An Efficient Ant Colony System Approach for New Energy Vehicle Dispatch Problem

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

***Abstract*— As a powerful measure to alleviate greenhouse gas emissions and global warming issue, new energy vehicle (NEV) has aroused extensive attention from the whole society in recent years. In the past few decades, many studies have been conducted on the dispatch of traditional fuel-driven vehicles. As a means of transportation, NEV has the characteristics of fuel-driven vehicles, but the dispatch is different because of its unique refueling manner. With the popularization of NEV, its unique dispatch research is imminent. This paper comprehensively considers electricity and charging piles during the NEV dispatch (NEVD) process. An NEVD framework containing a novel dispatch model is proposed, which elaborates the application service of NEV. To the best of our knowledge, this study is the first to combine NEVD with service system. Based on the formulated model, an efficient ant colony system (EACS) approach enhanced by pre-selection strategy and local pruning strategy is designed to dispatch NEVs to passengers. Experiments are carried out to investigate the applicable scenarios of ACS-based algorithms. The results verify that the proposed EACS algorithm is an effective and efficient approach to solve the NEVD problem.**

***Index Terms*— New energy vehicle dispatch (NEVD), ant colony system (ACS), pre-selection, local pruning.**

I. Introduction

With the growing travel needs and the development of mobile internet technology, online taxi booking has become very popular in current daily life [1]. Meanwhile, environmental protection policies have prompted the rise of new energy vehicle (NEV), making online booking service for NEV as a new trend. One representative commercial application is Caocao [2], which is a famous NEV online booking service in China. Different from traditional fuel-driven vehicles, electric-powered vehicles primarily rely on charging piles for energy supply and have different charging modes, such as fast-charging and regular-charging. In the actual traffic planning, factors such as charging facilities and battery characteristics are needed to be taken into account, which brings new challenges to the planning and design approaches that are oriented to traditional fuel-driven vehicles [3]. For example, as the charging power is limited, for fast-charging or regular-charging, the charging time needs to be considered. However, the fueling time of fuel-driven vehicles is very short and can be neglected. Research efforts on NEV mainly focus on energy management system [4]–[8], intelligent charging system [9]–[13], and transportation design such as optimizing charging station locations [14]–[16] and route optimization [17]–[19]. However, there are few studies on booking dispatch of NEV, which is exactly the primary focus of this paper.

Vehicle dispatch is usually performed by a centralized service platform in order to facilitate the unified management of vehicle resources. This global dispatch mode provides customers with strong security and high efficiency, while it also imposes heavy computational burden to the dispatch center. A robust dispatch approach is needed to support the allocation of daily service requests. The vehicle dispatch methods in the literature are mainly for traditional fuel-driven taxi [20]–[22]. That is, only the locations of vehicles and customers are considered when dispatching vehicles for bookings.

Some previous work focuses on a first-come-first-served (FCFS) approach [1], [23]. When a request comes, dispatch system assigns it to the nearest taxi or a shortest-travel-time taxi without considering whether this assignment will affect subsequent requests. The FCFS approach can reduce response time, but it cannot guarantee the global optimal allocation for all requests. As described in [23], in rush hours, over a hundred thousand passengers need to be matched to taxis every second by Didi Chuxing (a ride-hailing service platform in China). Therefore, in a certain area, requests within a small time window (e.g., 5 seconds) can be processed simultaneously. For doing this, Seow *et al.* [20] partition geographical region into some logical dispatch areas and propose a collaborative multi-agent system to automate taxi dispatch in a distributed fashion. The agents on behalf of taxi drivers conduct intragroup negotiation, so as to minimize the total travel time. However, they only consider scenarios where the number of requests is equal to that of taxis. Moreover, some heuristic algorithms are proposed to solve taxi dispatch problem [24]–[26]. Jung *et al.* [24] formulate a dynamic shared-taxi problem and apply a hybrid

Manuscript received April 16, 2019; revised July 30, 2019; accepted October 2, 2019. Date of publication October 17, 2019; date of current version October 30, 2020. This work was supported in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundations of China (NSFC) under Grant 61772207 and Grant 61873097, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the Guangdong-Hong Kong Joint Innovation Platform under Grant 2018B050502006. The Associate Editor for this article was A. Che. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

D. Liang and Z.-H. Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou 510006, China (e-mail: zhanapollo@163.com).

Y. Zhang and J. Zhang are with Victoria University, Melbourne, VIC 8001, Australia (e-mail: junzhang@ieee.org).

Digital Object Identifier 10.1109/TITS.2019.2946711

approach combining simulated annealing and insertion heuristic to assign passenger requests in real time.

As a kind of evolutionary computation optimization algorithms, ant colony optimization (ACO) [27] is notable for its adaptability and global search capability. Moreover, ant colony system (ACS) is an efficient ACO variant that is firstly proposed by Dorigo and Gambardella [28] to solve the traveling salesman problem (TSP), which is a typical combinatorial optimization problem (COP). Many studies have shown that ACO/ACS can effectively solve real-world COPs, such as aircraft arrival sequencing and scheduling [29], cloud workflow scheduling [30], [31], virtual machine placement [32], [33], water distribution system design [34], and vehicle routing problem [35]. In addition, a parallel ACS is performed based on region decomposition to optimize taxi-passenger matching in [26]. As NEV dispatch (NEVD) problem can be modeled as a COP similar to taxi dispatch (i.e., taxi-passenger matching) problem, ACS is applicable and suitable to solve it. Nevertheless, the NEVD problem is much more challenge than traditional fuel-driven taxi dispatch due to its more complex constraints such as electricity and charging facilities. Therefore, a more efficient ACS approach is in great need.

This paper makes an attempt to analyze the commercial application background and propose an NEVD framework, which elaborates the application mode of NEVD from the underlying facilities to the upper-level commercial objectives. Based on the actual traffic operation scenarios of NEV, we propose a novel dispatch model that takes into account the traffic factors such as electricity and charging facilities. To solve the NEVD problem, an efficient ACS (EACS) is designed by combining pre-selection strategy and local pruning strategy.

The remainder of this paper is organized as follows. Section II introduces and analyzes the NEVD framework. Section III outlines a model of NEVD problem. Section IV develops the EACS algorithm in detail. Section V shows the experimental process and gives a discussion about the applicability and efficiency of EACS by comparing it with other algorithms. Finally, conclusions and future work are drawn in Section VI.

II. NEVD System

The framework of NEVD system is depicted in Fig. 1, which demonstrates the service of NEV, with emphasis on the dispatch model and solution. The framework consists of the following six layers.

A. Facility Layer

As the basis of this framework, facility layer is composed of underlying hardware devices and network technologies that support intellisense. Global positioning system (GPS) is used to track NEVs and customers in real time. A wireless access point is a networking hardware device that allows wireless terminals to connect to a wired network. It can upload data such as electricity of NEVs acquired by wireless sensors. Customers send service requests and receive feedback from the dispatch center via mobile devices. A wireless communication network is a medium for information transmission.

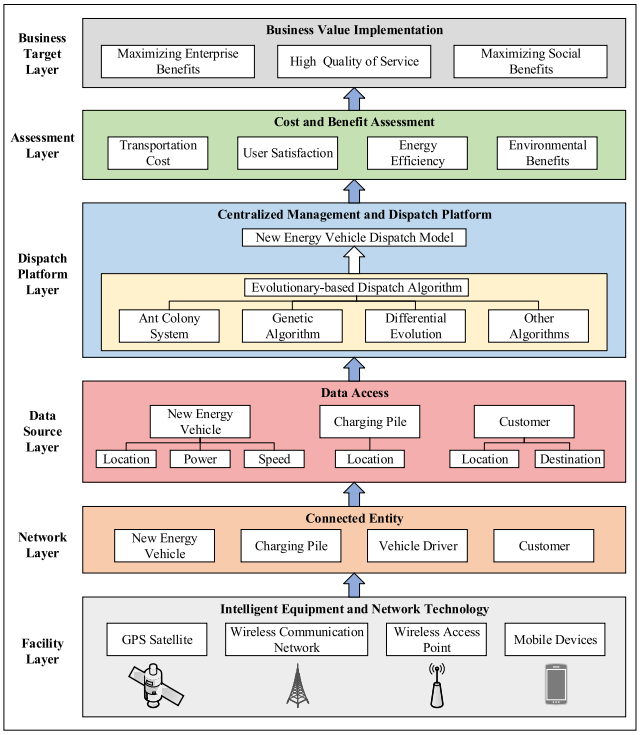


Fig. 1. The proposed NEVD framework.

B. Network Layer

Network layer lists entities connected by Internet of Things (IoT) technology, including NEVs, charging piles, drivers, and customers. Among them, the energy supply of NEVs mainly includes electric, hybrid electric, and fuel-cell [5]. Only the electric vehicle is considered in this paper. As an energy-supplement device, charging piles can provide power sources for NEVs. Through the underlying equipment and technology, these entities are connected as a whole for information sharing.

C. Data Source Layer

Through IoT technology, entities can exchange data with each other, which is shown in data source layer. The NEV-related data include location, speed, and power. When customers send requests by mobile phones, the dispatch center will acquire their current location and destination information. In addition, the location information of charging piles should be known. These data reflect the characteristics of each entity and provide support for upper-layer dispatch.

D. Dispatch Platform Layer

Dispatch platform layer is the core layer of the dispatch framework, which provides decision-making basis for unified and centralized dispatch. Based on the dispatch-related data acquired from data source layer, a dispatch model of NEV is established. Compared with the traditional taxi-passenger matching model, it considers the remaining electricity of NEVs, charging piles, and charging characteristics of the battery.

As a novel variant of taxi dispatch, the NEVD problem is an NP-hard problem, which can be solved by many evolutionary computation algorithms such as ACS, genetic algorithm (GA), and differential evolution algorithm. In this paper, we consider ACS as a powerful tool for solving NEVD. Pre-selection and local pruning strategies are integrated into ACS to improve its performance.

E. Assessment Layer

After the dispatch model and the solution are designed, the decision results need to be assessed, which is performed by assessment layer. Since NEVD is service oriented, the quality of NEVD design should be evaluated by service quality. For NEV, the implementation of the service is assessed according to four aspects in this paper: 1) customer satisfaction; 2) transportation cost; 3) energy efficiency; and 4) environmental benefits. Customer satisfaction consists of two levels: one is whether a request is accepted; another is customer waiting time. During transportation, the consumption of energy such as electricity, and the damage of equipment such as NEVs and batteries constitute transportation cost. Energy efficiency and environmental benefits are related to fuel consumption and carbon emissions, respectively.

F. Business Target Layer

Corresponding to the evaluation criteria of assessment layer, business target layer shows the ultimate commercial goal of the framework, which includes maximizing corporate profits, providing customers with high-quality service, and maximizing social benefits.

The NEVD framework combines the dispatch of NEV with application services to perform the matching task between service requests and NEVs. Moreover, this framework is scalable, it can implement different system functionalities or targets by changing the interfaces of certain layers. For example, in the research and development stage of NEV, it is vital to study the battery power to improve charging rate and reduce energy consumption, so as to improve energy efficiency. In the operation stage, if we expect to improve user experience, the cleanliness of NEVs or the friendliness of drivers should be taken into account in the dispatch platform layer.

This paper focuses on the establishment of NEVD model and the study of ACS-based dispatch algorithms. The two aspects will be highlighted in the following sections.

III. NEVD Modeling

Consider a real-world scenario, there are multiple NEVs and charging piles scattered in a certain geographical area. The location of each pile is fixed and the NEVs can be tracked in real time by GPS satellites. When some customers request NEV service in a given time window, these requests with destination information are sent to the dispatch center through the wireless communication network. The dispatch center assigns NEV service to all the customer requests based on a dispatch approach, so as to maximize total customer satisfaction. An example of NEVD scenarios is depicted in Fig. 2 where each customer is matched with the nearest NEV.

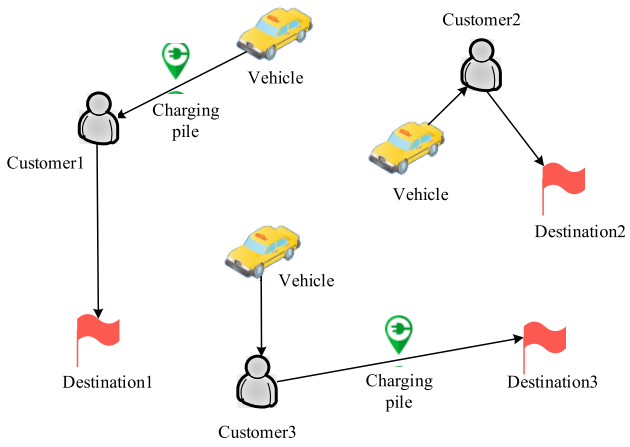


Fig. 2. A scenario of NEVD.

TABLE I
BASIC NOTATIONS FOR THE NEVD PROBLEM

Symbol	Definition
l	Side length of the scheduling domain.
N	Number of customers.
c_i	ith customer.
t_i	Destination of c_i.
$E(t_i)$	Whether there is a charging pile on the way to t_i.
M	Number of NEVs.
v_j	jth NEV.
e_j	Electricity of v_j.
$D(e_j)$	Distance that e_j can support, measured by kilometer.
$d(i,j)$	Distance between i and j, measured by kilometer.
s	Speed of NEVs.
$k(cr)$	Distance that an NEV can travel per unit time of charging, in km/min.
tl	Time limit that customers can tolerate.

The basic notations for the NEVD problem are listed in Table I. We consider the matching of N service requests (customers) to M NEVs in the area of $l \times l$ size that are made in a time window. It should be noted that the cancellation of customer requests is not considered in the NEVD model. In fact, if there is any request cancelled during the time window, this request will not be considered or processed by the dispatch algorithm. If there is any request cancelled during the execution of the dispatch algorithm, this request is still matched to NEV, but the matching result will not be fed back to the corresponding customer. Moreover, if there is any new request coming during the execution of the dispatch algorithm, it can be processed in the next time window.

Since NEVs rely on electricity for energy supply, it is necessary to consider whether the remaining electricity can take customers to their destinations during the dispatch process. In order to elaborate the *electricity constraint*, we define two scenarios that an NEV v_j $(1 \le j \le M)$ satisfies the request made by customer c_i $(1 \le i \le N)$. 1) The NEV itself has enough energy to take c_i to the destination, denoted as

$$D(e_j) \ge d(c_i, v_j) + d(c_i, t_i) \tag{1}$$

where $D(e_j)$ is the physical distance that the electricity e_j can support, measured by kilometer, and $d(c_i, t_i)$ is the distance

between departure location and destination of c_i. 2) In the case of insufficient energy, there is a charging pile on the way to t_i (i.e., the destination of c_i), and the required charging time is not more than time limit that customers can tolerate (denoted as tl):

$$E(t_i) = 1 \tag{2a}$$

$$\frac{d(c_i, v_j) + d(c_i, t_i) - D(e_j)}{k(cr)} \leq tl \tag{2b}$$

where $k(cr)$ represents the distance that an NEV can travel per unit time of charging. In order to shorten the charging time, a flexible mode is adopted, that is, the amount of charging is determined according to the needs.

This paper focuses primarily on total customer satisfaction, which can be measured from two levels. One is global acceptance rate, that is, the more requests accepted, the higher total customer satisfaction. The other is total customer waiting time, that is, shorter waiting time means higher customer satisfaction. The waiting time consists of the travel time of an NEV v_j to reach the departure location of customer c_i and the time to wait for charging if necessary, which is calculated by

$$wt(c_i, v_j) = \begin{cases} \dfrac{d(c_i, v_j)}{s}, & \text{if } D(e_j) \geq d(c_i, v_j) + d(c_i, t_i) \\ \dfrac{d(c_i, v_j)}{s} + \dfrac{d(c_i, v_j) + d(c_i, t_i) - D(e_j)}{k(cr)} & \\ \quad \text{else if } E(t_i) = 1 & \\ \quad \text{and } \dfrac{d(c_i, v_j) + d(c_i, t_i) - D(e_j)}{k(cr)} \leq tl & \\ \text{NA}, \quad \text{otherwise} & \end{cases} \tag{3}$$

where "NA" means that for a customer-NEV pair that does not satisfy the electricity constraint, i.e., constraint (1) or (2), it is not regarded as a candidate and therefore the waiting time is not calculated. In real-world scenarios, vehicle travel time is calculated using real-time traffic information, but for the sake of clarity, it is assumed that the traffic condition is static so that a shorter distance means a shorter driving time. Moreover, since the distance between a customer and its destination is not influenced by the dispatch results, it is not considered in the waiting time. To further simplify the model, the situation after an NEV has reached its destination is not considered in this paper. That is, if an NEV satisfies the electricity constraint of one customer c_i, it is a legal candidate to be assigned to c_i.

In order to better combine the two levels of satisfaction measurement, we define a '*logical distance*' between customers and NEVs. It combines the geographical distance between customers and NEVs with the time penalty for charging. The logic distance is calculated as

$$ld(c_i, v_j) = \begin{cases} s \cdot wt(c_i, v_j), & \text{if } v_j \text{ satisfies electricity} \\ & \text{constraint of } c_i \\ N \cdot l, & \text{otherwise} \end{cases} \tag{4}$$

Equation (4) indicates that if v_j satisfies the electricity constraint of c_i, the logical distance between them is determined by customer waiting time and vehicle speed. Otherwise, the logical distance is set to a large penalty value.

The NEVD problem for maximizing total customer satisfaction is formulated in a two-hierarchical structure as

$$\text{maximize } f_1(S) = \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} \tag{5}$$

$$\text{minimize } f_2(S) = \sum_{i=1}^{N} \sum_{j=1}^{M} \left(x_{ij} \cdot wt(c_i, v_j) \right) \tag{6}$$

$$\text{subject to } x_{ij} = \begin{cases} 1, & \text{if vehicle } j \text{ is assigned to} \\ & \text{customer } i \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$\sum_{j=1}^{M} x_{ij} \leq 1 \tag{8}$$

$$\sum_{i=1}^{N} x_{ij} \leq 1 \tag{9}$$

In (5), $f_1(S)$ is the number of requests that are satisfied in the solution S. The primary task of NEVD is to maximize the vehicle-passenger matching success rate. Equation (6) calculates the total waiting time of all accepted requests. When evaluating two solutions, we compare their f_1 values first and the one with a larger f_1 value is better. If they have the same f_1 value, then we compare their f_2 values, and the one with a smaller f_2 value is better. In (7), if an NEV is assigned to one request, it must first satisfy the electricity constraint of the request. Constraints (8) and (9) help guarantee that a request (or NEV) is only assigned to one NEV (or request) at most.

IV. EACS Algorithm for Solving the NEVD Problem

ACS is an efficient global stochastic search algorithm inspired by the foraging behavior of ants [28]. In ACS, a set of ants cooperate via pheromone, which records global experience. Moreover, a greedy heuristic is introduced to guide the search. Take the TSP as an example, each ant randomly starts from a city and then determines the next city to visit by considering the pheromone and heuristic information. It constructs a tour by repeating this process until all the cities are visited. When a complete tour has been constructed, a local pheromone updating rule is performed to evaporate the pheromone on the edges of the path, making other ants prefer to explore other edges. After all the ants have constructed their solutions, a global pheromone updating rule is applied to deposit the pheromone on edges of the globally best solution. Through repeated iterations, ACS can gradually approach the optimal solution.

Based on the ACS optimizer, the EACS algorithm is applied to solve the NEVD problem. Its complete procedures are described as follows.

A. Initialization Configurations

As the core issue is to assign NEVs to service requests, it is reasonable to set pheromone between vehicles and customers. The pheromone value $\tau(i, j)$ indicates the preference that NEV j is assigned to customer i according to the historical experience. To initialize the pheromone τ_0, we first construct a dispatch solution π_{FCFS} using the FCFS approach and set τ_0 by

$$\tau_0 = (M \cdot Lnn)^{-1} \tag{10}$$

$$Lnn = \sum_{i=1}^{N}\sum_{j=1}^{M}\left(x_{ij} \cdot ld(c_i, v_j)\right) + (\min(N, M) - f_1(\pi_{FCFS})) \cdot N \cdot l \quad \forall (i, j) \in \pi_{FCFS} \tag{11}$$

where $\min(N, M)$ represents the maximum number of requests that can be satisfied in an ideal situation, i.e., the smaller one between N and M. Herein, the FCFS approach dispatches the NEVs to the requests one by one in a greedy fashion according to the logical distance. This process follows the order in which requests arrive in the time window (although these requests can be processed concurrently, their arrival is still sequential). That is, the first request is assigned with the NEV with smallest logical distance to it, then the second request is assigned with the NEV with smallest logical distance to it among the remaining NEVs, and so on. This way, all the requests are assigned with their NEVs, or some requests can not be satisfied due to no NEVs can meet the constraints. Equation (11) contains the total logical distance of accepted requests and the penalty for refused requests.

B. Solution Construction

After initialization, ants iteratively construct feasible solutions to find an optimal solution (i.e., dispatch scheme). Different from randomly selecting the first city in TSP, in the NEVD problem, the order of requests arriving at the dispatch center in the time window is randomly shuffled before solution construction. Then each ant constructs a potential solution by assigning requests (customers) to NEVs one by one, in the order they are shuffled. Since the solution construction process for each ant is similar, without loss of generality, the following process is described based on one ant. In each step of construction, the ant selects a proper NEV for the corresponding customer. In the kth assignment, a set of candidate NEVs for c_i is defined as

$$J_i = \{j | \sum_{i=1}^{N} x_{ij} = 0 \text{ and } [D(e_j) \geq d(c_i, v_j) + d(c_i, t_i) \text{ or } (E(t_i) = 1 \text{ and } \frac{d(c_i, v_j) + d(c_i, t_i) - D(e_j)}{k(cr)} \leq tl)], 1 \leq j \leq M\} \tag{12}$$

whose element j represents the NEV that has not been assigned before (i.e., $\sum_{i=1}^{N} x_{ij} = 0$) and satisfies the electricity constraint of c_i. Then the ant selects an NEV j for c_i from J_i according to a state transition rule.

Pheromone and heuristic information are two key components in the state transition rule, where the pheromone has been defined above. Heuristic information is introduced to guide the search of ants according to the logical distance between customers and NEVs. Referring to the application of ACS on TSP [28], heuristic information in the state transition rule is designed as

$$\eta(i, j) = \frac{1}{ld(c_i, v_j)} \tag{13}$$

where $\eta(i, j)$ is the heuristic information between customer i and NEV j. The NEV with the shorter logical distance from the customer is given greater priority to be chosen. Then, the probability that NEV j is assigned to customer i is calculated by

$$p(i, j) = \begin{cases} \dfrac{[\tau(i, j)] \cdot [\eta(i, j)]^{\beta}}{\sum_{u \in J_i} [\tau(i, u)] \cdot [\eta(i, u)]^{\beta}}, & \text{if } j \in J_i \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where $\tau(i, j)$ is the pheromone value between customer i and NEV j, and β ($\beta > 0$) is a parameter that controls the relative importance of pheromone versus heuristic information [28].

In the ACS algorithm, the state transition rule is as follows: For customer i, the ant chooses NEV j from the set J_i according to the rule given by

$$j = \begin{cases} \arg\max_{u \in J_i}\{[\tau(i, u)] \cdot [\eta(i, u)]^{\beta}\}, & \text{if } q \leq q_0 \\ J, & \text{otherwise} \end{cases} \tag{15}$$

where q is a random number uniformly distributed in [0,1], J is a random variable selected from J_i according to the probability distribution in (14), and q_0 ($0 \leq q_0 \leq 1$) is a parameter that controls the exploitation and exploration behaviors of the ant. If $q \leq q_0$, then the ant chooses the NEV whose pheromone and heuristic information are maximal, measured by $\tau(i, u) \cdot \eta(i, u)^{\beta}$, for exploitation. Otherwise, the NEV is determined as a random number J which is selected according to the probability in (14), for better exploration.

C. Pheromone Updating Rule

In ACS, pheromone records the historical knowledge accumulated by ants. During the calculation process, a local pheromone updating rule and a global pheromone updating rule are performed to change the amount of pheromone. When an ant has constructed a feasible solution, the local pheromone updating rule is applied on each customer-NEV pair of the solution as

$$\tau(i, j) = (1 - \rho) \cdot \tau(i, j) + \rho \cdot \tau_0 \tag{16}$$

where ρ ($0 < \rho < 1$) is the pheromone decay parameter.

Conversely, only the historically best solution (optimal solution so far) π_{Best} is allowed to perform the global pheromone updating rule. After all the ants have constructed their solutions, the pheromone on each customer-NEV pair of π_{Best} is

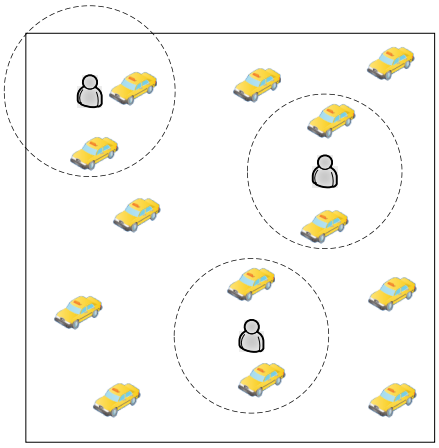


Fig. 3. Pre-selection strategy.

increased as

$$\tau(i,j) = (1-\varepsilon)\cdot\tau(i,j) + \varepsilon\cdot\Delta\tau \tag{17}$$

$$\Delta\tau = \left(\sum_{i=1}^{N}\sum_{j=1}^{M}\left(x_{ij}\cdot ld(c_i,v_j)\right) + (\min(N,M) - f_1(\pi_{Best}))\cdot l\right)^{-1} \quad \forall(i,j)\in\pi_{Best} \tag{18}$$

where ε is the pheromone enhancement parameter. Equation (18) ensures that customer-NEV pairs of a solution with more accepted requests and shorter waiting time can deposit more pheromone.

These two pheromone updating rules guide search behavior of ants by adjusting the impact of environment on them. The local pheromone updating rule allows other ants have a greater chance to explore new customer-NEV pairs by reducing the pheromone on the assigned pairs. Therefore, it helps to increase the population diversity. The global pheromone updating rule enhances the desirability of ants for customer-NEV pairs of the historically best solution. It is able to guide the search in a more promising direction and accelerate the convergence of the population.

D. Pre-Selection Strategy

When carrying out experiments, we found that with a certain number of customers, the running time of ACS generally increases proportionally with the number of NEVs, showing poor scalability. While in real-world scenarios, it is common that the number of NEVs is usually multiple of the concurrent requests. In order to reduce the running time, a pre-selection strategy is proposed to perform on initial NEVs before an ACS process so as to decrease the number of NEVs to be assigned. As some NEVs are filtered out, it is critical to effectively select the NEVs to reduce the loss of accuracy.

Fig. 3 shows an example of the pre-selection strategy for initial NEVs. For the convenience of illustration, only NEVs and requests (customers) in a given time window are shown and charging piles are ignored. An NEV closer to one customer in this figure means that the NEV has a shorter logical distance to the customer. The calculation of logical distance between customers and NEVs has been described above. Before an ACS process, some NEVs "near" these customers are selected. The process follows the order in which the requests arrive. For the first customer (request), we select K NEVs with the shortest logical distances from it. Then other K NEVs with the shortest distances from the second customer are selected by the second customer among the remaining M-K NEVs, and so on. By this means, $K \times N$ NEVs that have more power and are geographically closer with these customers are selected. In the subsequent search process of ants, only these pre-selected NEVs are assigned to customers. Note that the pre-selection strategy is only applied to scenarios where M is larger than $K \times N$. Therefore, if the pre-selection strategy is performed, the M in (12) and (18) is replaced by $K \times N$.

E. Local Pruning Strategy

The actual passenger-NEV matching is an instant service, which needs to be completed in a short time. In fact, it is not necessary to consider all of the available NEVs when dispatching for a customer because too long away NEVs (e.g., with very large logical distance) are unlikely to be suitable for this customer. Therefore, a local pruning is needed to reduce the number of considered NEVs, which can also significantly reduce the computation time. In the taxi dispatch service [20], it has been empirically confirmed that an efficient way is to assign a taxi in the vicinity of the customer location, so as to reduce response time. Therefore, this heuristic experience is integrated into the ACS process in this paper.

In order to further improve the efficiency of ACS, a local pruning strategy is proposed to perform during the solution construction process. It is assumed that the NEV assigned to c_i should be found among a limited number of T (e.g., 10) NEVs that are nearest from c_i in terms of logical distance [20]. Therefore, when an ant selects an NEV for c_i, only the T NEVs with the shortest logical distances to the customer location are considered, denoted as the considered NEVs set T_i for the customer c_i.

The pre-selection and local pruning strategies efficiently reduce computation time by decreasing the number of NEVs to be assigned. With these two strategies, the ACS is named EACS in this paper, and the set of candidate NEVs defined in (12) is adjusted as

$$J_i = \{j \mid \sum_{i=1}^{N} x_{ij} = 0 \text{ and } [D(e_j) \geq d(c_i,v_j) + d(c_i,t_i) \text{ or } (E(t_i) = 1 \text{ and } \frac{d(c_i,v_j) + d(c_i,t_i) - D(e_j)}{k(cr)} \leq tl)] \text{ and } j \in T_i, 1 \leq j \leq K\cdot N\} \tag{19}$$

F. Complete EACS Algorithm

The complete EACS algorithm is shown in Fig. 4 and is described in the following seven steps.

Step 1: Calculate the logical distance between customers and NEVs.

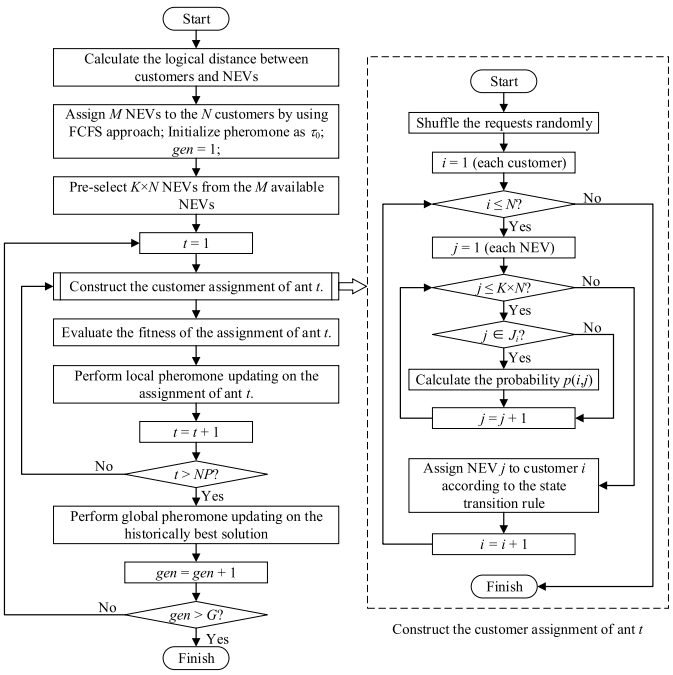


Fig. 4. Flowchart of the EACS algorithm.

Step 2: Assign NEVs to customers by the FCFS approach and set the initial pheromone τ_0 according to (10) and (11). Set the generation $gen = 1$.

Step 3: Pre-select $K \times N$ NEVs from the initial M NEVs.

Step 4: Let each ant construct a feasible solution and perform the local pheromone updating rule on each assigned customer-NEV pair as (16).

Step 5: Evaluate the fitness of each ant and record the best solution.

Step 6: Find out the historically best solution and perform the global pheromone updating rule as (17).

Step 7: Termination check. When the maximal generations G is reached, the algorithm terminates. Otherwise, move to step 4 and continue optimizing.

The solution construction process in step 4 is described in detail in the sub-flowchart of Fig. 4.

V. Experiments and Comparisons

A. Parameter Configurations and Test Cases

Experimental tests are conducted in this section to verify the performance of EACS. All the algorithms are implemented in C++ and run on a PC with a Xeon quad-core CPU E3 and 8.0GB RAM.

Two test scenarios are designed in different physical areas. One is resource-shortage scenario, which means the number of NEVs is less than that of customers. The other is resource-rich scenario, that is, NEVs are more than customers. We have published the datasets used in the experiments on GitHub (https://github.com/liangdii/NEVD). As EACS is different from ACS in the pre-selection strategy and local pruning strategy to reduce the running time on resource-rich scenario (i.e., M is much larger than N), the EACS is only applied in the resource-rich scenario and is to compare with FCFS, greedy algorithm (denoted as Greedy), multi-agent assignment algorithm with collaborative local mediation(MA^3-LM) [20], GA, and ACS. In the resource-shortage scenario (i.e., M is smaller than N), there is no need to perform the pre-selection strategy and local pruning strategy. Therefore, the EACS is exactly the ACS, and is compared with FCFS, Greedy, MA^3-LM, and GA.

Algorithm 1 Greedy Algorithm for the NEVD Problem

Begin

1. **For** i =1 to I //I is the number of shuffle times
2. Shuffle the order of requests;
3. Use FCFS to assign the NEVs based on the above ordered requests;
4. Record the fitness of the solution by (5) and (6);
5. **End of For**
6. Output the best dispatch solution;

End

The FCFS has been described in Section IV-A. Different from FCFS, the greedy algorithm randomly shuffles the order of requests before assignment and uses FCFS to assign the NEVs based on these ordered requests. The order is randomly shuffled for a number of times and the best solution among these tries is output as the final solution. The complete procedure of the greedy algorithm is shown in **Algorithm 1**. MA^3-LM is a well-perform state-of-the-art approach proposed to solve taxi-passenger dispatch problem [20]. In this paper, MA^3-LM is extended to solve the NEVD problem. Moreover, the cases in [20] are with the same number of customers and vehicles, while herein the number of vehicles is different from customers.

Algorithm 2 Genetic Algorithm for the NEVD Problem

Begin

1. $t \leftarrow 0$;
2. Initialize a population $P(t)$;
3. Evaluate fitness of $P(t)$;
4. Keep the best individual in $P(t)$;
5. **While** (termination criterion not met) **do**
6. $P(t) \leftarrow$selection operation to $P(t)$;
7. $P(t) \leftarrow$crossover operation to $P(t)$;
8. Resolve coding conflicts;
9. Mutation: randomly exchange two gene values of an individual;
10. $t \leftarrow t + 1$;
11. $P(t) \leftarrow P(t-1)$;
12. Evaluate fitness of $P(t)$;
13. Update the best individual;
14. **End of While**

End

As there are no evolutionary-based approaches to solve the NEVD problem in the literature, we also design a GA-based approach in this paper for comparisons. The complete procedure of GA for solving the NEVD problem is shown in **Algorithm 2**. Herein, each individual is encoded with the

TABLE II
PARAMETER CONFIGURATIONS FOR THE EACS ALGORITHM

ACS parameters						NEVD parameters				Other parameters	
NP	*G*	q_0	ρ	ε	β	*ep*	*tl*	*s*	*k*(*cr*)	*K*	*T*
20	150	0.9	0.1	0.1	2.0	0.6	10 min	50 km/h	3 km/min	2	10

TABLE III
EXPERIMENTAL RESULT COMPARISONS IN TEST I WITH RESOURCE-SHORTAGE SCENARIO

No.	*N*	*M*	FCFS		Greedy			MA^3-LM			GA			ACS	
			RAR	AWT	RAR	AWT		RAR	AWT		RAR	AWT		RAR	AWT
A1	10	8	70.0%	**9.10**	**80.0%**	9.87	(≈)	70.0%	9.21	(+)	**80.0%**	9.72	(–)	**80.0%**	9.83
A2	10	10	80.0%	**4.38**	**100.0%**	6.80	(+)	90.0%	4.85	(+)	**100.0%**	6.64	(≈)	**100.0%**	6.65
A3	20	16	65.0%	4.69	**80.0%**	5.27	(+)	75.0%	**4.07**	(+)	**80.0%**	4.87	(≈)	**80.0%**	4.89
A4	20	20	80.0%	**3.10**	**95.0%**	5.16	(+)	**95.0%**	4.66	(+)	**95.0%**	4.64	(≈)	**95.0%**	4.64
A5	50	40	68.0%	4.64	78.0%	5.54	(+)	78.0%	**3.85**	(+)	78.0%	4.30	(+)	**80.0%**	5.09
A6	50	50	80.0%	**3.49**	90.0%	4.03	(+)	96.0%	3.62	(+)	98.0%	4.44	(+)	**100.0%**	5.12
Number of +/≈/–					5/1/0			6/0/0			2/3/1			\	

number of customers N as its length, each gene stands for a customer. The value of each gene is generated randomly in [1, M], meaning the index of NEV that assigned to this customer. The fitness function (line 3) is set to the inverse of the total logical distance of all matched customer-NEV pairs like (10) and (11). If there are conflicts between customer-NEV pairs after crossover operation (line 8), we reassign an NEV with the shortest logical distance from the conflicting customer among all unassigned NEVs.

The parameter configures for EACS are shown in Table II. The ACS-related parameters are the population size $NP = 20$, maximal generation number $G = 150$, $q_0 = 0.9$, $\rho = 0.1$, $\varepsilon = 0.1$, and $\beta = 2.0$. Through the investigation of BYD (a famous high-tech company for NEVs in China), most electric vehicles on the market have a driving range between 150 km and 400 km, and can be quickly charged 80% in 0.5 to 2 hours [36]. Therefore, the charging rate $k(cr)$ is set to 3 km/min in this paper. In order to determine whether there is at least a charging pile on the way to the destination t_i, a probability parameter $ep = 0.6$ is defined here. For each request, if a random number uniformly distributed in [0, 1] is smaller than ep, then NEVs are allowed to charge on the way to the destination t_i. The other NEVD-related parameters are set as $tl = 10$ min and $s = 50$ km/h (i.e., 0.833 km/min). The other EACS-related parameters $K = 2$ and $T = 10$. The population size of GA is 20 and the maximal iterations of GA, MA^3-LM, and Greedy are 1000, 3000, and 3000, respectively. Moreover, distances between customers and their destinations $d(c_i, t_i)$ and kilometers that can be supported by the remaining electricity of NEVs $D(e_j)$ are randomly generated following uniform distribution in [10 km, 80 km] and [5 km, 100 km]. All distance values are accurate to one decimal place during the calculation in all test cases.

As Greedy, MA^3-LM, GA, ACS, and EACS are random algorithms (i.e., except FCFS), they perform 30 independent runs on each instance for fair comparison. Referring to the two-hierarchical objective defined in (5) and (6), we compare the requests acceptance rate (i.e., RAR, which can be calculated by f_1/N) and the average waiting time (i.e., AWT, which can be calculated by f_2/f_1) herein. For those random algorithms run 30 times, the mean values of RAR and AWT are compared. The best results are highlighted in **boldface**. To further validate the performance of the proposed ACS-based algorithms (i.e., EACS and ACS) on the NEVD problem, a statistical test called Wilcoxon's signed rank test is conducted between the ACS-based algorithms and other stochastic algorithms at the 5% significance level. In order to be consistent with the two-hierarchical objective evaluation rule, we first conduct the significance test on RAR of 30 runs. If there is no statistically significant difference on RAR, then AWT of 30 runs is tested. We mark the cases with "+", "≈", and "−" to indicate that EACS/ACS performs significantly better than, similarly to, and significantly worse than the compared algorithm, respectively. The symbol "\" in tables represents the tested algorithm.

B. Test I: Small Physical Area

A number of service requests are generated following uniform distribution in a physical area of 10 km × 10 km. In order to simulate different resource configurations, the number of NEVs is set to 0.8-10 times that of requests. The results of resource-shortage scenario and resource-rich scenario are given in Table III and Table IV, respectively. The RAR column and AWT column represent the requests acceptance rate and average waiting time (in minutes), respectively.

As shown in Table III, ACS and GA perform better while FCFS performs the worst compared with the other four algorithms because FCFS only allocates NEVs in a fixed order of requests. FCFS and MA^3-LM can obtain shorter customer waiting time, while they perform worse than the other three algorithms in terms of the number of accepted requests. This may be due to that FCFS simply dispatch the nearest vehicle to a customer without considering the influence on the subsequent customers. Therefore, the RAR may be low because many other customers can not be satisfied, while the

TABLE IV
EXPERIMENTAL RESULT COMPARISONS IN TEST I WITH RESOURCE-RICH SCENARIO

No.	N	M	FCFS		Greedy			MA^3-LM			GA			ACS			EACS	
			RAR	AWT	RAR	AWT		RAR	AWT		RAR	AWT		RAR	AWT		RAR	AWT
B1	20	40	**100.0%**	2.52	**100.0%**	**2.06**	(+)	80.0%	4.36	(+)	**100.0%**	2.11	(+)	**100.0%**	**2.06**	(≈)	**100.0%**	**2.06**
B2	20	60	**100.0%**	1.89	**100.0%**	1.54	(+)	95.0%	4.73	(+)	**100.0%**	1.58	(+)	**100.0%**	**1.53**	(≈)	**100.0%**	**1.53**
B3	20	100	**100.0%**	0.99	**100.0%**	**0.97**	(≈)	**100.0%**	4.30	(+)	**100.0%**	**0.97**	(≈)	**100.0%**	**0.97**	(≈)	**100.0%**	**0.97**
B4	20	160	**100.0%**	**0.57**	**100.0%**	**0.57**	(≈)	85.0%	3.52	(+)	**100.0%**	0.58	(≈)	**100.0%**	**0.57**	(≈)	**100.0%**	**0.57**
B5	20	200	**100.0%**	0.57	**100.0%**	**0.56**	(+)	95.0%	4.63	(+)	**100.0%**	0.57	(≈)	**100.0%**	**0.56**	(≈)	**100.0%**	**0.56**
B6	50	100	**100.0%**	1.12	**100.0%**	0.99	(+)	98.0%	3.11	(+)	**100.0%**	1.06	(+)	**100.0%**	**0.98**	(≈)	**100.0%**	**0.98**
B7	50	150	**100.0%**	0.87	**100.0%**	**0.78**	(+)	90.0%	3.85	(+)	**100.0%**	0.85	(+)	**100.0%**	**0.78**	(≈)	**100.0%**	**0.78**
B8	50	250	**100.0%**	0.48	**100.0%**	**0.46**	(–)	98.0%	3.36	(+)	**100.0%**	0.47	(+)	**100.0%**	**0.46**	(≈)	**100.0%**	**0.46**
B9	50	400	**100.0%**	0.41	**100.0%**	**0.38**	(≈)	98.0%	3.21	(+)	**100.0%**	0.40	(+)	**100.0%**	**0.38**	(≈)	**100.0%**	**0.38**
B10	50	500	**100.0%**	**0.33**	**100.0%**	**0.33**	(≈)	**100.0%**	2.72	(+)	**100.0%**	0.34	(+)	**100.0%**	**0.33**	(≈)	**100.0%**	**0.33**
Number of +/≈/–					5/4/1			10/0/0			7/3/0			0/10/0			\	

TABLE V
EXPERIMENTAL RESULT COMPARISONS IN TEST II WITH RESOURCE-SHORTAGE SCENARIO

No.	N	M	FCFS		Greedy			MA^3-LM			GA			ACS			
			RAR	AWT	RAR	AWT		RAR	AWT		RAR	AWT		RAR	AWT		
C1	50	40	62.0%	26.26	**66.0%**	21.12	(+)	64.0%	19.52	(+)	64.0%	**18.65**	(+)	**66.0%**	20.31		
C2	50	50	74.0%	16.19	84.0%	21.21	(+)	82.0%	**16.08**	(+)	82.0%	19.47	(+)	**90.0%**	24.45		
C3	100	80	71.0%	16.21	77.0%	17.64	(+)	75.0%	**13.29**	(+)	72.0%	19.93	(+)	**78.0%**	16.09		
C4	100	100	82.0%	14.92	87.0%	17.06	(+)	88.0%	**14.29**	(+)	83.0%	20.49	(+)	**94.0%**	18.98		
C5	150	120	66.7%	15.63	71.3%	15.92	(+)	70.0%	**11.88**	(+)	64.7%	21.27	(+)	**77.3%**	17.69		
C6	150	150	85.3%	12.76	88.7%	13.93	(+)	90.7%	**10.93**	(+)	83.3%	20.08	(+)	**97.3%**	17.33		
C7	200	160	67.5%	11.84	73.0%	13.03	(+)	72.5%	**9.65**	(+)	65.0%	19.61	(+)	**79.0%**	14.81		
C8	200	200	78.5%	11.55	84.5%	12.80	(+)	87.0%	**9.62**	(+)	78.0%	19.82	(+)	**94.5%**	16.17		
Number of +/≈/–					8/0/0			8/0/0			8/0/0			\			

AWT for those accepted customers can be shorter. However, too low acceptance rate means the FCFS and MA^3-LM may not be suitable for the resource-shortage NEVD problem. Also, the RAR of Greedy and GA become worse than that of ACS when the number of customers and vehicles increases. Therefore, ACS has the best global search ability to obtain the best dispatch results than the other compared approaches when considering both the RAR and AWT metrics.

For the resource-rich scenario, the results in Table IV further show that both EACS and ACS have the best performance. MA^3-LM is the worst approach on both the RAR and AWT, indicating that it is only suitable for cases where the number of vehicles is equal to that of requests as suggested in [20]. As the number of NEVs increases, the performance of FCFS and Greedy gradually improves because the influence of the assigned NEVs on subsequent dispatch becomes less significant. Therefore, the Greedy and FCFS may have good performance on resource-rich scenario, but perform significantly poorly on resource-shortage scenario. In general, only the ACS and EACS obtain the best results on both the RAR and the AWT metrics on all the 10 cases in resource-rich scenario, indicating that the ACS-based algorithms have the best ability in optimizing the NEVD problem.

C. Test II: Large Physical Area

Similar to Test I, Test II is made in a large physical area of 100 km × 100 km with resource-shortage scenario and resource-rich scenario. The dispatch results are given in Table V and Table VI, respectively.

From Table V and Table VI, we can see that MA^3-LM ranks third in resource-shortage scenario while performs the worst in resource-rich scenario. GA obtains poor results when the number of customers and NEVs is large. This may be due to that the increase of coding length makes GA converge more difficult. Similar to Test I, Greedy and FCFS perform gradually worse as the number of NEVs decreases when the number of customers is fixed. ACS obtains the maximum matching success rate in resource-shortage scenario. In general, EACS performs slightly better than ACS in resource-rich scenario. This indicates filtering out some unimportant NEVs before dispatch can help enhance the solution quality of EACS.

D. Efficiency Validation of EACS

To validate the efficiency of the EACS, it is also compared with IBM ILOG CPLEX Optimization Studio 12.8.0, which is a commercial linear program solver. We compare the RAR, AWT, and running time on some respective cases in Test I and II. Note that for the resource-shortage cases, the EACS algorithm is actually the ACS algorithm. The results listed in Table VII show that EACS can use much shorter time to obtain dispatch results similar to those of CPLEX. In the resource-shortage cases, it performs similarly to CPLEX. In the resource-rich cases, it can obtain the same optimal solutions as CPLEX. On the computation time, it runs much faster than CPLEX. Moreover, two large-scale cases with 500 requests in the large physical area are also tested. The EACS can obtain the optimal solutions in about 15 seconds. However, CPLEX spends more than 3 minutes with 3000 NEVs and

TABLE VI
EXPERIMENTAL RESULT COMPARISONS IN TEST II WITH RESOURCE-RICH SCENARIO

No.	N	M	FCFS		Greedy			MA^3-LM			GA			ACS			EACS	
			RAR	AWT	RAR	AWT		RAR	AWT		RAR	AWT		RAR	AWT		RAR	AWT
D1	50	100	98.0%	10.98	**100.0%**	10.20	(+)	82.0%	15.91	(+)	**100.0%**	11.09	(+)	**100.0%**	9.96	(+)	**100.0%**	**9.94**
D2	50	150	98.0%	**6.73**	**100.0%**	6.96	(≈)	86.0%	15.49	(+)	**100.0%**	7.43	(+)	**100.0%**	6.97	(≈)	**100.0%**	6.96
D3	50	250	**100.0%**	5.28	**100.0%**	4.91	(–)	86.0%	17.65	(+)	**100.0%**	5.56	(+)	**100.0%**	**4.90**	(–)	**100.0%**	4.92
D4	50	400	**100.0%**	4.63	**100.0%**	**4.47**	(≈)	86.0%	16.00	(+)	**100.0%**	4.95	(+)	**100.0%**	**4.47**	(≈)	**100.0%**	**4.47**
D5	50	500	**100.0%**	3.70	**100.0%**	**3.59**	(≈)	82.0%	16.19	(+)	**100.0%**	3.94	(+)	**100.0%**	**3.59**	(≈)	**100.0%**	**3.59**
D6	100	200	96.0%	7.07	**99.0%**	6.85	(+)	85.0%	12.18	(+)	98.0%	7.88	(+)	**99.0%**	6.61	(≈)	**99.0%**	**6.60**
D7	100	300	99.0%	7.19	**100.0%**	6.35	(+)	87.0%	13.39	(+)	**100.0%**	7.82	(+)	**100.0%**	6.26	(≈)	**100.0%**	**6.24**
D8	100	500	**100.0%**	4.67	**100.0%**	4.49	(+)	82.0%	11.74	(+)	**100.0%**	5.77	(+)	**100.0%**	**4.46**	(+)	**100.0%**	**4.46**
D9	100	800	**100.0%**	3.30	**100.0%**	3.19	(+)	85.0%	12.37	(+)	**100.0%**	4.45	(+)	**100.0%**	**3.18**	(+)	**100.0%**	**3.18**
D10	100	1000	**100.0%**	2.73	**100.0%**	**2.66**	(≈)	89.0%	12.92	(+)	**100.0%**	4.09	(+)	**100.0%**	**2.66**	(≈)	**100.0%**	**2.66**
D11	200	400	99.0%	5.01	**100.0%**	4.82	(+)	92.5%	9.40	(+)	99.5%	6.81	(+)	**100.0%**	4.70	(≈)	**100.0%**	**4.67**
D12	200	600	**100.0%**	4.56	**100.0%**	4.43	(+)	86.5%	10.00	(+)	99.0%	6.77	(+)	**100.0%**	4.32	(+)	**100.0%**	**4.24**
D13	200	1000	**100.0%**	3.11	**100.0%**	3.04	(+)	88.5%	9.82	(+)	99.0%	5.60	(+)	**100.0%**	3.01	(+)	**100.0%**	**2.99**
D14	200	1600	**100.0%**	2.34	**100.0%**	2.30	(+)	87.0%	9.46	(+)	99.0%	5.56	(+)	**100.0%**	2.29	(+)	**100.0%**	**2.28**
D15	200	2000	**100.0%**	2.13	**100.0%**	**2.09**	(+)	86.5%	10.39	(+)	98.5%	4.96	(+)	**100.0%**	2.10	(+)	**100.0%**	**2.09**
Number of +/≈/–					10/4/1			15/0/0			15/0/0			7/7/1			\	

TABLE VII
COMPARISON RESULTS BETWEEN EACS AND CPLEX

Case Type	No.	CPLEX			EACS		
		RAR	AWT	Time(s)	ARA	AWT	Time(s)
resource-shortage cases	A3	**80%**	**4.7**	0.13	**80%**	4.89	**0.0202**
	A4	**100%**	6.01	0.16	95%	**4.64**	**0.0261**
	C3	**78%**	**12.63**	0.64	**78%**	16.09	**0.3298**
	C4	**100%**	**16.75**	1.04	94%	18.98	**0.4407**
resource-rich cases	B6	**100%**	**0.98**	0.57	**100%**	**0.98**	**0.1918**
	B7	**100%**	**0.78**	0.70	**100%**	**0.78**	**0.1909**
	D14	**100%**	**2.28**	23.42	**100%**	**2.28**	**2.4635**
	D15	**100%**	**2.09**	30.77	**100%**	**2.09**	**2.4584**
	(500,3000)	**100%**	**1.62**	229.60	**100%**	**1.62**	**15.3700**
	(500,5000)	NA	NA	>1800	**100%**	**1.21**	**15.5270**

NA means that CPLEX cannot find a feasible solution within 30 minutes.

cannot find a feasible solution in 30 minutes with 5000 NEVs, which is unbearable for this time-sensitive application.

Therefore, EACS is a reasonable and efficient approach for solving the NEVD problem. It can provide promising dispatch results in various scenarios and has good scalability. To further reflect the efficiency of EACS, Table VIII lists the running time of EACS for all instances in Test I and Test II.

E. Analysis of ACS Parameters

The ACS parameters include the population size NP, the maximal generations G, q_0, β, ρ, and ε. In this section, we take C6, C7, D11, and D12 as examples to study the influences of these parameters on the performance of ACS and EACS on the NEVD problem in both the resource-shortage scenario and resource-rich scenario. Note that when one parameter is investigated, the others remain the same as in Table II.

The investigation begins with the parameters NP and G. We set NP from 5 to 50 with a step length of 5 and G from 50 to 500 with a step length of 50. Fig. 5 and Fig. 6 show the influences of the parameters on the number of accepted requests and the average customer waiting time. It can be seen that the solution quality of D11 and D12 is not very

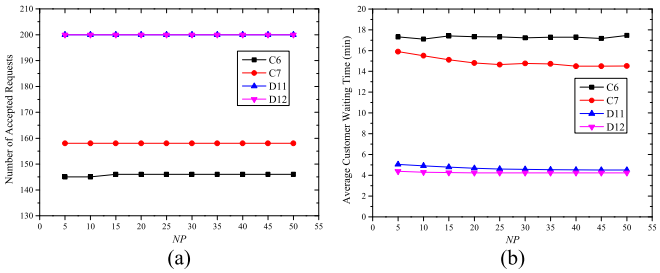


Fig. 5. Influence of the population size (NP) on C6, C7, D11, and D12. (a) Number of accepted requests. (b) Average customer waiting time (in minutes).

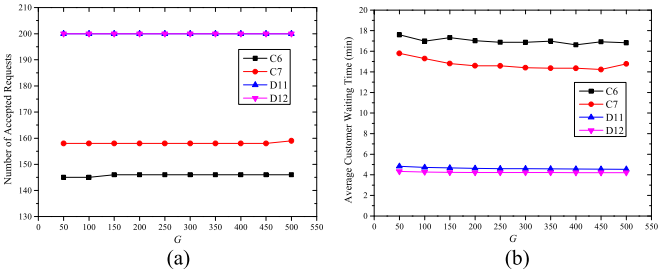


Fig. 6. Influence of the maximal generations (G) on C6, C7, D11, and D12. (a) Number of accepted requests. (b) Average customer waiting time (in minutes).

sensitive to the parameters and all the requests are satisfied. As anticipated, the solution quality of C6 and C7 improves as NP or G increases. However, larger population size or generation number can lead to delays in service response time. Since the improvement in quality is not very significant when $NP \geq 20$ or $G \geq 150$, especially in resource-rich scenario, this paper sets NP to 20 and G to 150, so as to make a tradeoff between the solution quality and the computation time.

The next parameter tested is q_0. We set q_0 from 0 to 1.0 with a step length of 0.1. The number of accepted requests and the average customer waiting time are plotted in Fig. 7. The tendency of the curves indicates that it is better to adopt a larger q_0 for better performance. More requests are satisfied when $q_0 \geq 0.8$ and the average customer waiting

TABLE VIII
COMPUTATION TIME OF EACS FOR ALL INSTANCES

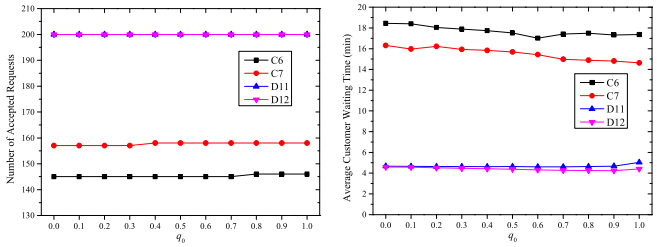
No.	A1	A2	A3	A4	A5	A6	B1	B2	B3	B4	B5	B6	B7
Time(s)	0.0058	0.0080	0.0202	0.0261	0.1054	0.1374	0.0382	0.0406	0.0412	0.0412	0.0413	0.1918	0.1909
No.	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	D1	D2
Time(s)	0.1943	0.1964	0.1943	0.0915	0.1231	0.3298	0.4407	0.6640	1.0214	1.1588	1.5609	0.2026	0.1923
No.	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Time(s)	0.1935	0.1968	0.2115	0.6979	0.7226	0.7310	0.7385	0.7123	2.4033	2.4264	2.4461	2.4635	2.4584

TABLE IX
EXPERIMENTAL RESULT COMPARISONS OF EACS, EACS-noPS, EACS-noLP, AND ACS FOR PROBLEM D11-D15

No.	EACS			EACS-noPS			EACS-noLP			ACS		
	AWT	CPU Time(s)	FEs	AWT	CPU Time(s)	FEs	AWT	CPU Time(s)	FEs	AWT	CPU Time(s)	FEs
D11	**4.67**	2.2362	2790.2	**4.67**	**2.2124**	2760.1	4.70	3.1641	**2756.5**	4.70	3.2450	2841.5
D12	**4.24**	**2.2407**	2768.7	4.25	3.1247	**2742.7**	4.31	3.4032	2773.6	4.32	4.4113	2759.6
D13	**2.99**	**1.8003**	**2200.9**	**2.99**	4.1150	2328.0	3.00	3.0795	2401.7	3.01	6.9499	2600.6
D14	**2.28**	**1.8084**	**2192.7**	**2.28**	6.6516	2445.5	**2.28**	2.8744	2254.5	2.29	10.8585	2539.0
D15	**2.09**	**1.6856**	**2044.3**	**2.09**	7.3315	2224.0	**2.09**	2.9112	2222.5	2.10	13.1194	2473.2

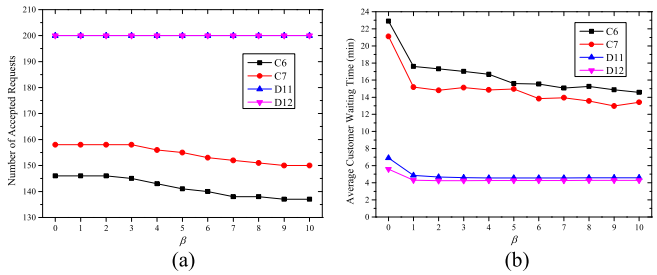
TABLE X
GEOGRAPHICAL DISTANCE COMPARISONS FOR PROBLEM D12-D15

No.	EACS	ACS	GA	MA^3-LM	Greedy	FCFS
D12	**682.97**	692.30	1057.29	1215.93	709.57	726.90
D13	**477.98**	480.18	872.40	1245.37	484.96	496.50
D14	**371.01**	372.57	866.48	1176.93	373.59	377.40
D15	**336.30**	337.29	779.61	1285.90	337.61	341.20



(a) (b)

Fig. 7. Influence of the parameter q_0 on C6, C7, D11, and D12. (a) Number of accepted requests. (b) Average customer waiting time (in minutes).

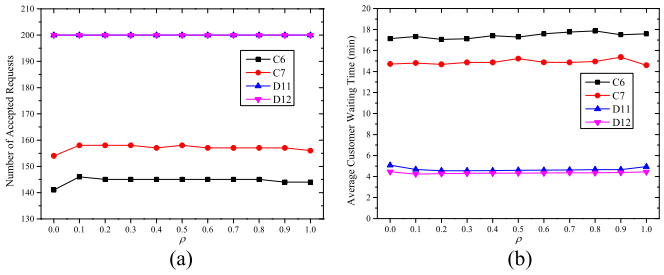


(a) (b)

Fig. 8. Influence of the parameter β on C6, C7, D11, and D12. (a) Number of accepted requests. (b) Average customer waiting time (in minutes).

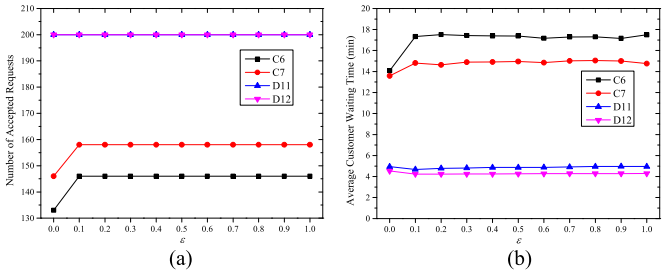
time generally decreases with the increase of q_0 from 0 to 0.9. Therefore, q_0 is set to 0.9 in this paper.

Then parameter β is investigated. As shown in Fig. 8, although the customer waiting time tends to decrease with the



(a) (b)

Fig. 9. Influence of the parameter ρ on C6, C7, D11, and D12. (a) Number of accepted requests. (b) Average customer waiting time (in minutes).



(a) (b)

Fig. 10. Influence of the parameter ε on C6, C7, D11, and D12. (a) Number of accepted requests. (b) Average customer waiting time (in minutes).

increase of β, fewer requests are accepted with a larger β in both C6 and C7. Moreover, the large value of customer waiting time when β is 0 indicates that the heuristic information plays an important role in the search process. Therefore, β is set to 2.0 in this paper.

Finally, parameter ρ in local updating rule and ε in global updating rule are investigated. The results are plotted in Fig. 9 and Fig. 10. The fewest requests are accepted when ρ or ε is set to 0, which indicates the importance of the pheromone updating. As shown in Fig. 9, a relatively small ρ seems to be better in improving the solution quality. This may be due to that a smaller ρ helps to slow down the pheromone

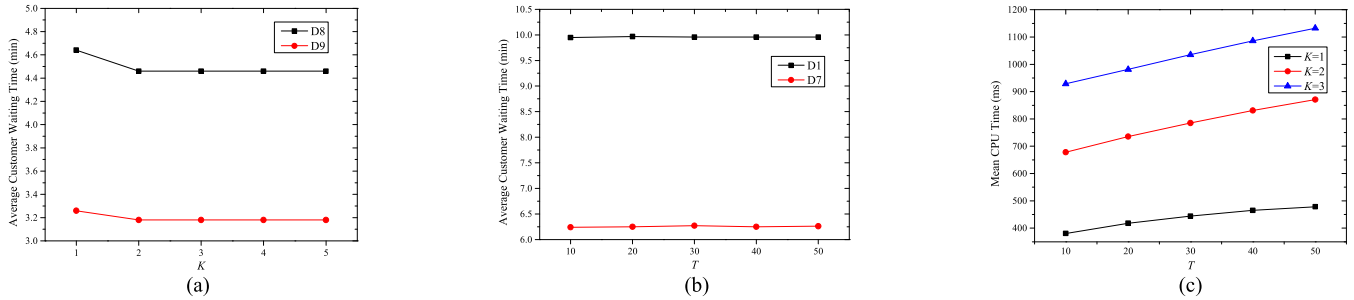


Fig. 11. Influence of the parameters K and T on EACS. (a) Average customer waiting time with different K values on D8 and D9. (b) Average customer waiting time with different T values on D1 and D7. (c) Mean CPU time (in milliseconds) on D8.

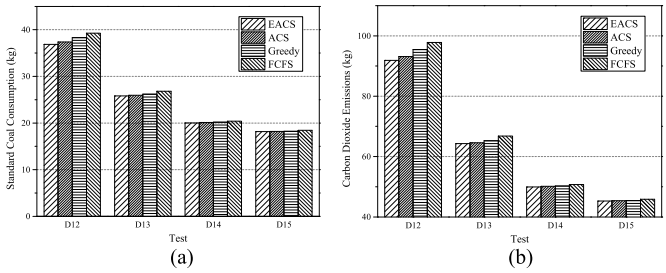


Fig. 12. Power consumption and emissions of different algorithms under different test cases. (a) Standard coal consumption. (b) Carbon dioxide emissions.

evaporation on the matched customer-NEV pairs and to use the historical knowledge about the problem. Fig. 10 shows that the solution quality is not very sensitive to ε when it is set from 0.1 to 1.0.

F. Analysis of Other EACS-Related Parameters

In this section, the impact of parameter K and T on the performance of EACS is investigated. We set parameter K from 1 to 5 with a step length of 1 and T from 10 to 50 with a step length of 10. For each case tested, 30 independent runs are carried out and the mean dispatch results are plotted in Fig. 11(a) and (b). As all the requests are accepted in these cases, only the average customer waiting time is shown. As shown in Fig. 11(a) and (b), the dispatch results are nearly invariant with different K and T, except when $K = 1$. In addition, the runtime (in milliseconds) on D8 with different K and T is investigated. Fig. 11(c) confirms that the mean CPU time increases as K or T increases.

G. Effectiveness of Pre-Selection and Local Pruning Strategies

In order to investigate the effectiveness of two strategies in the proposed EACS, a further comparison is conducted between EACS and its variants on test D11-D15. The variants without pre-selection or local pruning are termed as EACS-noPS and EACS-noLP, respectively. The situations are similar in other resource-rich cases.

The dispatch results obtained by EACS, EACS-noPS, EACS-noLP, and ACS are listed in Table IX. As all the requests are accepted in these cases, the requests acceptance rate (i.e., RAR) is not listed. The results show that EACS gains the best performance, followed by EACS-noPS and EACS-noLP, whereas ACS performs the worst. In addition, a further comparison is carried out in terms of running time and maximal function evaluations (FEs) when the solution quality cannot be improved any more. The results show that the running time of EACS and EACS-noLP is not sensitive to the increase of NEVs and is maintained at a low level. However, the running time of EACS-noPS and ACS increase with the number of NEVs and ACS runs the slowest.

Therefore, both pre-selection and local pruning help EACS to be more effective and efficient. Pre-selection helps EACS reduce runtime and local pruning helps improve the accuracy of solutions and further reduce runtime.

H. Energy and Carbon Savings

In the above sections, we focus on requests acceptance and average waiting time, so as to maximize customer satisfaction. In this section, geographical distance traveled by NEVs is concerned to study the energy and carbon emissions that can be saved by each algorithm.

Table X gives the geographical distance (in km) traveled by NEVs according to the dispatch results of each algorithm for problem D12-D15. The results show that EACS is the best approach that can obtain the dispatch solution with shortest geographical distance for all the dispatched NEVs.

According to [3], an electric vehicle will consume 0.36 kg/kWh of standard coal and generate 0.897 kg/kWh of carbon dioxide emissions during the fuel production phase, and consume 15 kWh per 100 km during the fuel use phase. By using this information and the results in Table X, we can calculate the standard coal consumption and carbon dioxide emissions according to the dispatch results of EACS, ACS, Greedy, and FCFS for problem D12-D15. The results are plotted and compared in Fig. 12(a) and (b), respectively. The results of GA and MA^3-LM are not plotted because they are too poor and are out of the range.

Fig. 12 shows that EACS can obtain the least standard coal consumption and carbon dioxide emissions while ensuring high quality of service to customers.

VI. Conclusion

Different from the traditional dispatch for fuel-driven vehicles, this paper proposes a dispatch framework oriented to

NEV and constructs an NEVD model, which considers other traffic factors such as the electricity of vehicles and charging piles compared to traditional dispatch. Two-level measurement of customer satisfaction is designed, which combines the acceptance rate of service requests and average waiting time. EACS, an efficient ACS-based algorithm, is proposed to solve this NP-hard problem. In EACS, pheromone is distributed between customers and NEVs, which represents the preference of one NEV is assigned to one customer. Moreover, pre-selection and local pruning strategies are integrated to speed up global search.

Extensive experimental tests are carried out in four scenarios, including both small and large physical areas, both resource-shortage and rich scenarios. The results show the effectiveness and advantage of the ACS-based algorithms. In the resource-shortage scenario, ACS has obvious advantages in terms of solution quality. In the resource-rich scenario, EACS can obtain higher user satisfaction within less runtime. Moreover, transportation cost, fuel consumption, and carbon emissions can be reduced by EACS. Future work will include dynamic dispatch of NEV by considering the real-time traffic data during the execution of EACS and implementing NEVD in a distributed way to further reduce the computation time. In this sense, dynamic optimization strategies [37] and distributed strategies [38] can be adopted to enhance the performance of EACS.

REFERENCES

[1] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in real-time spatial data: Experiments and analysis," in *Proc. VLDB Endowment*, 2016, pp. 1053–1064.

[2] Caocao. (Jul. 2019). *Introduction to Caocao Special Car*. [Online]. Available: https://www.caocaokeji.cn/

[3] J. Ye, X. Chen, and Y. Lv, "Design factors of new energy vehicles transportation planning," in *Proc. Int. Conf. Remote Sens., Environ. Transp. Eng.*, 2011, pp. 3963–3966.

[4] Z. Chen, L. Li, X. Hu, B. Yan, and C. Yang, "Temporal-difference learning-based stochastic energy management for plug-in hybrid electric buses," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, pp. 2378–2388, Jun. 2019. doi: 10.1109/TITS.2018.2869731.

[5] 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.

[6] H. Khayyam and A. Bab-Hadiashar, "Adaptive intelligent energy management system of plug-in hybrid electric vehicle," *Energy*, vol. 69, pp. 319–335, May 2014.

[7] 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.

[8] G. Wu, K. Boriboonsomsin, and M. J. Barth, "Development and evaluation of an intelligent energy-management strategy for plug-in hybrid electric vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 1091–1100, Jun. 2014.

[9] C. Hutson, G. K. Venayagamoorthy, and K. A. Corzine, "Intelligent scheduling of hybrid and electric vehicle storage capacity in a parking lot for profit maximization in grid power transactions," in *Proc. IEEE Energy Conf.*, Nov. 2008, pp. 1–8.

[10] M. Honarmand, A. Zakariazadeh, and S. Jadid, "Optimal scheduling of electric vehicles in an intelligent parking lot considering vehicle-to-grid concept and battery condition," *Energy*, vol. 65, pp. 572–579, Feb. 2014.

[11] Y. He, B. Venkatesh, and L. Guan, "Optimal scheduling for charging and discharging of electric vehicles," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1095–1105, Sep. 2012.

[12] W. Yao *et al.*, "A multi-objective collaborative planning strategy for integrated power distribution and electric vehicle charging systems," *IEEE Trans. Power Syst.*, vol. 29, no. 14, pp. 1811–1821, Jul. 2014.

[13] R. Wolbertus, M. Kroesen, R. van den Hoed, and C. Chorus, "Fully charged: An empirical study into the factors that influence connection times at EV-charging stations," *Energy Policy*, vol. 123, pp. 1–7, Dec. 2018.

[14] J. Asamer, M. Reinthaler, M. Ruthmair, M. Straub, and J. Puchinger, "Optimizing charging station locations for urban taxi providers," *Transp. Res. A, Policy Pract.*, vol. 85, pp. 233–246, Mar. 2016.

[15] Y. Li, P. Zhang, and Y. Wu, "Public recharging infrastructure location strategy for promoting electric vehicles: A bi-level programming approach," *J. Clean. Prod.*, vol. 172, pp. 2720–2734, Jan. 2018.

[16] X. Wang, C. Yuen, N. U. Hassan, N. An, and W. Wu, "Electric vehicle charging station placement for urban public bus systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 1, pp. 128–139, Jan. 2017.

[17] 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.

[18] Y. Wang, J. Jiang, and T. Mu, "Context-aware and energy-driven route optimization for fully electric vehicles via crowdsourcing," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1331–1345, Sep. 2013.

[19] M. M. de Weerdt, S. Stein, E. H. Gerding, V. Robu, and N. R. Jennings, "Intention-aware routing of electric vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1472–1482, May 2016.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] J. Jung, R. Jayakrishnan, and J. Y. Park, "Dynamic shared-taxi dispatch algorithm with hybrid-simulated annealing," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 31, no. 4, pp. 275–291, 2016.

[25] M. Adamczyk and D. Król, "Modelling of taxi dispatch problem using heuristic algorithms," in *Proc. Int. Conf. Multimedia Netw. Inf. Syst.*, 2018, pp. 170–179.

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

[27] T. Liao, K. Socha, M. A. M. 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.

[28] 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.

[29] Z.-H. Zhan *et al.*, "An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 399–412, Jun. 2010.

[30] 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.

[31] Z. H. Zhan, X. F. Liu, Y. J. Gong, J. Zhang, H. S. H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Comput. Surv.*, vol. 47, no. 4, Jul. 2015, Art. no. 63.

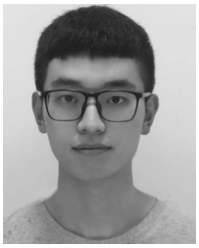
[32] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. L. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.

[33] X. F. Liu, Z. H. Zhan, and J. Zhang, "An energy aware unified ant colony system for dynamic virtual machine placement in cloud computing," *Energies*, vol. 10, no. 5, pp. 1–15, May 2017.

[34] F. Zheng, A. C. Zecchin, J. P. Newman, H. R. Maier, and G. C. Dandy, "An adaptive convergence-trajectory controlled ant colony optimization algorithm with application to water distribution system design problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 773–791, Oct. 2017.

[35] 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.

[36] (Jul. 2019). *BYD Auto*. [Online]. Available: https://en.wikipedia.org/wiki/BYD_Auto
[37] X. F. Liu *et al.*, "Neural network-based information transfer for dynamic optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. doi: 10.1109/TNNLS.2019.2920887.
[38] Z.-H. Zhan *et al.*, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.



Di Liang (S'18) received the B.S. degree in information security from the South China University of Technology, Guangzhou, China, in 2018, where he is currently pursuing the M.S. degree.

His current research interests include artificial intelligence, evolutionary computation, swarm intelligence, and their applications in the real world, such as intelligent transportation design.

Zhi-Hui Zhan (M'13–SM'18) received the bachelor's and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

From 2013 to 2015, he was a Lecturer and an Associate Professor with the Department of Computer Science, Sun Yat-sen University. Since 2016, he has been a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, where he is currently a Changjiang Scholar Young Professor and a Pearl River Scholar Young Professor. His current research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems, and in environments of cloud computing and big data.

Dr. Zhan's doctoral dissertation was awarded the China Computer Federation (CCF) Outstanding Ph.D. Dissertation and the IEEE Computational Intelligence Society (CIS) Outstanding Ph.D. Dissertation. He was a recipient of the Outstanding Youth Science Foundation from National Natural Science Foundations of China (NSFC) in 2018 and the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is an Associate Editor of the *Neurocomputing* and the *International Journal of Swarm Intelligence Research*.

Yanchun Zhang received the Ph.D. degree in computer science, University of Queensland, in 1991.

He is currently the Director of IT Program (Data Science and AI) with the Institute for Sustainable Industries and Liveable Cities, VU Research, Victoria University, Melbourne, VIC, Australia, and coordinates a multidisciplinary e-research program across Victoria University. He is also an International Research Leader in databases, data mining, health informatics, Web information systems, and Web services. He has published over 300 research articles in international journals and conference proceedings, including ACM *Transactions on Computer and Human Interaction* (TOCHI), the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), VLDBJ, SIGMOD, and ICDE conferences, and a dozen of books and journal special issues in the related areas.

Dr. Zhang is a Founding Editor and the Editor-in-Chief of *World Wide Web Journal* (Springer) and *Health Information Science and Systems Journal* (Springer). He is also the Chairman of the International Web Information Systems Engineering Society. He also holds a Professor position at Guangzhou University, China.

Jun Zhang (F'17) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high performance computing, operations research, and power electronic circuits.

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