Comput. Electr. Eng.*, vol. 67, pp. 596–607, Apr. 2018.

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 currently 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 ZHAO received the Ph.D. degree in computer science from The University of New South Wales, Canberra, Australia, in 2018, and the Ph.D. degree in management science and engineering from the Beijing University of Technology, Beijing, China, in 2019.

He is currently 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.

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 currently working on research with the China Telecom Research Institute, Guangzhou, China. His research interests include artificial intelligence 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 currently a Researcher with the China Telecom Research Institute, Guangzhou, China. He is also 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.