Memory-Based Ant Colony System Approach for Multi-Source Data Associated Dynamic Electric Vehicle Dispatch Optimization

Lin Shi, *Student Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, Di Liang, *Student Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*— The developments of electric vehicle (EV) technology and mobile internet technology have made the EV-oriented ride-hailing service a trend in smart cities. In the service scenario, a high-quality order allocation approach is in great need to quickly process a series of customer request orders, so as to reduce total customer waiting time and transportation cost. To simulate real-world customer-EV allocation scenarios, in this paper, a dynamic EV dispatch (DEVD) model is established by considering multi-source data association from five sources, including customer, vehicle, charging, station, and service. To solve the proposed multi-source data associated DEVD model, a memory-based ant colony optimization (MACO) approach is developed. MACO maintains a memory archive to store the historically good solutions, which not only can be used to update pheromone to guide the search, but also can be used to help the reactions to environmental changes. In response to dynamic changes, a partial reassignment strategy is also proposed to re-optimize some of the assigned customer-EV pairs in the historically best solution. Moreover, an exchange or replace local search procedure is designed to enhance the performance. The MACO algorithm is applied to a set of dynamic test cases with different customer request and EV sizes. Experimental results show that MACO generally outperforms the first-come-first-served approach and some state-of-the-art ACO-based dynamic optimization algorithms.**

***Index Terms*— Dynamic electric vehicle dispatch (DEVD), memory-based ant colony optimization (MACO), intelligent transportation, multi-source data association.**

Manuscript received 24 November 2020; revised 15 June 2021 and 9 November 2021; accepted 26 January 2022. Date of publication 8 March 2022; date of current version 11 October 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102102, in part by the National Natural Science Foundation of China under Grant 62176094 and Grant 61873097, in part by the Key-Area Research and Development of Guangdong Province under Grant 2020B010166002, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the National Research Foundation of Korea under Grant NRF-2021H1D3A2A01082705. The Associate Editor for this article was R. Langari. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

Lin Shi and Zhi-Hui Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, also with the Pazhou Laboratory, Guangzhou 510330, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, Guangzhou 510006, China (e-mail: zhanapollo@163.com).

Di Liang is with the School of Engineering, Westlake University, Hangzhou 310000, China, and also with the Westlake Institute for Advanced Study, Hangzhou 310000, China.

Jun Zhang is with Hanyang University, Ansan 15588, South Korea, and also with the Chaoyang University of Technology, Taichung 41349, Taiwan (e-mail: junzhang@ieee.org).

Digital Object Identifier 10.1109/TITS.2022.3150471

I. Introduction

WITH the development of mobile internet technology, online car hailing services (e.g., Didi and Uber) have become popular in people's travel in smart cities [1]. Moreover, electric vehicles (EVs) are gradually being promoted as an alternative to fuel vehicles in smart cities due to the increasing green energy requirements in society [2] and the low energy consumption and environmental protection of EVs [3]. For example, Didi Chuxing, China's largest online ride-hailing platform, launched the first customized car, an EV called D1, in 2020 through cooperation with the BYD company [4]. Also, a recent study based on Uber shows that the use of EVs in online ride-hailing services is greatly beneficial for reducing emissions and has no statistical difference for services when compared with using fuel vehicles [5]. Therefore, nowadays EVs have gradually become an important part of the online ride-hailing services. The significance of online ride-hailing services and the universality of EVs raise the urgent need for research into EVs operations.

Compared to the research on fuel vehicles [6], [7], the research into EVs operations mainly includes energy management [8], [9], charging station allocation [10], power and charging system [11], [12], and EV route planning [13]–[15]. Traditional fuel vehicles use diesel or petrol as energy, while EVs rely on electricity [16]. For traditional fuel vehicles, due to the large capacity fuel tank and the quick refueling speed, the influence of tank capacity and refueling is always negligible in research. However, for EVs, they have relatively small battery capacity and slow recharging speed, which cannot be ignored in practical applications. These differences between fuel vehicles and EVs make it more complicated to dispatch EVs in vehicle dispatch problems of online ride-hailing service, due to the need for considering multi-source data, such as the EV battery status and the charging station information.

Many studies have been made in solving vehicle dispatch problem. Some of them are on traditional fuel vehicle dispatch, and some of them are on EV dispatch. For solving vehicle dispatch problem, a simple way is dispatching the nearest vehicle to the customer who makes a request, based on the first-come-first-served (FCFS) approach [17], [18]. However, this approach only focuses on individual customer satisfaction and cannot provide a satisfactory solution at the global level.

As described in [19], during peak demand periods, Didi Chuxing [20] needs to match over a hundred thousand orders every second in China. Therefore, for the ride-hailing service platform, a global dispatch scheme is in great need. To this aim, some studies are conducted by considering the dispatch of vehicles to customers in a global view. Seow *et al.* [21] proposed a multiagent system called *N*TuCab to assign taxis to all customer requests made in a given time window. The agents on behalf of drivers cooperatively negotiate the assignment of customer requests in a distributed fashion. Zhang *et al.* [19] modeled the taxi order dispatch as a combinatorial optimization problem (COP). They predicted the probability of a customer request accepted by a driver based on various factors and used a hill-climbing method to maximize the global success rate. Miao *et al.* [22] presented a dynamic taxi dispatch problem based on real-time sensing data. A receding horizon control approach is applied to allocate vacant taxis to different regions for matching the passenger demands. Hu and Dong [23] proposed an optimization-based dispatch model that considered both the taxi system efficiency and customer equity. Moreover, an artificial-neural-network-based model is proposed and trained using the optimization model's dispatch solutions to learn the optimal dispatch strategies.

Different from the traditional fuel vehicle dispatch problem that only needs to consider the customer data (e.g., location and destination) and the vehicle data (e.g., location, velocity, and service status), the EV dispatch (EVD) problem needs to consider more data from other sources such as the charging data (e.g., remaining battery capacity and charging information) and the station data (e.g., location and available status). For example, Shi *et al.* [24] regarded the scheduling of an EV with insufficient battery to complete the service as an infeasible action and developed a reinforcement learning based algorithm to solve a community owned EVD problem for providing ride-hailing services to local residents. In [25], we studied to solve the EVD problem in static environment. However, in the real-world real-time service environment of EVD, dynamic changes may always occur such as the arrival of new customer requests, the cancellation of old customer requests, or the entry and exit of EVs. In order to make our EVD model more practical for real-world application, a dynamic EVD (DEVD) model is proposed in this paper, so that the real-time dynamic information about new and cancelled customer requests can be considered. Therefore, the DEVD model is a practical dispatch model associated with multi-source (i.e., five sources) data from the customer, vehicle, charging, station, and service.

In order to efficiently solve the multi-source data associated DEVD problem, a swarm intelligence algorithm named ant colony optimization (ACO) [26]–[28] is adopted in this paper because swarm intelligence algorithms have shown promising performance in many kinds of optimization and scheduling problems [29]–[32]. The ACO is a meta-heuristic search algorithm inspired by the foraging behavior of ants in nature. Real ants cooperate to find the shortest path from their nest to the food via pheromone. In recent years, ACO and its ant colony system (ACS) variant [33] have been successfully applied to many COPs, such as cloud resource scheduling [34], [35], scheduling problems [36], [37], personalized trip recommendation [38], disassembly planning problems [39], vehicle routing problem [40]–[42], and taxi dispatch problems [43]. Since the DEVD problem studied in this paper is a COP extended from taxi order dispatch, ACO can be a promising solver. In fact, in our previous study [25], an ACS-based approach has shown its effectiveness and efficiency in the static EVD problem, showing the potential of ACO in dealing with the DEVD problem.

The DEVD is also a dynamic optimization problem (DOP) and is challenging for traditional ACO/ACS algorithms. In the literature, the research into ACO-based approaches for solving DOP and dynamic COP has also raised great attention [44]–[47]. The simplest way is to restart the algorithm once a dynamic change occurs [48]. However, the re-optimization process is time-consuming for the re-convergence, resulting in poor efficiency for most DOPs that are with smooth/slight changes [49]. Therefore, some studies have been made to better solve dynamic COP. Guntsch and Middendorf [50] proposed a population-based ACO (P-ACO) to solve dynamic traveling salesman problem (TSP) and also extended the P-ACO to solve dynamic quadratic assignment problem [51]. Inspired by [51], Montemanni *et al.* [52] further designed a pheromone conservation parameter to manage the information transfer from the previous environment to the new environment, so as to solve dynamic vehicle routing problem (DVRP), resulting in the ACS-DVRP algorithm. Mavrovouniotis and Yang [53] studied the dynamic TSP with traffic factors and proposed an ACO framework with three immigrant schemes to increase the diversity, including random immigrant, elitism-based immigrant, and memory-based immigrant. Their experimental results show that the elitism-based immigrant ACO (EIACO) performs best in random dynamic environments and the memory-based immigrant ACO (MIACO) performs best in cyclic dynamic environments.

In this paper, we propose a novel and more realistic memory-based ACO (MACO) approach for efficiently solving the DEVD problem. The MACO uses a memory archive to record the solutions that perform well in previous environments. These well-performing solutions can be utilized to help fast indicate the new global optimal solution in the new environment. Moreover, two special strategies are designed to further help MACO better solve the DEVD problem. Firstly, in response to dynamic changes, we design a partial reassignment (PR) strategy to re-optimize some of the customer requests instead of re-optimizing all valid customer requests in the new environment. Secondly, a local search procedure called exchange or replace (EoR) strategy is designed to enhance the performance. We conduct experiments on a set of dynamic test cases with different customer and EV sizes and compare MACO algorithm with not only the FCFS approach, but also some state-of-the-art and recent ACO-based dynamic optimization algorithms. Experimental results show that MACO has generally better performance than the compared algorithms on the DEVD problem.

Therefore, the main contributions of this paper are summarized as follows:

(1) Firstly, we establish a dynamic EV dispatch model for simulating real-world dynamic EV dispatch application scenarios, by associating multi-source data from five sources, including customer, vehicle, charging, station, and service.

(2) Secondly, we propose a memory-based ACO approach for efficiently solving the DEVD problem by enhancing the adaptability in dynamic environments via pheromone transfer through archived solutions.

(3) Thirdly, we propose a partial reassignment strategy to optimize partial requests in the new environment to better respond to the dynamic environment, and a new local search procedure to enhance the performance. The proposed strategies help the MACO algorithm obtain a better balance between the performance and execution time, being more suitable for the DEVD real-world application.

The rest of this paper is organized as follows. Section II introduces the multi-source data associated DEVD model. Section III describes the MACO algorithm in detail. Section IV presents the experimental results, comparisons, and analysis. Finally, Section V concludes this paper.

II. Electric Vehicle Dispatch Problem

A. Static EV Dispatch Problem

In a dispatch scenario, a number of EVs and charging stations are distributed within a certain geographical region. The dispatch center monitors the activity of EVs and the status of charging stations (whether available or not) in real time via GPS and wireless communication network. In a small time window, multiple customers send out service requests. The dispatch center arranges suitable EVs for these customers simultaneously, so as to maximize total service quality. The basic notations for the EVD problem that is modeled by multi-source data are listed in Table I.

The service process of EVD is illustrated in Fig.1, which is divided into two stages: 1) EV goes to the customer location; and 2) EV delivers the customer to the destination. When the dispatch center selects a candidate EV for a customer, it needs to consider both the locations of the customer and the EV. Furthermore, the remaining power of the EV battery should also be taken into account. That is, the EV's remaining battery capacity should be 'sufficient' to deliver the customer to the destination. To ensure the availability of EV after reaching the destination, 'sufficient' is defined as that the remaining battery capacity can support the EV to reach at least one charging station after completing a customer's service request. If the remaining battery capacity of an EV is not sufficient, this EV needs to be recharged during the service process.

In an actual EV service scenario, there are three types of service routes for EVs as shown in Fig.1. The selection of route type depends on the remaining battery capacity of the EV: 1) if the remaining battery capacity of an EV is sufficient, then the route for this EV is $R_1 + R_2$; 2) if the remaining battery capacity is not sufficient to support this EV to reach the customer location or reach any charging station after reaching the location of the customer. That is, in this service, the EV needs to be recharged in the first stage of the service process, so the route is $x_1 + x_2 + R_2$; 3) if the remaining battery capacity is not sufficient but can support this EV to reach at least one charging station after reaching the customer location. In this case, this EV can be recharged in the first stage or the second stage, so it is necessary to judge in which stage the transportation cost is lower, so as to select route $x_1+x_2+R_2$ or route $R_1+x_3+x_4$ for this customer-EV pair. It is assumed that if an EV is needed to be recharged, it is only recharged once during the service. Moreover, to avoid too long charging time, the charging process stops once the remaining battery capacity is sufficient to support the EV to complete the service, without having to fully recharge the battery. It should be noted that when a low-power EV completes a service during which it needs to recharge, it will stop accepting request, which is controlled by the dispatch center. The EV will be recharged at the charging station until it is fully recharged or the charging process is terminated by the owner, and then the EV could continue to serve. This can avoid the situation where the EV is always in low-power status and has to be recharged again and again. Also note that how to manage EV charging may vary due to different policies on different platforms. The policy considered herein is just an example.

Fig.2 illustrates how to select the charging station when the remaining battery capacity of the EV is not sufficient, taking recharging in the first stage as an example. First, the range that the remaining battery capacity of the EV can support its arrival is calculated and signed by the dashed circle in Fig.2.

TABLE I
Basic Notations for the Associated Data in EVD Problem

Data Source	Symbol	Definition and Setting
Customer Data	N	Number of customers (requests).
	c_i	ith customer (request).
	g_i	Destination of c_i.
Vehicle Data	M	Number of EVs.
	v_j	jth EV.
	vel	EV velocity (50 km/h).
Charging Data	Q	Full battery capacity (60 kWh).
	q_j	Remaining battery capacity of v_j.
	P	Charging power (100 kW).
	r	Electricity consumption per km (0.3 kWh/km).
Station Data	W	Number of charging stations.
	s_k	kth charging station.
Service Data	T	Charging time threshold (5 minutes).
	f	Penalty factor (10).

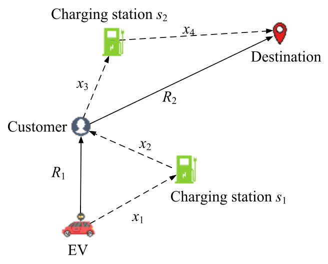


Fig. 1. Illustration of EV serving process.

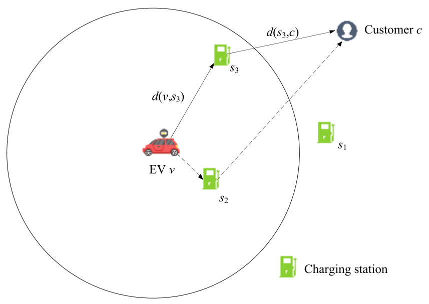


Fig. 2. Illustration of charging station selection.

Charging stations s_2 and s_3 located in this range are candidate charging stations. Then based on the minimal increase in transportation cost, s_3 in the candidate set is selected as the best charging station. In this case, although s_2 is closer to v_j, s_3 is selected because the total driving distances from v to c via $s_3(i.e., d(v, s_3) + d(s_3, c))$ is smaller than the total driving distances from v to c via $s_2(i.e., d(v, s_2) + d(s_2, c))$. If the charging station selection occurs in the second stage, the selection mechanism is the same as that in the first stage. That is, among all the charging stations reachable by the EV, the one with the minimal increase in transportation cost is selected.

Therefore, if an EV v_j is assigned to a customer c_i to drive to the destination g_i, its driving distance $l(c_i, v_j)$ is calculated as:

$$l(c_i,v_j)=\begin{cases} d(v_j,c_i)+d(c_i,g_i), & \text{if } \frac{q_j}{r}\ge d(v_j,c_i) \\ \quad +d(c_i,g_i)+d(g_i,ns_{g_i}) & \\ \min(d(v_j,s_1)+d(s_1,c_i)+d(c_i,g_i),\ d(v_j,c_i) & \\ \quad +d(c_i,s_2)+d(s_2,g_i)), & \\ \quad \text{else if}\frac{q_j}{r}\ge d(v_j,c_i)+d(c_i,ns_{c_i}) & \\ d(v_j,s_1)+d(s_1,c_i)+d(c_i,g_i), & \\ \quad \text{else if}\frac{q_j}{r}\ge d(v_j,ns_{v_j}) & \\ \text{NA, otherwise} & \end{cases} \tag{1}$$

where ns_{g_i}, ns_{c_i}, and ns_{v_j} represent the nearest charging stations to g_i, c_i, and v_j, respectively; q_j is the remaining battery capacity of v_j; r is the electricity consumption per kilometer; s_1 and s_2 represent the charging stations selected in the first stage and the second stage, respectively. It should be noted that s_1 and s_2 are not necessarily ns_{v_j} and ns_{c_i}. "NA" means that the remaining battery capacity of an EV is not enough to reach any charging stations, so it is not available and not considered in the dispatch process. In the description below, we only discuss EVs that can reach at least one charging station according to their remaining battery capacity.

To improve the service quality, the time cost of a service process should be considered, including driving time and charging time. The driving time $dt(c_i, v_j)$ is calculated as

$$dt(c_i,v_j)=\frac{l(c_i,v_j)}{vel} \tag{2}$$

where vel is the EV velocity. The charging time $ct(c_i, v_j)$ is calculated as

$$ct(c_i,v_j) = \begin{cases} 0, & \text{if } \frac{q_j}{r}\ge d(v_j,c_i) \\ & +d(c_i,g_i) \\ & +d(g_i,ns_{g_i}) \\ \frac{\left(l(c_i,v_j)+d(g_i,ns_{g_i})\right)\cdot r-q_j}{P}, & \text{otherwise} \end{cases} \tag{3}$$

where P is the EV charging power. Therefore, the time cost of one service process between c_i and v_j is

$$tc(c_i,v_j)=dt(c_i,v_j)+ct(c_i,v_j) \tag{4}$$

For a customer c_i, the distance $d(c_i, g_i)$ between the starting position and the destination is fixed, regardless of which EV is assigned. It has no influence on the evaluation of dispatch results but may have a negative impact on the optimization process because of the relatively large value. Therefore, $d(c_i, g_i)$ is not considered in the proposed model. In addition, to avoid excessive charging time and improve customer's using experience, a penalty for the charging time is set. That is, if the charging time exceeds a threshold T, extra charging time will be penalized. Therefore, the time cost in (4) is updated as

$$tc(c_i,v_j)=\begin{cases} dt(c_i,v_j)-\frac{d(c_i,g_i)}{vel}+ct(c_i,v_j), & \\ \quad \text{if } ct(c_i,v_j)\le T & \\ dt(c_i,v_j)-\frac{d(c_i,g_i)}{vel}+T+\left(ct(c_i,v_j)-T\right)\cdot f, & \\ \quad \text{otherwise} & \end{cases} \tag{5}$$

where T is the charging time threshold and f is the penalty factor. Equation (5) indicates that for a customer c_i, an EV, that is closer to the customer and has more remaining battery capacity, has a lower time cost. Thus, it is more likely to be assigned to this customer. It should be noted that our model is generic that it can consider both the situations of allowing and not-allowing recharging during the service. That is, we can simply set $T = 0$ and $f =$ infinity so that the low-power EV cannot be considered to be assigned to the customer.

The objective of the EV dispatch problem is to minimize the time cost formulated as

$$\text{minimize} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij}\cdot tc(c_i,v_j) \tag{6}$$

$$\text{subject to } x_{ij}=\begin{cases}1, & \text{if } v_j \text{ is assigned to } c_i\\ 0, & \text{otherwise}\end{cases} \tag{7}$$

$$\forall i\in[1,N],\quad \sum_{j=1}^{M} x_{ij}=1 \tag{8}$$

$$\forall j\in[1,M],\quad \sum_{i=1}^{N} x_{ij}\le 1 \tag{9}$$

where x_{ij} is an indicator variable. Constraint (8) guarantees that every customer will be assigned an EV and constraint (9)

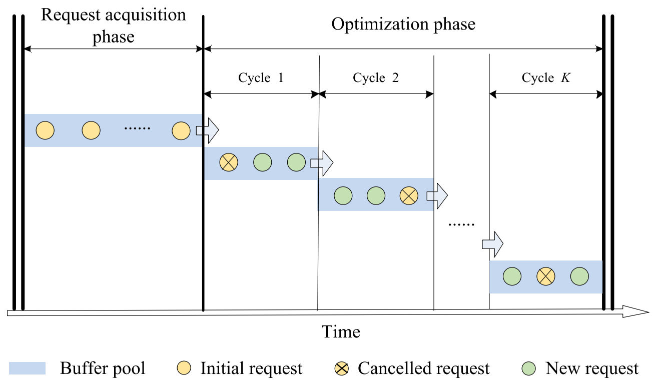


Fig. 3. Illustration of the DEVD dispatch process.

ensures that an EV is assigned to at most a customer. In this paper, it is supposed that the number of EVs to be assigned is not smaller than the number of customers, so that all the requests can be satisfied.

B. Dynamic EV Dispatch Problem

During the working time of the dispatch center, the dispatch algorithm is repeated carried out again and again. Each execution of the dispatch algorithm is regarded as a segment, which is a complete dispatch process that includes two phases: the request acquisition phase and the optimization phase. The customers in each segment can obtain the dispatch results only after the current segment is completed.

In our previous work that uses ACS to solve the EVD problem [25], the algorithm is carried out at the optimization phase after the customer requests are obtained in the request acquisition phase. However, in a real-time service environment, dynamic changes can occur during the optimization phase. Therefore, in this paper, we build the DEVD model to consider the dynamic information that occurs during the optimization phase, including the new customer requests coming and the old requests cancellation. Specifically, we divide the optimization phase into several cycles and execute them one after one, so that the new coming requests and the cancelled requests during one optimization cycle can be considered in the next optimization cycle.

Fig.3 illustrates the dispatch process of the DEVD model, in which optimization cycles of the optimization phase are carried out after the request acquisition phase. In the request acquisition phase, the dispatch center receives customers' requests. In the optimization phase, these requests (customers) are dispatched by an optimization algorithm to assign appropriate EVs. However, as new requests may come and old requests may be cancelled during the optimization phase, the DEVD model treats the dispatch as a DOP and divides the optimization phase into several cycles. In the first cycle, the optimization algorithm only considers the requests acquired in the request acquisition phase. Then, new requests and cancelled requests during the first cycle are put into the buffer pool and will be considered in the second cycle. Note that the first cycle is not aware of these new and cancelled requests. Moreover, the second cycle carries out the optimization algorithm by considering all these requests, that is, the requests acquired in the request acquisition phase and the requests acquired and cancelled in the first cycle. Similarly, the third cycle considers all the requests dispatched in the second cycle and those new/cancelled requests appeared during the second cycle. This way, after the optimization of the last cycle, the algorithm obtains the dispatch solution, which considered the dynamic changes in the optimization phase, and will send the results to both the customers and EVs. Because the requests to be optimized in each cycle are different from each other, each cycle can be regarded as being in a different environment. These different environments also reflect the dynamicity of DEVD problem.

III. MACO FOR SOLVING THE DEVD PROBLEM

As mentioned above, to efficiently solve the DEVD problem, the proposed MACO approach is executed on the optimization phase in a DEVD dispatch process, and the optimization phase is divided into several optimization cycles so that each cycle can consider the dynamic service information that occurred in the previous cycle. Therefore, without loss of generality, the optimization process described in the follows is based on the t^{th} optimization cycle in a DEVD dispatch process, named the environment E_t in this paper.

A. Encoding of MACO for DEVD

Each ant in MACO is encoded as a matrix, as shown in

$$X = \begin{bmatrix} x_{1,1} & \cdots & \cdots & \cdots & x_{1,M} \\ \vdots & \ddots & & & \vdots \\ \vdots & & x_{i,j} & & \vdots \\ \vdots & & & \ddots & \vdots \\ x_{N_t,1} & \cdots & \cdots & \cdots & x_{N_t,M} \end{bmatrix} \tag{10}$$

where the number of rows and columns of the matrix are the number of valid requests in current environment E_t, denoted by N_t, and the number of available EVs, denoted by M, respectively. Each element $x_{i,j}$ in the matrix refers to whether the customer i is assigned the EV$_j$. If the value is 1, it means that customer i has been assigned EV$_j$, while value 0 means not assigned, as shown in (7).

B. Initialization State Configurations

In ACO-based algorithms, pheromone records accumulated experience in the colony, which affects the path construction of ants. For the DEVD problem, the pheromone value $\tau(i, j)$ is set between customers and EVs to indicate the preference that an EV$_{v_j}$ is assigned to a customer c_i, and its initial value τ_0 is set as

$$\tau_0 = (N_t \cdot T_{nn})^{-1} \tag{11}$$

where N_t is the number of customers to be assigned in the current environment E_t and T_{nn} is the time cost (i.e., the fitness value calculated by (6)) of the solution obtained by the FCFS approach. The FCFS approach works simply as that, when a customer request is received, an EV with the lowest time cost among the currently available EVs is assigned to it.

C. Solution Construction

The solution construction process in MACO is the same as that in the traditional ACS algorithm, where ants iteratively construct solutions by using a state transition rule. To increase the diversity of solutions, the order of customers to be assigned is shuffled randomly before construction. Each ant searches for feasible solutions by assigning EVs to customers one by one, according to the order in which customers are shuffled. The search behavior is influenced by pheromone (swarm knowledge) and heuristic information (individual knowledge). Similar to TSP [33], heuristic information in the DEVD problem is given by

$$\eta(i,j)=\frac{1}{tc(c_i,v_j)} \tag{12}$$

where $\eta(i,j)$ represents the heuristic information between customer i and EV$_j$. Based on the pheromone and heuristic information, the probability that an unassigned EV$_j$ is selected for customer i is calculated as

$$p(i,j)=\begin{cases}\dfrac{[\tau(i,j)]\cdot[\eta(i,j)]^{\beta}}{\sum\limits_{u\in J_i}[\tau(i,u)]\cdot[\eta(i,u)]^{\beta}}, & \text{if} j\in J_i\\ 0, & \text{otherwise}\end{cases} \tag{13}$$

where J_i is the set of EVs that have not been assigned and β ($\beta>0$) is a predetermined parameter that determines the relative importance of heuristic information.

The state transition rule is as follows: for customer i, an EV j is dispatched by applying the rule given by

$$j=\begin{cases}\arg\max\limits_{u\in J_i}\{[\tau(i,u)]\cdot[\eta(i,u)]^{\beta}\}, & \text{if} q\le q_0\\ J, & \text{otherwise}\end{cases} \tag{14}$$

where q is a random variable uniformly distributed in [0,1], J is a random number selected by roulette wheel selection according to the probability calculated in (13), and $q_0(0\le q_0\le 1)$ is a parameter to control the exploitation and exploration behaviors of ants. For the customer i, if $q\le q_0$, then the ant greedily chooses the EV with the maximal pheromone and heuristic information, measured by $\tau(i,u)\cdot\eta(i,u)^{\beta}$. Otherwise, the EV is determined as J.

D. Memory-Based Pheromone Updating Rule

In MACO, the best solution of the current iteration and the historically best solution (i.e., the global optimal solution from the beginning of the current environment) are denoted as S^b and S^{gb}, respectively. In every iteration, the S^b is added into the memory archive. Furthermore, if the S^b is better than the S^{gb}, the S^{gb} will be updated by the S^b in every iteration. With the help of the memory archive, the pheromone can be updated when a solution enters or leaves the memory.

The memory archive is with the size of K and uses a first-in-first-out fashion to keep the latest best solutions information. For the first K iterations, solutions can be stored in the memory archive one by one during the iterations. Therefore, when the iteration index $g\le K$, the pheromone is positive updated according to the stored solution at the end of the g^{th} iteration as

$$\tau(i,j)=\tau(i,j)+\Delta\tau(i,j),\quad \forall(i,j)\in S^b \tag{15}$$

where $\Delta\tau(i,j)=(\tau_{\max}-\tau_0)/K$ and $\tau_{\max}$ denotes the maximum pheromone value, which is a predetermined parameter. Note that the τ_0 obtained in (11) will be re-calculated in every environmental change, as described later in Section III-E-2).

From the $(K+1)^{th}$ iteration, the oldest (i.e., the first-in) solution in the memory archive is removed (i.e., first-out) before the new solution is added and the pheromone information is negative updated according to the removed solution as

$$\tau(i,j)=\tau(i,j)-\Delta\tau(i,j),\quad \forall(i,j)\in S^{oldest} \tag{16}$$

where S^{oldest} is the oldest solution in the memory archive (i.e., the removed solution) and $\Delta\tau$ is the same as defined in (15). After (16), the new solution enters the memory archive and the pheromone is positive updated according to this newly entered solution via (15). The pheromone values are maintained between τ_0 and τ_{max} during the optimization process.

E. Strategies Reacting to Dynamic Change Based on Memory

During the optimization process, some old customers may cancel requests and some new customers may send requests. The solutions obtained in the previous environment (i.e., the previous optimization cycle) may no longer be feasible in the current environment, and the requests to be optimized may be different from those in the previous environment. In this paper, partial reassignment (PR) strategy and pheromone transfer operation are performed after the environment changes (i.e., when entering the next optimization cycle). The PR strategy is used to update the requests to be optimized in the new environment. The pheromone transfer operation is used to repair the solutions in the memory archive and update the pheromone.

1) Partial Reassignment: When the environment changes, new coming customer requests and cancelled requests are considered. The first thing to do is to invalidate the cancelled requests. Then, for all currently valid customer requests, a simple method is re-assigning EVs for all requests, no matter whether they have been assigned before or not [49]. However, such a method discards matching information in previous environments and is not suitable for slightly changing environments. Another method is to optimize the new coming customer requests independently once a change occurs, based on the incremental (INC) optimization method [54], which can reduce response time but may easily fall into local optima. In the customer-EV dispatch service, the environment often changes slightly due to short optimization time. Therefore, the restart strategy may be computationally expensive by considering all the valid customer requests, while the INC strategy may trap into local optimum by only considering the new/cancelled customer requests. Thus, this paper proposed a new PR strategy, different from the restart strategy and the INC strategy, to respond to the environmental change. PR strategy considers both the new/cancelled customer requests and

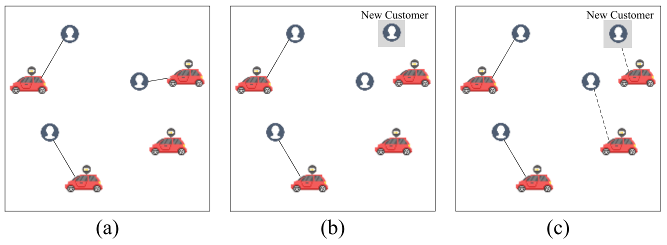


Fig. 4. Illustration of the partial reassignment strategy. (a) The globally best solution in the previous environment. (b) Release matched customer-EV pairs near new customer c_{new} with $N_{init} = 3$, $N_{new} = 1$, and $R = 0.3$. (c) One dispatch results in the new environment. The dotted line refers to the customer-EV pairs that participate in the optimization in the new environment.

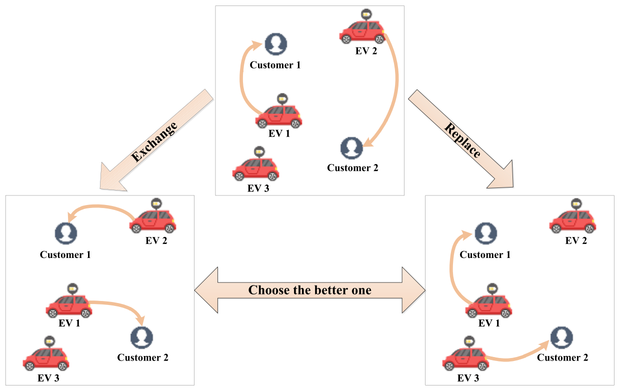


Fig. 5. Illustration of the EoR local search.

previously assigned requests. However, not all the previously assigned requests, but only some of them are considered. Therefore, the number of customer requests to be assigned in the new environment is between that of INC and restart, so as to make a trade-off between reducing computational time and improving solution quality.

To get the requests to be optimized in the new environment, the PR strategy is implemented. The PR strategy releases some matched customer-EV pairs to get customers and EVs that need to be re-dispatched, so as to avoid local optimal to some extent. The specific operation of PR strategy is as follows. It should be noted that all operations of PR are performed on the historically best solution S^{gb}. First, for the cancelled requests, their assigned EVs are released. Second, for each new customer (i.e., request), find its nearest $R \cdot \left\lceil \frac{N_{init}}{N_{new}} \right\rceil$ customers (measured by Euclidean distance) and release those customer-EV pairs. The R is a predefined parameter, N_{init} represents the number of customer requests received in the request acquisition phase, and N_{new} is the number of new requests in the new environment. The more initial requests than new requests, the more customer-EV pairs will be released for optimization, so as to effectively avoid local optima. Fig. 4 shows an example of the two steps of the PR process. After the PR process, we can obtain the requests that need to be optimized in the new environment, that is, all new requests and released requests. The task of DEVD in the new environment (i.e., the next optimization cycle in Fig.3) is to assign currently idle EVs to these new and released customer requests. Note that the solution in the new environment should consider both the new dispatched and previous remained customer-EV pairs to calculate its fitness value.

2) Pheromone Transfer: In this section, we discuss how to transfer pheromone from the previous environment to the new environment. In MACO, the pheromone updating operation is performed based on the solutions in the memory archive as described in Section III-D. However, when an environmental change occurs, these solutions may be infeasible, because some customers may cancel requests. Hence, solutions in the memory archive need to be repaired and re-evaluated. As suggested in [51], a principle called *KeepElite* [55] is adopted to repair solutions after a change. That is, for each solution in the memory, the cancelled customer-EV pairs are released and the new customer-EV pairs are added, where each new customer will be assigned an EV that minimizes the increase of total time cost. Then re-evaluate each solution in the memory.

After repairing and re-evaluating all the solutions in the memory, conduct the pheromone transfer as follows. Firstly, re-initialize pheromone between all customers and EVs by (11). Then, update pheromone positively based on all these repaired solutions in the memory archive according to (15).

F. EoR Local Search Procedure

To improve solution quality, the EoR local search is carried on the historically best solution S^{gb} before updating the memory archive and pheromone in every iteration. In order to reduce the extra computational time caused by local search and to improve the solution quality as much as possible, the EoR is conducted on the customer-EV pair with the maximal time cost in S^{gb}, denoted by π_{max}. The EoR includes two components: “exchange” and “replace”. The example of the EoR local search procedure is shown in Fig.5.

1) Exchange Operation: The exchange operation swaps EVs between π_{max} and other customer-EV pairs in S^{gb}. For each customer-EV pair in S^{gb}, if the total time cost is reduced after swapping the EV with π_{max}, put this pair into an exchange list *exl*.

$$exl = \{(i, j) | tc(c_{\max}, \upsilon_{\max}) + tc(c_i, \upsilon_j) > tc(c_{\max}, \upsilon_j) + tc(c_i, \upsilon_{\max})\} \tag{17}$$

where c_{max} and υ_{max} represent the customer and EV of π_{max}. The pair with the largest value of Δ in *exl* is the selected pair to be exchanged, where Δ is defined by

$$\Delta = tc(c_{\max}, \upsilon_{\max}) + tc(c_i, \upsilon_j) - tc(c_{\max}, \upsilon_j) - tc(c_i, \upsilon_{\max}) \tag{18}$$

2) Replace Operation: The replace operation replaces υ_{max} with idle EVs, so that the total time cost is reduced. The idle EV that reduces the total time cost the most is the selected EV to be replaced.

The EoR local search chooses the better operation between “exchange” and “replace” to perform. That is, the “exchange” or the “replace” operation that reduces more total time cost is performed. It should be noted that if there is no idle EV (i.e., the number of available EVs is not larger than the number of currently valid customer requests), only the exchange operation is performed.

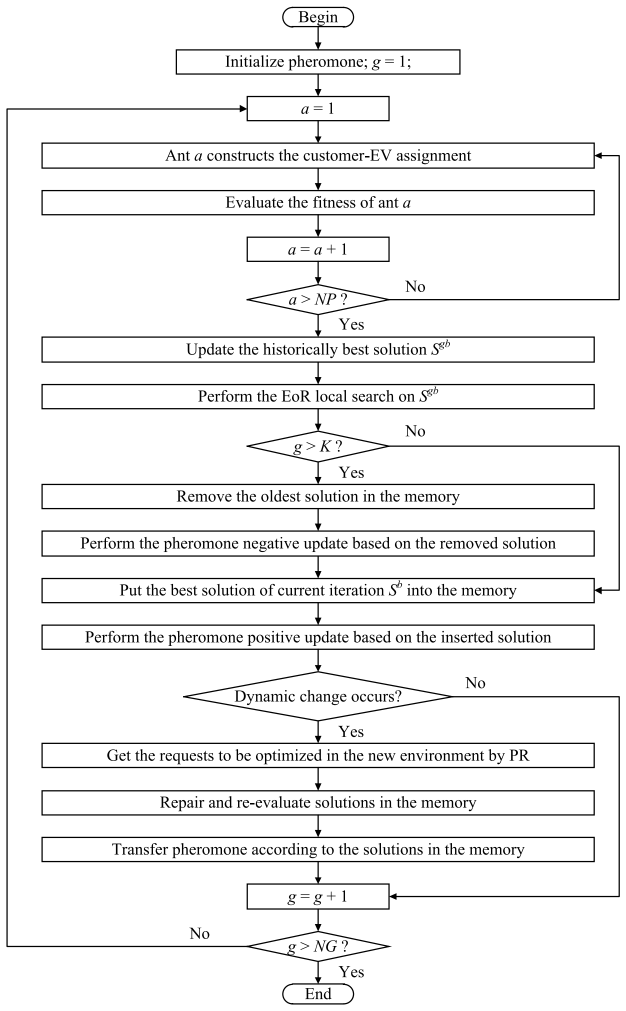

Begin
Initialize pheromone; g = 1;
a = 1
Ant a constructs the customer-EV assignment
Evaluate the fitness of ant a
a = a + 1
a > NP ?
No
Yes
Update the historically best solution S^gb
Perform the EoR local search on S^gb
g > K ?
No
Yes
Remove the oldest solution in the memory
Perform the pheromone negative update based on the removed solution
Put the best solution of current iteration S^b into the memory
Perform the pheromone positive update based on the inserted solution
Dynamic change occurs?
No
Yes
Get the requests to be optimized in the new environment by PR
Repair and re-evaluate solutions in the memory
Transfer pheromone according to the solutions in the memory
g = g + 1
g > NG ?
No
Yes
End

Fig. 6. Flowchart of the MACO algorithm.

Algorithm 1 MACO

```
Input: customer requests to be matched, available EVs
Output: historically best matching solution $S^{gb}$
Begin
1:  Initialize pheromone according to (11); $g = 1$;
2:    While $g <= NG$ Do
3:    For each ant $a$ Do
4:      Ant $a$ constructs the customer-EV assignment by (12)-(14);
5:      Evaluate the fitness of ant $a$;
6:    End For
7:    Update the historically best solution $S^{gb}$;
8:    Perform the EoR local search;
9:    If $g > K$ Do
10:     Remove the oldest solution in the memory;
11:     Update pheromone based on the removed solution by (16);
12:   End If
13:   Put the best solution of current iteration $S^b$ into the memory;
14:   Update pheromone based on the inserted solution by (15);
15:   If dynamic change occurs Do
16:     Get the requests to be optimized in the new environment by PR;
17:     Repair and re-evaluate solutions in the memory;
18:     Re-initialize pheromone by (11);
19:     Update pheromone based on the solutions in memory by (15);
20:   End If
21:     $g = g + 1$;
22: End While
End
```

G. Complete MACO Algorithm and Complexity Analysis

The flowchart and the pseudocode of the whole MACO algorithm are shown in Fig. 6 and Algorithm 1, respectively. Moreover, the complexity analysis of MACO is given as followings.

Herein, we denote the total number of customer requests and the number of available EVs as N and M, respectively. The number of ants in MACO is *NP*. First, the time complexity of initializing pheromone is $O(N \times M)$, as obtained by line 1 in Algorithm 1. Then, as for constructing solution for *NP* ants and updating the historically best solution, the time complexity is $O(NP \times N_t^{'} \times M)$ in every iteration, as obtained by lines 3-7 in Algorithm 1. The N_t' is the number of customer requests to be dispatched in the environment E_t. In the first optimization cycle, N_t' is the initial number of customer requests N_{init}; and in other cycles, N_t' is the sum of new customer requests N_{new} and released customer requests N_{rels}, computed by $R \cdot \left\lceil \frac{N_{init}}{N_{new}} \right\rceil$. The EoR local search procedure includes exchange operation and replace operation, so the time complexity of EoR is $O(NG \times N) + O(NG \times M)$, as obtained by line 8 in Algorithm 1. Next, the memory archive is updated, and the time complexity is $O(NG \times N)$, as obtained by lines 9-14 in Algorithm 1. When a dynamic change occurs, the PR strategy and pheromone transfer are performed, as obtained by lines 16-19 in Algorithm 1. The PR strategy needs to find the nearest customer requests for each new customer request, so the time complexity is $O(NG \times N_{new} \times N)$. The pheromone transfer procedure includes repairing solutions in the memory archive and updating pheromone, whose time complexities are $O(NG \times K \times N_{new} \times M)$ and $O(NG \times N \times M)$, respectively, where K is the size of the memory. Therefore, the overall time complexity of MACO is $O(NG \times NP \times N_t^{'} \times M) + O(NG \times N_{new} \times N) + O(NG \times K \times N_{new} \times M) + O(NG \times N \times M)$. The comparison of the time complexity between MACO and other algorithms (listed in Section IV-A) can be seen in Table II.

IV. Experiments

In this section, experimental tests are conducted to investigate the performance of MACO on DEVD. All the algorithms are implemented in $C++$ and run on a PC with a Core quad-core CPU i7 and 8.0GB RAM.

A. Experimental Settings

We compare MACO with the FCFS approach and five dynamic optimization algorithms, including restart ACS (RSACS), incremental (INC) method-based [54] ACS (INCACS), ACS-DVRP [52], P-ACO [50], and EIACO [53].

1) FCFS: Assign EVs with the minimum time cost for all valid customer requests according to their orders of arriving at the dispatch center.
2) RSACS: Re-initialize the pheromone and re-optimize all customer requests when a dynamic change occurs. The optimizer is ACS [33] and the algorithm is termed as restart ACS (RSACS).
3) INCACS: Once a dynamic change occurs, it only focuses on the new coming and cancelled customer requests and assigns suitable EVs for newly

TABLE II

COMPARISON OF THE TIME COMPLEXITY BETWEEN MACO AND OTHER ALGORITHMS

Algorithm	Time Complexity
MACO	$O(NG\times NP\times N_t'\times M)+O(NG\times N_{new}\times N)+O(NG\times K\times N_{new}\times M)+O(NG\times N\times M)$
FCFS	$O(N\times M)$
RSACS	$O(NG\times NP\times N\times M)$
INCACS	$O(NG\times NP\times N_{new}\times M)$
ACS-DVRP	$O(NG\times NP\times N\times M)$
P-ACO	$O(NG\times NP\times N\times M)+O(NG\times K\times N\times M)$
EIACO	$O(NG\times NP\times N\times M)+O(NG\times K\times(N\times M+N^2))+O(NG\times K^2)$

TABLE III

PARAMETER SETTINGS OF SIX DYNAMIC OPTIMIZATION ALGORITHMS

Algorithms	Parameters Settings
RSACS	q_0=0.9, ρ=ε=0.1, β=5.0, NP=20, NG=150
INCACS	q_0=0.9, ρ=ε=0.1, β=5.0, NP=20, NG=150
ACS-DVRP [52]	q_0=0.9, ρ=ε=0.1, β=5.0, NP=20, NG=150, γ_r=0.3
P-ACO [50]	q_0=0.5, β=5.0 , $\tau_{\max}$=1.0, K=10, NP=20, NG=150
EIACO [53]	q_0=0.5, β=5.0, $\tau_{\max}$ =1.0, K=10, r=0.3, NP=20, NG=150
MACO	q_0=0.5, β=5.0, $\tau_{\max}$=1.0, K=10, R=0.5, NP=20, NG=150

added requests. The optimized result will be directly attached to the globally best solution of the previous environment. The optimizer is ACS and the algorithm is termed as incremental ACS (INCACS).

4) ACS-DVRP [52]: ACS-DVRP introduces a parameter to transfer pheromone from the previous environment to the new environment for solving DVRP. When ACS-DVRP is adopted to solve DEVD problem, the optimization process is similar to that of RSACS. Except that when a dynamic change occurs, the migration of pheromone is carried out and controlled by a parameter.
5) P-ACO [50]: In every iteration, P-ACO stores the iterative best solution S^b found by the ant colony into a population list of size K for solving dynamic TSP. When a change occurs in DEVD problem, P-ACO repairs the solutions in the population list by releasing the cancelled customer-EV pairs and re-assigning EVs to new coming customer requests and then re-optimizes all currently valid customer requests.
6) EIACO [53]: EIACO is an improved variant of P-ACO to solve the dynamic TSP, which introduces elitism-based immigrants to replace the worst ants in the population list in every iteration. The immigrants are generated based on the best solution in the previous iteration, using the inver-over operator [56].

Some parameter settings of EVs and DEVD have been given in Table I, where the EV technical parameters are based on the BYD e6 product parameters [57], [58]. The parameters settings of the six compared algorithms are listed in Table III.

The EV dispatch is conducted in a physical area covering 100 km $\times$ 100 km. The position of all objects is generated uniformly, including EV, customer (request), destination, and charging station. The remaining battery capacity of the EV is randomly generated within [1 kWh, 60 kWh]. The requests to be cancelled are randomly selected from all customer requests.

TABLE IV

DEVD TEST CASES

No.	W	N	M	Number of initial requests	Number of new requests	Number of cancelled requests
A1	50	50	100	25	(5,5,5,5,5)	(1,1,1,1,1)
A2	50	50	200	25	(5,5,5,5,5)	(1,1,1,1,1)
A3	50	100	100	50	(10,10,10,10,10)	(1,1,1,1,1)
A4	50	100	200	50	(10,10,10,10,10)	(1,1,1,1,1)
A5	50	100	500	50	(10,10,10,10,10)	(1,1,1,1,1)
A6	50	200	200	100	(20,20,20,20,20)	(1,2,2,2,2)
A7	50	200	500	100	(20,20,20,20,20)	(1,2,2,2,2)
A8	50	500	500	400	(20,20,20,20,20)	(4,5,5,5,5)

The specific data of the test cases can be downloaded from https://zhanapollo.github.io/zhanzhh/resources.htm.

For DEVD model, we design various test cases (i.e., A1 to A8) by considering different EV sizes and customer request sizes, ranging from 100 to 500 and from 50 to 500, respectively, shown in Table IV. The W is the number of charging stations. The N is the total number of customer requests (including valid and cancelled requests) and the M is the number of available EVs which can reach at least one charging station with the remaining battery capacity. Initial requests in the table refer to requests acquired in the request acquisition phase. After the request acquisition phase, the optimization phase starts, from the 1th iteration. For dynamic changes, it is assumed that new customer requests and cancelled requests are processed every 25 iterations. Thus, the environment changes are considered after the 25th, 50th, 75th, 100th, and 125th iterations. Changes that occur between the 125th and 150th iterations are not considered in current dispatch process and are postponed to the next dispatch process. In each environmental change, new requests are taken from the buffer pool and $\lceil \gamma \times N_t \rceil$ customer requests are randomly cancelled, where N_t is the number of valid requests in current environment E_t and γ is set to 0.01 herein. The numbers of new requests and cancelled requests for each change are listed in the "Number of new requests" column and the "Number of cancelled requests" column, respectively. For example, on the test case A8, the numbers of charging stations, available EVs, and initial customer requests in the initial iteration are 50, 500, and 400, respectively. After the 25th iteration, 4 old requests are cancelled and 20 new requests arrive. So after the first dynamic change, the number of valid customer requests is 416. The same changes also occur after the 50th, 75th, 100th, and 125th iterations, according to Table IV.

In the experiments, all the stochastic algorithms perform 30 independent runs on each case, with their mean values compared. The best results are marked in **boldface**. Moreover, Wilcoxon's rank-sum test is conducted at a 0.05 significance level. The results marked with "+", "≈", and "−" indicate that MACO is significantly better than, similar with, and significantly worse than the compared algorithm, respectively.

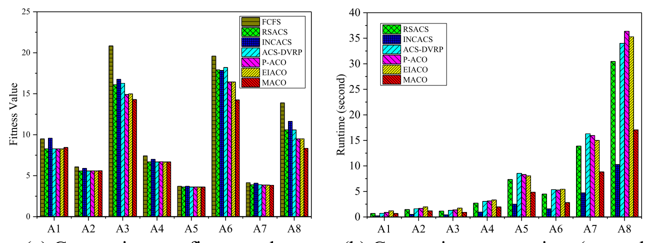
B. Experimental Results

Table V lists the fitness values (i.e., the average time cost in minutes required for completing all valid customer requests, calculated by (6)) obtained by MACO, FCFS, and other five

TABLE V

COMPARISONS OF THE FITNESS VALUES AND THE RUNTIME (SECOND) BETWEEN MACO AND OTHER ALGORITHMS

Instances	MACO		FCFS	RSACS		INCACS		ACS-DVRP		P-ACO		EIACO	
	Fitness	Runtime	Fitness	Fitness	Runtime	Fitness	Runtime	Fitness	Runtime	Fitness	Runtime	Fitness	Runtime
A1	8.45	0.69	9.50	8.28 (−)	0.67	9.58 (+)	0.24	8.29 (−)	0.72	8.28 (−)	0.89	**8.27** (−)	1.20
A2	5.61	1.20	6.07	**5.58** (−)	1.45	5.91 (+)	0.51	**5.58** (−)	1.61	5.59 (−)	1.68	**5.58** (−)	1.97
A3	**14.29**	0.91	20.84	16.10 (+)	1.15	16.77 (+)	0.41	16.28 (+)	1.29	14.91 (+)	1.38	14.99 (+)	1.73
A4	6.69	1.97	7.42	6.69 (≈)	2.71	7.01 (+)	0.94	**6.68** (≈)	3.05	6.71 (+)	3.13	**6.68** (≈)	3.32
A5	**3.62**	4.85	3.72	**3.62** (≈)	7.33	3.72 (+)	2.50	**3.62** (≈)	8.54	**3.62** (≈)	8.31	**3.62** (≈)	8.05
A6	**14.25**	2.82	19.59	17.92 (+)	4.48	17.84 (+)	1.58	18.20 (+)	5.34	16.43 (+)	5.26	16.43 (+)	5.43
A7	**3.84**	8.82	4.15	3.90 (+)	13.88	4.09 (+)	4.72	3.90 (+)	16.29	3.86 (+)	15.94	3.86 (+)	14.99
A8	**8.35**	17.05	13.89	10.60 (+)	30.45	11.63 (+)	10.26	10.59 (+)	34.01	9.49 (+)	36.37	9.51 (+)	35.28
Number of +/≈/−				4/2/2		8/0/0		4/2/2		5/1/2		4/2/2	



(a) Comparison on fitness values (b) Comparison on runtime (seconds)

Fig. 7. Performance comparison via histogram.

dynamic optimization algorithms. The histogram comparisons of Table V are illustrated in Fig. 7(a). It can be observed that:

1) FCFS performs the worst, because it solves the problem from a local perspective and cannot guarantee the solution quality at the global level.
2) INCACS performs better than FCFS, but performs the worst among the six dynamic optimization algorithms. This may be due to that INCACS only optimizes the new requests when an environmental change occurs. Therefore, it only focuses on the new information, but is regardless of the previous information, being easy to fall into local optima.
3) RSACS and ACS-DVRP have a similar performance, and perform worse than MACO. This means that simply transferring pheromone by a parameter cannot improve the solution quality very efficiently.
4) EIACO, which introduces elite immigrant ants, has no significant improvement compared to the original P-ACO. This means that the immigrants generated by the inver-over operator designed for TSP have no significant effect on the DEVD problem.
5) In the cases that the number of EVs is more than the number of customer requests (i.e., A1, A2, A4, A5, and A7 with rich EV resources), MACO can get results that are similar to the best values obtained by the other six algorithms, although slightly better or worse in some cases. However, MACO performs significantly better than the other six algorithms when the number of requests is the same as the number of EVs (i.e., A3, A6, and A8 with limited EV resources).

Moreover, to compare the computational efficiency of the six dynamic optimization algorithms, the average CPU runtime (in second) of each algorithm over 30 independent runs is also presented in Table V and the histogram is compared in Fig. 7(b). Notice that the runtime of FCFS is not presented because it is a kind of greedy algorithm that consumes very little CPU time. From the results, we can see that INCACS runs fastest because it only optimizes new customer requests. The runtime of RSACS and ACS-DVRP is similar, which is reasonable because ACS-DVRP runs the same mechanism as RSACS except for transferring pheromone from the previous environment by a parameter. The runtime of P-ACO is similar to that of RSACS and ACS-DVRP, indicating that the operation of the embedded population-list does not cause excessive computational time consumption. Compared to P-ACO, EIACO runs slightly longer because of the extra time spent by its elitism-based immigrant replacement strategy.

The MACO algorithm runs faster than other algorithms except INCACS. This means when new customer requests arrive, releasing some of the assigned customer requests around them can significantly reduce the runtime compared to the approaches that re-optimize all requests (e.g., the RSACS). Taking both fitness values and the runtime in Table V into consideration, it can be concluded that MACO can effectively make a trade-off between improving the solution quality and reducing computational time.

C. Necessity of Power Awareness and Recharging

In this paper, we consider that there are low-power EVs and they can be recharged during the service. To verify the necessity of vehicle recharging in the service, we design a comparative experiment. The experiment is designed based on two variants of the DEVD model: not considering the remaining power of the EV battery and not allowing the EV recharging. In the first model variant, the algorithm variant is denoted as MACO-full-power, where the remaining power of the EV battery is always regarded as sufficient no matter how much it remains. In the second model variant, the algorithm variant is denoted as MACO-w/o-recharging, where the recharging is not allowed for EVs during the service, that is, if the remaining power of an EV battery is not sufficient to complete a customer request, this EV will not be assigned to this customer. We compare MACO, MACO-full-power, and MACO-w/o-recharging in terms of fitness values, SatPercen, and UnsatPercen. The SatPercen (i.e., satisfied percentage) is the percentage of requests that can be served by EVs. However, in these customer-EV pairs, some pairs obtained by

TABLE VI
COMPARISONS OF THE FITNESS VALUES, SATPERCEN, AND UNSATPERCEN OF MACO, MACO-FULL-POWER, AND MACO-W/O-RECHARGING

Ins	MACO		MACO-full-power			MACO-w/o-recharging	
	Fitness	SatPercen	Fitness	SatPercen	UnsatPercen	Fitness	SatPercen
A1	8.45	100%	**7.00**	100%	26.67%	9.77	100%
A2	5.61	100%	**4.50**	100%	40.00%	6.04	100%
A3	14.29	100%	**11.33**	100%	28.28%	13.66	86%
A4	6.69	100%	**5.79**	100%	25.12%	6.79	100%
A5	3.62	100%	**2.91**	100%	34.74%	3.80	100%
A6	14.25	100%	**10.74**	100%	33.25%	12.87	85%
A7	3.84	100%	**3.21**	100%	28.46%	4.05	100%
A8	8.35	100%	**5.53**	100%	28.92%	7.73	90%

the MACO-full-power variant may be invalid. For example, for a low-power EV, the remaining power of its battery is regarded as sufficient in the MACO-full-power variant, but in the practical application, its remaining battery capacity is not sufficient to let it serve its assigned customer to the destination. In this case, the assignment is invalid and unsatisfied, and therefore the UnsatPercen (i.e., unsatisfied percentage) is the percentage of unsatisfied customer requests in all assigned customer requests. The results are given in Table VI.

It can be observed that MACO-full-power can get better fitness values than MACO does on all test cases, but has at least 25% unsatisfied percentage. It means that not all assigned EVs have enough power to serve the assigned customer requests. So when we dispatch EVs to customer requests, we need to consider the remaining power of the EV battery to avoid the invalid assignments. MACO-w/o-recharging obtains a worse global service quality (i.e., considering both the fitness values and satisfied percentage) than MACO does. On A1, A2, A4, A5, and A7, MACO-w/o-recharging can satisfy all customer requests, but has worse fitness values than MACO. On A3, A6, and A8, the fitness values of MACO-w/o-recharging are better than those of MACO, but not all customer requests can be assigned with an EV. It means that if the EV is not allowed to recharge in the service, some customer requests may not be served or take more time to complete. So it is necessary to consider the remaining power of the EV battery and recharging in the service for EVs.

D. Effectiveness of EoR Local Search and PR Strategy

To validate the effectiveness of the EoR local search, we integrate the EoR into the five compared dynamic optimization algorithms (i.e., except FCFS), termed RSACS-EoR, INCACS-EoR, ACS-DVRP-EoR, PACO-EoR, and EIACO-EoR, respectively. The fitness values obtained by these five algorithms are given in Table VII. To show the efficiency more directly, the fitness values of the EoR-enhanced algorithms are plotted as color bars in Fig. 8. Moreover, the improved performance brought by EoR is also plotted as the shaded bar. It can be observed that the EoR local search can improve the performance of all the five dynamic optimization algorithms to varying degrees. Nevertheless, the results in Table VII show that MACO is still the best algorithms among all the EoR algorithm variants.

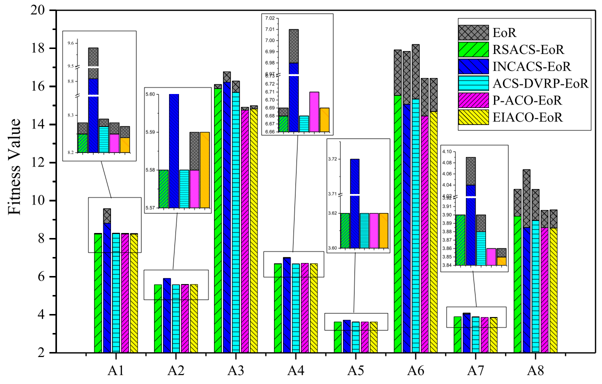


Fig. 8. Performance of EoR on different algorithms via histogram.

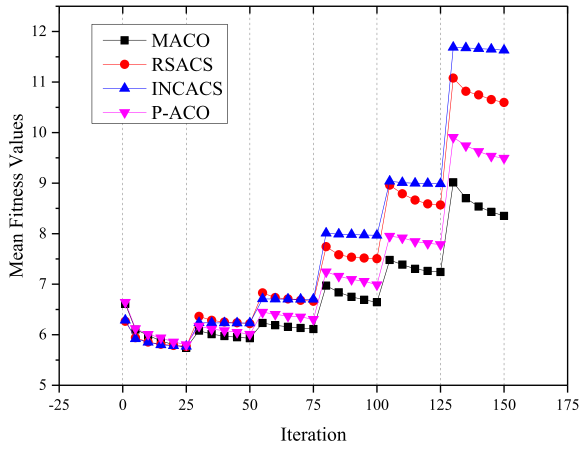


Fig. 9. Convergence curves of MACO, RSACS, INCACS, and P-ACO on A8.

To further investigate the effectiveness of EoR and PR strategies, we compare MACO with the MACO variants without the PR or EoR. There is no PR but EoR in MACO-w/o-PR, no EoR but PR in MACO-w/o-EoR, and no PR or EoR in MACO-w/o-PR-EoR. Table VIII lists the mean fitness values and runtime of the four algorithms over 30 independent runs. For the solution quality, it can be observed that MACO obtains similar results to MACO-w/o-PR, while outperforms MACO-w/o-EoR and MACO-w/o-PR-EoR. This means that the EoR local search helps MACO improve the solution quality. On the runtime, MACO runs slightly slower than MACO-w/o-EoR, which indicates the EoR local search consumes only a little computational time. Moreover, MACO runs much faster than MACO-w/o-PR and MACO-w/o-PR-EoR. It means the PR strategy helps improve computational efficiency on running time. Therefore, both EoR and PR help the MACO algorithm quickly find promising solutions in a short time. Based on the above, both EoR and PR have their contributions to the promising performance of MACO, and removing any of them will has bad influence on the performance of MACO.

E. Convergence Analysis

In order to conduct the convergence analysis, we plot the convergence curves of MACO, RSACS, INCACS, and P-ACO during the optimization process on A8 in Fig. 9. In the figure,

TABLE VII
COMPARISONS OF THE FITNESS VALUES BETWEEN MACO AND FIVE DYNAMIC OPTIMIZATION ALGORITHMS WITH EOR LOCAL SEARCH

Instance	MACO	RSACS-EoR		INCACS-EoR		ACS-DVRP-EoR		P-ACO-EoR		EIACO-EoR	
A1	8.45	8.25	(–)	8.81	(+)	8.27	(–)	8.25	(–)	**8.24**	(–)
A2	5.61	**5.58**	(–)	5.91	(+)	**5.58**	(–)	**5.58**	(–)	5.59	(–)
A3	**14.29**	15.89	(+)	16.22	(+)	15.69	(+)	14.75	(+)	14.82	(+)
A4	6.69	**6.68**	(≈)	6.98	(+)	**6.68**	(≈)	6.71	(+)	6.69	(≈)
A5	**3.62**	**3.62**	(≈)	3.72	(+)	**3.62**	(≈)	**3.62**	(≈)	**3.62**	(≈)
A6	**14.25**	15.53	(+)	15.06	(+)	15.34	(+)	14.45	(+)	14.68	(+)
A7	**3.84**	3.90	(+)	4.04	(+)	3.88	(+)	3.86	(+)	3.85	(+)
A8	**8.35**	9.18	(+)	8.59	(+)	8.96	(+)	8.58	(+)	8.56	(+)
Number of +/≈/–		4/2/2		8/0/0		4/2/2		5/1/2		4/2/2	

TABLE VIII
COMPARISONS OF THE FITNESS VALUES AND RUNTIME (SECOND) OF MACO, MACO-W/O-PR, MACO-W/O-EOR, AND MACO-W/O-PR-EOR

Instance	MACO		MACO-w/o-PR		MACO-w/o-EoR		MACO-w/o-PR-EoR	
	Fitness	Runtime	Fitness	Runtime	Fitness	Runtime	Fitness	Runtime
A1	8.45	0.69	**8.23** (–)	0.90	8.58 (+)	0.64	8.26 (–)	0.89
A2	5.61	1.20	**5.58** (–)	1.68	5.61 (≈)	1.19	**5.58** (–)	1.67
A3	**14.29**	0.91	14.49 (+)	1.38	14.46 (+)	0.98	14.77 (+)	1.37
A4	**6.69**	1.97	**6.69** (≈)	2.96	6.70 (≈)	1.92	**6.69** (≈)	2.94
A5	**3.62**	4.85	**3.62** (≈)	7.62	3.63 (+)	4.81	**3.62** (≈)	7.58
A6	**14.25**	2.82	14.53 (+)	4.77	15.57 (+)	2.80	16.33(+)	4.73
A7	**3.84**	8.82	**3.84** (≈)	14.27	3.85 (+)	8.73	3.85 (+)	14.18
A8	**8.35**	17.05	8.38 (≈)	31.27	9.40 (+)	16.77	9.60 (+)	30.91
Number of +/≈/–			2/4/2		6/2/0		4/2/2	

every time a sharp peak appears, it means that a dynamic change occurs. In the first 25 iterations, which is the initial environment, the results obtained by the four algorithms are similar. In the next iterations, MACO can quickly find new promising solutions when an environmental change occurs, compared to the other three algorithms. This may be due to two advantages brought by the memory archive information in MACO. Firstly, after being repaired, the solutions in the memory archive will have good fitness values (i.e., low average time cost) in the new environment. Since the environment before and after a dynamic change is relevant in the DEVD application, such a re-use of the past solutions can help the algorithm converge faster in the new environment. Secondly, by transferring pheromone from the previous environment via the solutions in the memory, the MACO can guide ants to search promising areas of the new environment quickly. The fastest convergence speed and best fitness value obtained by MACO show the good utilization of the memory archive information of our proposed MACO algorithm.

F. Parameter Analysis of MACO

In this section, we take A3 to A8 as examples to study the influences of parameters on MACO. The parameters q_0 and β are set according to standard ACO, and then we investigate the other parameters K, R, and $\tau_{\max}$. Note that when performing analysis of a parameter, the other parameters are consistent with the settings in Table III.

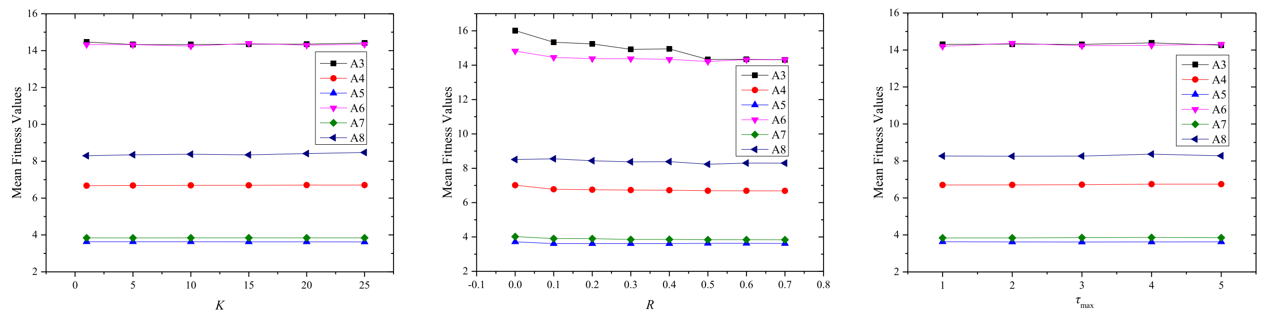
K represents the size of the memory, that is, the number of stored solutions that perform well in the previous iterations. The mean fitness values obtained by MACO with different K values (i.e., $K = 1$, 5, 10, 15, 20, and 25) are plotted in Fig. 10(a). It is shown that the MACO performs slightly worse on A3 and A6 when $K = 1$. This indicates that too little information stored in the memory archive will lead to inefficient pheromone update. In addition, the performance of MACO is similar when K is 5 to 25. Therefore, we set $K = 10$ in this paper.

Then the parameter R is investigated, which determines the number of customer-EV pairs released in the historically best solution when the environment changes. We set R from 0 to 0.7 with an interval of 0.1 and the mean fitness values are shown in Fig. 10(b). The results when $R = 0$ are similar to those obtained by INCACS-EoR in Table VIII. This is reasonable because they adopt the same mechanism to deal with dynamic changes, that is, only optimizing new requests. The fitness values generally decrease with the increase of R, which indicates that releasing some assigned customer-EV pairs can help escape from local optima. The performance improvement is not significant when R is greater than 0.5. Therefore, to avoid the waste of computational time caused by releasing too many customer-EV pairs, R is set to 0.5 in this paper.

Finally, the maximum pheromone value $\tau_{\max}$ is tested. We set $\tau_{\max}$ from 1.0 to 5.0 with a step of 1.0. The tendency of the curves in Fig. 10(c) shows that the $\tau_{\max}$ value has little effect on the performance of MACO. In this paper, $\tau_{\max}$ is set to 1.0.

G. Transportation Costs

From the perspective of the enterprises, it is important to reduce the transportation costs of EV during service. In this section, we compare the transportation cost obtained by each



(a) (b) (c)

Fig. 10. Influence of the parameters K, R, and $\tau_{\max}$ on the MACO. (a) Different K values. (b) Different R values. (c) Different $\tau_{\max}$ values.

TABLE IX
COMPARISONS OF TOTAL DRIVING DISTANCE (km)

Instance	MACO	FCFS	RSACS	INCACS	ACS-DVRP	P-ACO	EIACO
A1	2608.17	2648.34	2602.86	2651.51	**2601.96**	2602.94	2603.27
A2	2718.22	2736.20	2717.35	2728.99	2717.36	2717.40	**2717.34**
A3	**6021.55**	6361.72	6126.78	6165.60	6132.61	6041.46	6043.12
A4	5590.45	5646.41	5589.56	5616.77	**5588.92**	5592.40	5590.29
A5	5368.04	5379.67	5367.89	5379.70	**5367.48**	5367.92	5367.51
A6	**12089.30**	12382.90	12340.50	12302.50	12348.70	12236.70	12245.60
A7	**10574.10**	10624.50	10583.70	10614.60	10582.50	10577.40	10576.20
A8	**27797.40**	28241.70	28071.00	28225.60	28161.00	27958.90	28012.60

algorithm. Table IX lists the total driving distance (km) of all the EVs, including the distance to pick up the customer and the distance to deliver the customer to the destination, calculated according to the dispatch results on all the eight test cases A1 to A8.

It can be observed from Table IX that MACO can obtain the minimal transportation cost or the approximate minimal cost among all the seven algorithms. In particular, MACO performs significantly better than other algorithms in A6 and A8, which is consistent with the results in Table V.

Therefore, it can be concluded that MACO can effectively solve the DEVD problem. It can reduce customers' waiting time and improve the quality of service. Meanwhile, it can reduce the driving distance of EVs during the service process, so as to reduce operating costs.

H. Comparison on Real-World Dataset

To evaluate the performance of MACO on real-world application, a real dataset from Didi Chuxing GAIA Initiative [59] is adopted. The real-world data include all the customer requests in a whole day in a China city. Herein, we select the first 500 customer requests (i.e., the same number of customers as our test case A8) from the dataset to conduct a real-world case, termed as DidiTest. However, the data only contain the original locations and destinations of the customers, which are located in a 55 km × 30 km physical area. Therefore, in order to complete the DidiTest case, we further set up the number of EVs and charging stations as 500 and 50, respectively, the same as those in A8. Then, the locations of EVs and the charging stations are generated randomly within the coordinate range of the customer locations, while the charging data is generated randomly within the battery capacity of EVs. We compare the mean fitness values and the transportation costs obtained by each algorithm on this DidiTest test case. The experimental results are compared in Table X.

TABLE X
COMPARISONS OF THE FITNESS VALUES AND TOTAL DRIVING DISTANCE (km) ON THE TEST CASE DIDITEST

Algorithms	Fitness Values	Driving Distance
RSACS	18.17	9572.77
INCACS	18.35	9645.29
ACS-DVRP	18.19	9582.76
P-ACO	18.31	9625.76
EIACO	18.41	9663.88
MACO	**18.00**	**9508.90**

It can be observed that MACO can get the best fitness values and obtain the minimal transportation costs (i.e., the total driving distances of all the EVs, including the distances to pick up the customers and the distances to deliver the customers to the destinations) among all comparison algorithms on the real-world test case. Thus, it can be concluded that MACO is practical and effective in solving the real DEVD problem.

V. CONCLUSION

EV dispatch is a challenging issue due to the charging characteristics and dynamic scenarios. To simulate real-world dynamic EV dispatch application scenarios, a dynamic EV dispatch model is established by considering multi-source data association from five sources, including customer, vehicle, charging, station, and service. To solve the DEVD problem, we propose a memory-based ACO approach MACO as the optimizer to enhance the adaptability in dynamic environments. Furthermore, the PR strategy and the EoR local search procedure are incorporated into MACO to obtain a better balance between the performance and execution time. The PR strategy gets the requests to be optimized in the new environment to better respond to the dynamic environment. The EoR local search procedure integrates empirical knowledge into the search process to enhance the performance of MACO.

Experimental results show that MACO outperforms the traditional FCFS approach and some state-of-the-art ACO-based dynamic optimization algorithms. It has effectively achieved the objectives of minimizing customer waiting time and transportation costs at the global level. The experimental results on the real-world dataset also show the practicability of MACO.

In the future work, we can consider the personalized preferences of both the passengers and drivers on the EV recharging to make the problem model more practical. Some other promising future research directions include the scheduling over larger areas by dividing the dispatch area and the scheduling with more objectives. For these goals, some recent large-scale optimization algorithms [60] and multi-objective optimization algorithms [61] are worth studying.

References

[1] B.-J. Tang, X.-Y. Li, B. Yu, and Y.-M. Wei, "How app-based ride-hailing services influence travel behavior: An empirical study from China," *Int. J. Sustain. Transp.*, vol. 14, no. 7, pp. 554–568, Jul. 2020.

[2] H. Yuan, M. Zhou, Q. Liu, and A. Abusorrah, "Fine-grained resource provisioning and task scheduling for heterogeneous applications in distributed green clouds," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 5, pp. 1380–1393, Sep. 2020.

[3] C. C. Chan, A. Bouscayrol, and K. Chen, "Electric, hybrid, and fuel-cell vehicles: Architectures and modeling," *IEEE Trans. Veh. Technol.*, vol. 59, no. 2, pp. 589–598, Feb. 2010.

[4] *DIDI Released the First Customized Car D1*. Accessed: Nov. 16, 2020. [Online]. Available: https://www.didiglobal.com/news/newsDetail?id=973

[5] A. Jenn, "Emissions benefits of electric vehicles in Uber and Lyft ride-hailing services," *Nature Energy*, vol. 5, no. 7, pp. 520–525, Jun. 2020.

[6] J. Wang, Y. Sun, Z. Zhang, and S. Gao, "Solving multitrip pickup and delivery problem with time windows and manpower planning using multiobjective algorithms," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 4, pp. 1134–1153, Jul. 2020.

[7] S. Hulagu and H. B. Celikoglu, "Environment-friendly school bus routing problem with heterogeneous fleet: A large-scale real case," *IEEE Trans. Intell. Transp. Syst.*, early access, Nov. 17, 2020, doi: 10.1109/TITS.2020.3036696.

[8] A. K. Madhusudhanan and X. Na, "Effect of a traffic speed based cruise control on an electric vehicle's performance and an energy consumption model of an electric vehicle," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 386–394, Mar. 2020.

[9] R. Abousleiman, O. Rawashdeh, and R. Boimer, "Electric vehicles energy efficient routing using ant colony optimization," *SAE Int. J. Alternative Powertrains*, vol. 6, no. 1, pp. 1–14, Apr. 2017.

[10] R. Atat, M. Ismail, E. Serpedin, and T. Overbye, "Dynamic joint allocation of EV charging stations and DGs in spatio-temporal expanding grids," *IEEE Access*, vol. 8, pp. 7280–7294, 2020.

[11] H. Ma, Z. Yang, P. You, and M. Fei, "Multi-objective biogeography-based optimization for dynamic economic emission load dispatch considering plug-in electric vehicles charging," *Energy*, vol. 135, pp. 101–111, Sep. 2017.

[12] T. Ghanbarzadeh, S. Goleijani, and M. P. Moghaddam, "Reliability constrained unit commitment with electric vehicle to grid using hybrid particle swarm optimization and ant colony optimization," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, Jul. 2011, pp. 1–7.

[13] M. Mavrovouniotis, G. Ellinas, and M. Polycarpou, "Ant colony optimization for the electric vehicle routing problem," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 1234–1241.

[14] S. Hulagu and H. B. Celikoglu, "An electric vehicle routing problem with intermediate nodes for shuttle fleets," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1223–1235, Feb. 2022.

[15] H. Mao, J. Shi, Y. Zhou, and G. Zhang, "The electric vehicle routing problem with time windows and multiple recharging options," *IEEE Access*, vol. 8, pp. 114864–114875, 2020.

[16] C. M. Martinez, X. Hu, D. Cao, E. Velenis, B. Gao, and M. Wellers, "Energy management in plug-in hybrid electric vehicles: Recent progress and a connected vehicles perspective," *IEEE Trans. Veh. Technol.*, vol. 66, no. 6, pp. 4534–4549, Jun. 2017.

[17] D.-H. Lee, H. Wang, R. Cheu, and S. Teo, "Taxi dispatch system based on current demands and real-time traffic conditions," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1882, pp. 193–200, Jan. 2004.

[18] M. Maciejewski, J. Bischoff, and K. Nagel, "An assignment-based approach to efficient real-time city-scale taxi dispatching," *IEEE Intell. Syst.*, vol. 31, no. 1, pp. 68–77, Jan. 2016.

[19] L. Zhang *et al.*, "A taxi order dispatch model based on combinatorial optimization," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 2151–2159.

[20] *Introduction to DIDI Chuxing*. Accessed: Aug. 1, 2020. [Online]. Available: https://en.wikipedia.org/wiki/DiDi

[21] K. T. Seow, N. H. Dang, and D.-H. Lee, "A collaborative multiagent taxi-dispatch system," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 607–616, Jul. 2010.

[22] F. Miao *et al.*, "Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 463–478, Apr. 2016.

[23] L. Hu and J. Dong, "An artificial-neural-network-based model for real-time dispatching of electric autonomous taxis," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1519–1528, Feb. 2022.

[24] J. Shi, Y. Gao, W. Wang, N. Yu, and P. A. Ioannou, "Operating electric vehicle fleet for ride-hailing services with reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4822–4834, Nov. 2020.

[25] D. Liang, Z.-H. Zhan, Y. Zhang, and J. Zhang, "An efficient ant colony system approach for new energy vehicle dispatch problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4784–4797, Nov. 2020.

[26] T. Liao, K. Socha, M. A. Montes de Oca, T. Stützle, and M. Dorigo, "Ant colony optimization for mixed-variable optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 503–518, Aug. 2014.

[27] Q. Yang *et al.*, "Adaptive multimodal continuous ant colony optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 191–205, Apr. 2017.

[28] Z. Wang, H. Xing, T. Li, Y. Yang, R. Qu, and Y. Pan, "A modified ant colony optimization algorithm for network coding resource minimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 325–342, Jun. 2016.

[29] Z.-H. Zhan, L. Shi, K. C. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artif. Intell. Rev.*, vol. 55, no. 1, pp. 59–110, Jan. 2022.

[30] A. H. Khan, X. Cao, S. Li, V. N. Katsikis, and L. Liao, "BAS-ADAM: An Adam based approach to improve the performance of beetle antennae search optimizer," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 461–471, Mar. 2020.

[31] J.-R. Jian, Z.-G. Chen, Z.-H. Zhan, and J. Zhang, "Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 779–793, Aug. 2021.

[32] X. Zhang, Z.-H. Zhan, W. Fang, P. Qian, and J. Zhang, "Multi population ant colony system with knowledge based local searches for multiobjective supply chain configuration," *IEEE Trans. Evol. Comput.*, early access, Jul. 15, 2021, doi: 10.1109/TEVC.2021.3097339.

[33] 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.

[34] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. L. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.

[35] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.

[36] R.-H. Huang and T.-H. Yu, "An effective ant colony optimization algorithm for multi-objective job-shop scheduling with equal-size lot-splitting," *Appl. Soft Comput.*, vol. 57, pp. 642–656, Aug. 2017.

[37] M. Mavrovouniotis, G. Ellinas, and M. Polycarpou, "Electric vehicle charging scheduling using ant colony system," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 2581–2588.

[38] W. Luan, G. Liu, C. Jiang, and M. Zhou, "MPTR: A maximal-marginal-relevance-based personalized trip recommendation method," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 11, pp. 3461–3474, Nov. 2018.

[39] Y. Feng, M. Zhou, G. Tian, Z. Li, Z. Zhang, Q. Zhang, and J. Tan, "Target disassembly sequencing and scheme evaluation for CNC machine tools using improved multiobjective ant colony algorithm and fuzzy integral," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 12, pp. 2438–2451, Dec. 2019.

[40] M. Mavrovouniotis, C. Li, G. Ellinas, and M. Polycarpou, "Parallel ant colony optimization for the electric vehicle routing problem," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2019, pp. 1660–1667.

[41] Y. Li, H. Soleimani, and M. Zohal, "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives," *J. Cleaner Prod.*, vol. 227, pp. 1161–1172, Aug. 2019.

[42] X. Wang, T.-M. Choi, H. Liu, and X. Yue, "Novel ant colony optimization methods for simplifying solution construction in vehicle routing problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3132–3141, Nov. 2016.

[43] X. Situ *et al.*, "A parallel ant colony system based on region decomposition for taxi-passenger matching," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 960–967.

[44] N. Zhu, Y. Liu, S. Ma, and Z. He, "Mobile traffic sensor routing in dynamic transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2273–2285, Oct. 2014.

[45] L. Wang and J. Shen, "Multi-phase ant colony system for multi-party data-intensive service provision," *IEEE Trans. Serv. Comput.*, vol. 9, no. 2, pp. 264–276, Mar./Apr. 2016.

[46] G. Li, L. Boukhatem, and J. Wu, "Adaptive quality-of-service-based routing for vehicular ad hoc networks with ant colony optimization," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3249–3264, Apr. 2017.

[47] M. Mavrovouniotis, F. M. Müller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Trans. Cybern.*, vol. 47, no. 7, pp. 1743–1756, Jul. 2017.

[48] L. M. Gambardella, É. D. Taillard, and M. Dorigo, "Ant colonies for the quadratic assignment problem," *J. Oper. Res. Soc.*, vol. 50, no. 2, pp. 167–176, Feb. 1999.

[49] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.

[50] M. Guntsch and M. Middendorf, "A population based approach for ACO," in *Proc. 2nd Eur. Workshop Evol. Comput. Combinat. Optim.*, 2002, pp. 72–81.

[51] M. Guntsch and M. Middendorf, "Applying population based ACO to dynamic optimization problems," in *Proc. 3rd Int. Workshop ANTS*, 2002, pp. 111–122.

[52] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *J. Combinat. Optim.*, vol. 10, no. 4, pp. 327–343, Dec. 2005.

[53] M. Mavrovouniotis and S. Yang, "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors," *Appl. Soft Comput.*, vol. 13, no. 10, pp. 4023–4037, 2013.

[54] R. Cheng, M. N. Omidvar, A. H. Gandomi, B. Sendhoff, S. Menzel, and X. Yao, "Solving incremental optimization problems via cooperative coevolution," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 762–775, Oct. 2019.

[55] M. Guntsch, M. Middendorf, and H. Schmeck, "An ant colony optimization approach to dynamic TSP," in *Proc. Genet. Evol. Comput., Conf.*, 2001, pp. 860–867.

[56] G. Tao and Z. Michalewicz, "Inver-over operator for the TSP," in *Proc. Parallel Problem Solving Nature*, 1998, pp. 803–812.

[57] H. Yang, S. Yang, Y. Xu, E. Cao, M. Lai, and Z. Dong, "Electric vehicle route optimization considering time-of-use electricity price by learnable partheno-genetic algorithm," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 657–666, Mar. 2015.

[58] *Performance Characteristics of BYD e6*. Accessed: Aug. 1, 2020. [Online]. Available: https://en.wikipedia.org/wiki/BYD_e6

[59] *DIDI Chuxing GAIA Initiative*. Accessed: Apr. 26, 2021. [Online]. Available: https://gaia.didichuxing.com

[60] J.-R. Jian, Z.-H. Zhan, and J. Zhang, "Large-scale evolutionary optimization: A survey and experimental comparative study," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 3, pp. 729–745, Mar. 2020.

[61] P. Wu, F. Chu, A. Che, and M. Zhou, "Bi-objective scheduling of fire engines for fighting forest fires: New optimization approaches," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1140–1151, Apr. 2018.

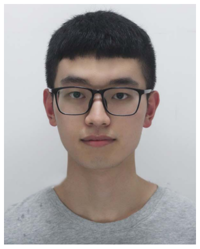
Lin Shi (Student Member, IEEE) received the B.S. degree from the South China University of Technology, Guangzhou, China, in 2018, where she is currently pursuing the Ph.D. degree.

Her current research interests include artificial intelligence, evolutionary computation, swarm intelligence, and their applications in intelligent transportation.

Zhi-Hui Zhan (Senior Member, IEEE) received the bachelor's and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

He is currently the Changjiang Scholar Young Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems in intelligent transportation and smart cities.

Dr. Zhan was a recipient of the IEEE Computational Intelligence Society (CIS) Outstanding Early Career Award in 2021, the Outstanding Youth Science Foundation from the National Natural Science Foundation of China (NSFC) in 2018, and the Wu Wen-Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. His doctoral dissertation was awarded the IEEE CIS Outstanding Ph.D. Dissertation and the China Computer Federation Outstanding Ph.D. Dissertation. He is one of the World's Top 2% Scientists for both Career-Long Impact and Year Impact in Artificial Intelligence and one of the Highly Cited Chinese Researchers in computer science. He is also an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the *Neurocomputing*, and the *Memetic Computing*.



Di Liang (Student Member, IEEE) received the B.S. and M.S. degrees in computer science from the South China University of Technology, Guangzhou, China, in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree with the School of Engineering, Westlake University, Hangzhou, China.

His current research interests include speech processing, deep learning, and speaker recognition.

Jun Zhang (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Kowloon, Hong Kong, in 2002.

He is currently a Korea Brain Pool Fellow Professor with Hanyang University, South Korea, and a Visiting Professor with the Chaoyang University of Technology. His current research interests include computational intelligence, cloud computing, operations research, and power electronic circuits. He has published over more than 150 IEEE TRANSACTIONS papers in his research areas.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, the National Science Fund for Distinguished Young Scholars of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is also an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.